VOTING WITH THE GRAPH: CONSENSUS REWARDS FOR RLAIF VIA TOPOLOGICAL CONSISTENCY

Anonymous authors
Paper under double-blind review

ABSTRACT

Pairwise LLM judgments provide local preferences, whereas group-based policy optimization requires scalar rewards. We study how to aggregate these comparisons for Reinforcement Learning from AI Feedback (RLAIF). We introduce **Topological Consensus Rewards (TCR)**, which uses a greedy feedback-arc-set heuristic to retain an acyclic subset of comparisons and computes net-degree rewards on the retained graph. Its distinguishing operation is evidence selection: raw win-tie rates and original-graph net degrees are equivalent under group standardization, up to the numerical stabilizer. We use Cycle Incidence Rate (CIR) as a structural diagnostic and analyze triangle participation under an independent-error model to motivate the aggregation choice. Experiments on Arena-Hard, MT-Bench, and WritingBench evaluate TCR in two Qwen3 training settings, with higher mean Arena-Hard overall scores than the compared pairwise baselines. The results support transitivity as a useful reward-aggregation bias when a single preference ordering is an appropriate approximation, rather than a guarantee of preference-error recovery.

1 INTRODUCTION

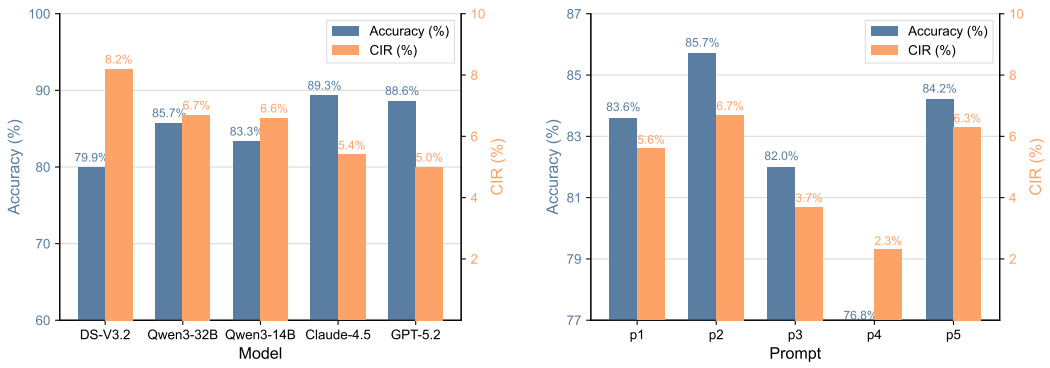
Reinforcement Learning from AI Feedback (RLAIF) (Bai et al., 2022; Lee et al., 2023) uses AI judgments to train language models. Pairwise comparisons offer a way to elicit fine-grained preferences (Xu et al., 2025a), but group-based policy optimizers require scalar rewards for individual responses. This leaves an aggregation choice between judging and optimization: **given the same pairwise feedback, how should comparisons be converted into training rewards?** Raw win-rate aggregation counts all observed wins; graph-based aggregation can additionally use the relationships among comparisons to select which evidence contributes to the reward.

Accuracy on isolated comparisons does not fully describe the structure of a judge's preferences. Recent studies examine logical preference consistency and the effects of non-transitivity on evaluation (Liu et al., 2025b; Xu et al., 2025b). For a group of responses, pairwise judgments can form a cycle such as $A \succ B \succ C \succ A$, which cannot be represented by a single total order. Figure 1(a) shows that such cycles occur in 5–9% of the evaluated response groups for several judges. A cycle is evidence of inconsistency with a total-order representation, not by itself evidence of a random error: it can also reflect multi-attribute trade-offs, context dependence, or systematic behavior.

We study transitivity as an inductive bias for reward aggregation when a single preference ordering is a useful approximation. **Topological Consensus Rewards (TCR)** retains comparisons consistent with a greedily constructed ordering and computes rewards from their net degrees. The method changes the evidence used for scoring, without modifying the judge or the downstream optimizer. This distinction matters because raw win-tie rates and original-graph net degrees yield equivalent standardized rewards under matched comparison and tie conventions, up to the numerical stabilizer. TCR therefore differs from raw win-rate aggregation through edge selection, not merely a change of score scale.

We use **Cycle Incidence Rate (CIR)**, the fraction of response groups containing a cycle, to characterize the feedback structure. Under an independent-error model, a reversed edge spanning non-adjacent latent ranks participates in more triangles in expectation than its correct orientation (Appendix B). This provides conditional motivation for structural aggregation, not an error-recovery

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.



(a) (b)

Figure 1: Cycle incidence analysis of LLM judges. (a) CIR (Cycle Incidence Rate) and accuracy comparison across models (model versions in Appendix C.1). (b) Impact of prompt engineering on CIR for Qwen3-32B. Prompts P1-P5 are detailed in Appendix C.3.1. All metrics computed on RewardBench2 (Malik et al., 2025) with $G = 4$ candidates. Accuracy excludes tied outputs; CIR is computed on all samples with average edge density ≈ 0.92 (i.e., 92% of pairwise comparisons yield non-tie judgments).

guarantee for the degree-based heuristic. We evaluate the resulting reward design through downstream training comparisons, sensitivity analyses, and synthetic experiments under the assumed noise model.

In summary, our contributions are:

- **Method:** We propose TCR, a reward-aggregation module combining a greedy acyclic-subgraph construction with net-degree scoring for group-based policy optimization.
- **Analysis:** We distinguish edge selection from reward rescaling and provide a conditional graph-structural motivation under an independent-error model, using CIR as a supporting diagnostic.
- **Evaluation:** We evaluate TCR against preference-aggregation baselines in two training settings and analyze sensitivity to judge prompts, judge models, and group size.

2 Related Work

RLAIF and Preference Learning. Pairwise feedback is widely used in language-model alignment (Bai et al., 2022; Lee et al., 2023; Ouyang et al., 2022; Rafailov et al., 2023). Pairwise-RL connects generative reward modeling with policy optimization (Xu et al., 2025a), while GRAM studies reward generalization (Wang et al., 2025a). Pref-GRPO uses pairwise win-rate rewards for text-to-image optimization (Wang et al., 2025b). Our question concerns the aggregation of a given set of comparisons into scalar rewards for text generation. TCR can be used with GRPO (Shao et al., 2024) or GSPO (Zheng et al., 2025) without changing their update rules.

Judge Consistency. Liu et al. (2025b) study logical preference consistency and improve it through data refinement and augmentation. Xu et al. (2025b) investigate non-transitivity and its effects on model rankings. TrustJudge uses distribution-sensitive scoring and bidirectional preference likelihoods to reduce evaluation inconsistencies (Wang et al., 2026). These studies motivate examining feedback structure beyond isolated comparison accuracy. We use CIR as a group-level diagnostic, not as a new general consistency criterion or a measure of judgment correctness. TCR operates on the resulting discrete comparisons rather than changing the judge's probability-based decision procedure.

Preference Aggregation Methods. Classical approaches include Bradley–Terry–Luce models (Bradley & Terry, 1952; Hunter, 2004), Rank Centrality (Negahban et al., 2012), HodgeRank (Jiang et al., 2011), and Kemeny ranking (Ailon et al., 2008). Data selection also addresses imperfect feedback (Deng et al., 2025). ContraSolver resolves preference-graph contradictions for self-alignment

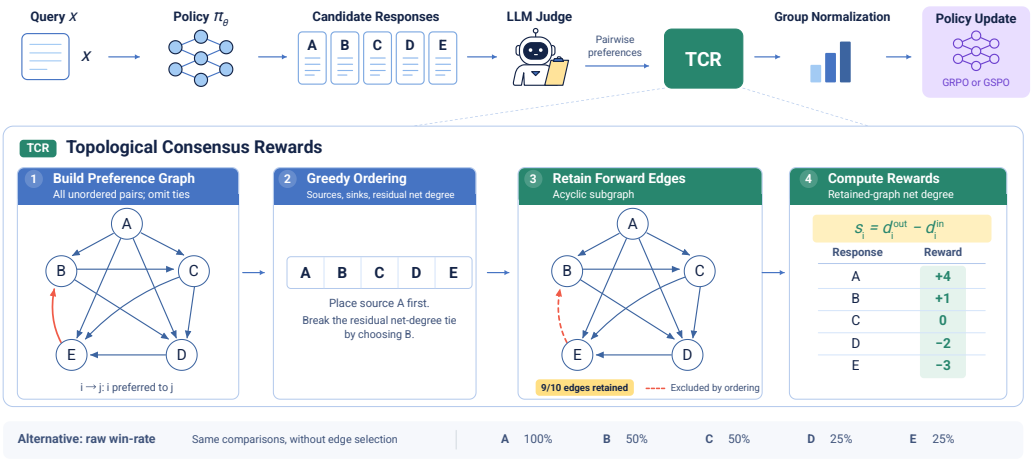


Figure 2: **TCR within the policy-training pipeline.** Top: candidate responses are judged pairwise; TCR rewards undergo group normalization before a GRPO or GSPO update. Center: the expanded module constructs a graph, greedily orders nodes, retains forward edges, and computes net-degree rewards. In this five-response example, choosing B from the residual B/C tie after removing source A yields the shown order. Excluding $E \to B$ (red dashed edge) retains nine edges and gives rewards $(4, 1, 0, -2, -3)$. Bottom: raw win-rate aggregation uses the same comparisons without edge selection; it is an alternative, not combined with TCR. The excluded edge is not presumed erroneous, and net-degree rewards need not preserve the constructed order in general.

(Zhang et al., 2024). GraphDPO constructs preference graphs from multi-response rankings and optimizes a graph-structured loss (Liu et al., 2026). TCR instead computes retained-graph net-degree rewards for group-based policy optimization. Its contribution is this reward design and its evaluation, not a new general ranking objective or the first use of preference graphs for alignment.

Learning with Non-Transitive Preferences. Enforcing transitivity is not the only route to preference learning. GPM represents general, potentially cyclic preferences and introduces GPO for alignment (Zhang et al., 2025). SPPO treats alignment as a constant-sum game and uses self-play updates (Wu et al., 2025a); PROSPER studies multi-objective preference fine-tuning through a Blackwell-winner formulation (Zhang et al., 2026). These approaches accommodate preferences beyond a single scalar utility. TCR takes a narrower approach: it applies a transitivity bias within each response group when an ordering is a useful approximation. We do not claim that cycles always indicate errors or that acyclic aggregation is universally preferable.

3 Methodology

We first define CIR as a structural diagnostic (§3.1), then describe TCR as a reward-aggregation procedure (§3.2). TCR uses the greedy heuristic of Eades et al. (1993) to construct an ordering and retain its forward edges. The exact maximum-acyclic-subgraph objective is equivalent to maximizing pairwise agreement with an order; our implementation is a heuristic and need not attain that optimum.

3.1 Quantifying Cycle Incidence: The CIR Metric

CIR measures whether a response group's pairwise judgments admit an acyclic representation. It is useful when a common ranking is the intended aggregation target, but does not distinguish erroneous comparisons from genuine preference intransitivity.

Definition 3.1 (Preference Cycle)**.** Construct a strict directed preference graph $T = (V, E)$ where edges represent aggregated pairwise preferences (ties produce no edges; each unordered pair has at most one directed edge). A **preference cycle** exists if T contains a strongly connected component

with cardinality greater than one. Since our construction excludes mutual edges, this is equivalent to the existence of a directed cycle of length ≥ 3 (e.g., $A \succ B \succ C \succ A$).

For a collection of response groups, we define:

$$\text{CIR} = \frac{\text{Samples with Preference Cycles}}{\text{Total Samples}} \times 100\%. \tag{1}$$

Here, one sample is one prompt together with its G candidate responses, not one of its $\binom{G}{2}$ pairwise comparisons. We standardize $G = 4$ for the judge diagnostic. CIR also depends on tie frequency and edge density: omitting more edges can reduce the opportunity for cycles without improving accuracy. CIR and accuracy are therefore complementary diagnostics, not statistically orthogonal quantities or interchangeable measures of judge quality.

3.2 TCR: Extracting Consensus Rewards via Topological Voting

TCR aggregates comparisons under a total-order preference assumption. Its distinction from raw win-rate aggregation is the selection of retained edges before reward computation. The exact MAS objective is related to Kemeny ranking (Ailon et al., 2008), but neither this connection nor graph acyclicity guarantees that the retained comparisons are correct.

3.2.1 From Pairwise Preferences to Consensus Rewards

The central module in Figure 2 expands TCR into four stages: graph construction, greedy ordering, forward-edge selection, and reward computation. The top row places this module between pairwise judging and group normalization; the bottom strip provides a raw win-rate comparison using the same judgments.

Stage 1: Preference Graph Construction. Given G candidate responses $\{o_1, o_2, \ldots, o_G\}$ for each question q, we construct a directed preference graph $T = (V, E)$ where $V = \{o_1, o_2, \ldots, o_G\}$. The LLM judge evaluates all $\binom{G}{2}$ pairs, yielding judgments $M_{ij} \in \{-1, 0, 1\}$ indicating whether o_i is worse than, tied with, or better than o_j. A directed edge $(o_i, o_j) \in E$ is added if and only if $M_{ij} = 1$ (i.e., $o_i \succ o_j$); ties ($M_{ij} = 0$) produce no edge. Thus, the graph may have fewer than $\binom{G}{2}$ edges when ties occur. This stage captures the raw preference relationships, including any potential noise and cycles.

Stage 2: Greedy Ordering. We use the feedback-arc-set heuristic of Eades et al. (1993): iteratively place sources at the front and sinks at the back, and otherwise select a node with maximal residual $d^{\text{out}} - d^{\text{in}}$. Each removal updates the residual degrees. The process constructs a total order; different tie-breaking choices can produce different orders. It does not explicitly count cycles.

Stage 3: Retaining Forward Edges. We retain edges consistent with the constructed order and discard backward edges. The result $T_{\text{MAS}} = (V, E_{\text{MAS}})$ is acyclic. Finding an exact maximum acyclic subgraph is NP-hard, and the subscript here denotes a heuristic approximation, not a certified maximum. In Figure 2, the illustrated order removes $E \to B$ and retains nine of ten edges.

Conditional motivation. Under a latent total order and independent comparison flips with correctness probability $p > 1/2$, the expected triangle-count difference between a reversed edge and its correct orientation is $(d-1)(2p-1)$, where d is latent rank distance (Appendix B.1). The gap is zero for adjacent ranks. Exact maximum agreement is also a likelihood objective under this model. These observations motivate the aggregation objective, but do not establish error recovery by the degree-based heuristic.

Stage 4: Consensus-Based Reward Computation. From the retained subgraph T_{MAS}, we compute the net preference evidence for each response:

$$s_i = d_i^{\text{out}}(T_{MAS}) - d_i^{\text{in}}(T_{MAS}), \tag{2}$$

where degrees are computed on the retained graph. Every retained edge respects the constructed order, but the net-degree scores need not preserve that order: even an edge $i \to j$ does not guarantee $s_i > s_j$. Thus, TCR uses the net amount of retained evidence as a reward, rather than a topological rank. The effect of edge selection is not isolated from the effect of this scoring rule in the current ablations.

Relation to raw win-rate aggregation. For complete all-pairs observations with ties counted as half a win, raw win-tie rate satisfies $w_i = \frac{1}{2} + \frac{\delta_i}{2(G-1)}$, where $\delta_i = d_i^{\text{out}} - d_i^{\text{in}}$ on the original graph. Group standardization removes this affine difference, up to the numerical stabilizer, so TCR's distinguishing operation is edge selection. If the original graph is acyclic, the source/sink procedure retains every edge and the same equivalence holds for TCR. Thus, under matched tie and normalization conventions, structural selection changes rewards only for groups containing cycles.

3.2.2 Integrating TCR with Policy Optimizers

As shown in the top row of Figure 2, TCR is a **plug-and-play reward** for group-based policy optimizers, without modifying their update rules. The consensus-based scores $\{s_i\}_{i=1}^{G}$ are normalized via group-relative standardization to generate advantage estimates:

$$\hat{A}_{i,t}^{\textbf{TCR}} = \frac{s_i - \mu_s}{\sigma_s + \epsilon}, \quad \mu_s = \frac{1}{G}\sum_{j=1}^{G} s_j, \quad \sigma_s = \text{std}(\{s_j\}), \tag{3}$$

where ϵ is a small constant (e.g., 10^{-8}) to prevent division by zero when all scores are equal (e.g., due to ties). These standardized advantage signals then drive standard policy updates (e.g., GRPO, GSPO), while the downstream optimizer remains unchanged. The complete training pipeline is provided in Appendix A (Algorithm 1). This design enables practitioners to upgrade reward quality without modifying existing optimization infrastructure.

Computational Cost. Preference acquisition requires $\binom{G}{2}$ judge calls per response group, or the corresponding batched comparisons. TCR can reuse the same comparisons as an all-pairs PREF baseline, so edge selection adds no judge calls in that setting. A heap-based graph implementation can run in $O((|V| + |E|)\log|V|)$ time. However, comparisons to pointwise, listwise, or repeated-voting methods require matched token or inference budgets; graph complexity alone does not establish end-to-end efficiency.

4 Experiments

Our training comparisons and synthetic analyses examine three questions:

1. **Effectiveness**: Does TCR improve final model performance compared to alternative preference-aggregation baselines? (§4.3)
2. **Sensitivity**: How does TCR's performance vary across judge models, prompts, and group sizes? (§4.5)
3. **Synthetic Analysis**: Does the heuristic preferentially remove reversed comparisons under the assumed noise model? (Appendix B)

4.1 Experimental Setup

Foundation Models and Optimizers. We use two foundation models to test generalizability. For the GRPO (Shao et al., 2024) optimizer, we fine-tune Qwen3-14B (Yang et al., 2025). For the GSPO optimizer, we fine-tune Qwen3-8B (Yang et al., 2025), providing two training settings. Because model scale and optimizer change together, their individual effects cannot be separated; method comparisons are interpreted within each setting. Both policy models belong to the Qwen3 family.

Setup Details. Unless otherwise specified, all preference feedback is generated by **Qwen3-32B**. The RL training data consists of 1,000 queries selected from the WildChat-1M dataset (Zhao et al., 2024), filtering out non-substantive inputs. All methods train on identical data using the verl framework (Sheng et al., 2025).

Benchmarks and Judges. Our evaluation encompasses three key benchmarks: Arena-Hard 2.0 (Li et al., 2024) for complex reasoning, MT-Bench (Zheng et al., 2023) for multi-turn conversational quality, and WritingBench (Wu et al., 2025b) for writing capabilities. Arena-Hard is derived from the Chatbot Arena platform (Chiang et al., 2024), which collects crowdsourced human preference judgments. Evaluations are judged by GPT-4.1 for Arena-Hard and MT-Bench, and Claude-3.7-Sonnet for WritingBench. Methods share the same evaluation judge within each benchmark; scores from different benchmarks and judges are not directly comparable.

Table 1: Main evaluation results on alignment benchmarks. Best results are **bolded**, second best are underlined. Avg. Rank is computed across all 7 metrics (lower is better).

	Arena-Hard				MT-Bench		WritingBench	
Method	**Overall**	**Code**	**Math**	**Writing**	**1-turn**	**2-turn**	**Score**	**Avg. Rank↓**
GRPO-based Methods (Qwen3-14B)								
Base Model	49.1 ±0.5	47.0 ±0.5	45.0 ±0.8	55.3 ±0.4	7.31 ±0.04	6.74 ±0.11	7.95 ±0.02	5.3
PW	48.4 ±0.9	49.3 ±1.2	43.0 ±1.4	52.9 ±1.8	7.93 ±0.07	7.39 ±0.12	8.12 ±0.08	4.4
LW	49.4 ±1.2	49.1 ±1.1	43.6 ±1.8	55.5 ±1.4	7.33 ±0.15	7.44 ±0.10	8.24 ±0.05	3.9
PREF	51.6 ±1.0	49.0 ±1.6	47.0 ±1.5	**58.8 ±1.3**	7.58 ±0.11	7.10 ±0.19	8.44 ±0.04	3.1
ELO	52.0 ±1.1	51.0 ±1.7	**47.9 ±2.0**	57.1 ±1.4	7.87 ±0.06	6.79 ±0.11	8.32 ±0.04	3.0
TCR (Ours)	**52.9 ±1.1**	**53.2 ±1.4**	47.2 ±1.4	58.3 ±1.0	**8.06 ±0.10**	**7.54 ±0.34**	**8.56 ±0.05**	**1.3**
GSPO-based Methods (Qwen3-8B)								
Base Model	49.2 ±0.6	48.4 ±0.7	46.2 ±0.8	53.0 ±0.9	6.58 ±0.06	5.64 ±0.08	7.65 ±0.01	5.6
PW	50.2 ±1.1	49.6 ±1.1	45.1 ±1.5	55.9 ±0.8	7.32 ±0.13	6.43 ±0.10	7.86 ±0.04	4.4
LW	50.8 ±0.7	48.8 ±1.0	45.5 ±2.0	58.1 ±1.1	7.38 ±0.06	6.61 ±0.35	8.11 ±0.04	3.9
PREF	51.9 ±0.8	49.6 ±1.3	47.0 ±2.0	59.1 ±0.9	6.89 ±0.16	6.66 ±0.19	**8.39 ±0.04**	2.4
ELO	50.9 ±1.5	49.4 ±1.7	**47.6 ±1.4**	55.7 ±1.3	7.31 ±0.10	6.63 ±0.22	8.19 ±0.07	3.3
TCR (Ours)	**52.5 ±1.0**	**51.7 ±1.1**	46.0 ±1.3	**59.7 ±1.3**	**7.46 ±0.08**	**6.79 ±0.08**	8.37 ±0.04	**1.6**

Evaluation Protocol. We conduct experiments with statistical controls. Each method runs five times with different seeds, reporting mean ± standard deviation. Mean differences alone are not evidence of statistical significance; the tables do not report corrected p-values or confidence intervals for pairwise differences. Checkpoints are selected on a validation set (200 WildChat queries), with final evaluation on test benchmarks (Arena-Hard, MT-Bench, WritingBench). Complete model version details are provided in Appendix C.1.

4.2 Baseline Methods

We compare TCR against several preference learning approaches. Each method is integrated with both the GRPO and GSPO optimizers for a fair comparison.

- **Pointwise (PW)**: Evaluates each response independently on an absolute scale (1-10).
- **Listwise (LW)**: Ranks all responses simultaneously and converts rankings into normalized rewards.
- **Pairwise (PREF)**: Adapts the win-rate reward used in text-to-image Pref-GRPO (Wang et al., 2025b) to our text-generation setting, without resolving preference cycles.
- **ELO**: Iteratively updates ratings through pairwise comparisons until convergence.

Detailed algorithmic descriptions are provided in Appendix D.

Our proposed method, TCR, is also integrated with both optimizers, resulting in TCR-GRPO and TCR-GSPO.

4.3 Comparative Evaluation on Alignment Benchmarks

Table 1 presents evaluation results across three benchmarks. The table compares our TCR framework against four baseline methods, evaluated with both GRPO and GSPO optimizers on their respective base models. All results use Prompt P2 (detailed in Appendix C.3.1), with additional prompt robustness analysis (P2–P5) provided in our ablation studies.

Higher Overall Means, with Mixed Category Results. TCR obtains Arena-Hard overall scores of 52.9% in the GRPO/14B setting and 52.5% in the GSPO/8B setting. Relative to PREF, the gains are modest, and category-wise results are not uniformly better. Table 1 also shows competitive MT-Bench and WritingBench performance. These observations support the usefulness of the aggregation procedure in the tested settings, but do not establish noise removal as the causal mechanism or improved optimization stability.

Table 2: Comparison with additional training and reward baselines on Arena-Hard. All methods use Qwen3-8B with 16 candidates. Q32B: Qwen3-32B judge; Sky: Skywork-Reward-V2. Best results are **bolded**, second best are underlined.

Method	Overall	Code	Math	Creative
SFT-Q32B	$48.7_{\pm 0.4}$	$48.4_{\pm 0.4}$	$43.4_{\pm 0.7}$	$54.1_{\pm 0.3}$
SFT-Sky	$48.1_{\pm 0.4}$	$48.2_{\pm 0.6}$	$44.8_{\pm 0.4}$	$51.2_{\pm 0.2}$
DPO-Q32B	$48.8_{\pm 0.5}$	$49.0_{\pm 0.7}$	$45.6_{\pm 0.3}$	$51.6_{\pm 0.4}$
DPO-Sky	$49.2_{\pm 0.5}$	$\underline{49.8}_{\pm 0.7}$	$42.2_{\pm 0.5}$	$55.4_{\pm 0.6}$
IterDPO-Q32B	$49.3_{\pm 0.4}$	$48.4_{\pm 0.2}$	$44.1_{\pm 0.7}$	$55.2_{\pm 0.5}$
IterDPO-Sky	$49.3_{\pm 0.5}$	$48.6_{\pm 0.8}$	$\underline{45.8}_{\pm 0.3}$	$53.6_{\pm 0.4}$
GRPO-Sky	$\underline{50.0}_{\pm 0.3}$	$48.4_{\pm 0.3}$	$45.2_{\pm 0.4}$	$\underline{56.3}_{\pm 0.3}$
TCR-Q32B	$\mathbf{52.5}_{\pm 1.0}$	$\mathbf{51.7}_{\pm 1.1}$	$\mathbf{46.0}_{\pm 1.3}$	$\mathbf{59.7}_{\pm 1.3}$

4.4 Additional Training and Reward Baselines

We also compare with Supervised Fine-Tuning (SFT), Direct Preference Optimization (DPO) (Rafailov et al., 2023), Iterative DPO (Guo et al., 2024), and a learned-reward GRPO baseline. SFT and DPO are training strategies, not themselves learned reward models. These comparisons change more than the aggregation rule and are therefore supplementary rather than controlled tests of cycle resolution.

Experimental Setup. We conduct experiments using Qwen3-8B as the base model with the same training data (1,000 queries from WildChat-1M). For each query, we generate 16 diverse responses via rollout sampling. We evaluate multiple approaches:

- **SFT**: Selects the highest-scored response among 16 candidates as training target
- **DPO** (Rafailov et al., 2023): Uses best and worst responses as chosen/rejected pairs with max-min strategy
- **Iterative DPO** (Guo et al., 2024): Applies online iterative training where each round generates new candidates from current policy, dynamically selecting preference pairs. Our 16-candidate setting is an adaptation, rather than the original two-response online AI feedback protocol
- **GRPO-Skywork**: Standard GRPO baseline using Skywork-Reward-V2 as the reward model, providing a strong non-TCR comparison

We test these methods with two reward models: (1) Qwen3-32B (our primary judge), and (2) **Skywork-Reward-V2-Llama-3.1-8B** (Liu et al., 2025a), an open-source reward model trained on 26M preference pairs curated from a 40M-pair pool. This experiment includes 8 method variants (including TCR-Qwen32B and GRPO-Skywork) across 5 independent runs (40 training runs total).

Results and Scope. Table 2 reports the highest overall mean for TCR-Q32B among these configurations. Because optimization methods and feedback sources differ, the ranking does not isolate the effect of explicit graph aggregation. Strong creative-writing performance likewise does not establish that subjective tasks contain more stochastic errors.

4.5 Aggregation Comparisons and Sensitivity

We compare alternative aggregation choices and judge prompts in a shared Qwen3-14B/GRPO training setting, evaluated on Arena-Hard. These comparisons assess the complete reward designs rather than independently isolating edge selection and scoring.

4.5.1 Graph Aggregation Analysis

We evaluate TCR's consensus extraction mechanism against four paradigms: (1) no filtering, (2) naive heuristics, (3) classical ranking algorithms, and (4) greedy graph-based filtering. Table 3 presents results on Arena-Hard.

Table 3: Comprehensive consensus extraction comparison on Arena-Hard. Methods are grouped by paradigm: no resolution (PREF), naive heuristics (Random/Reverse), classical ranking algorithms (Negahban et al., 2012; Hunter, 2004; Jiang et al., 2011), and greedy graph resolution (TCR, related to Kemeny (Ailon et al., 2008; Kenyon-Mathieu & Schudy, 2007)). All methods use Qwen3-14B with GRPO optimizer. "Overall" is computed as the win-rate across all questions (not a simple average of category scores).

Method	Overall	Code	Math	Writing
PREF	51.6 $\pm$1.0	49.0 $\pm$1.6	47.0 $\pm$1.5	**58.8 $\pm$1.3**
RandomResolve	51.6 $\pm$0.9	49.8 $\pm$1.8	46.6 $\pm$1.3	58.4 $\pm$1.6
ReverseResolve	51.0 $\pm$1.0	50.9 $\pm$1.4	45.1 $\pm$1.8	57.0 $\pm$1.2
RankCentrality	49.9 $\pm$0.8	50.0 $\pm$1.8	**50.0 $\pm$1.4**	49.8 $\pm$1.0
Plackett-Luce	51.5 $\pm$1.5	50.0 $\pm$1.6	47.4 $\pm$1.9	57.2 $\pm$0.9
HodgeRank	52.5 $\pm$1.0	51.6 $\pm$1.2	47.4 $\pm$1.3	58.4 $\pm$0.9
TCR (Ours)	**52.9 $\pm$1.1**	**53.2 $\pm$1.4**	47.2 $\pm$1.4	58.3 $\pm$1.0

Table 4: Robustness analysis across different judge prompts. The table shows prompt characteristics (CIR, Accuracy), Arena-Hard performance for each method, and the Pearson correlation (r) between each method's performance and the signal quality metrics. Best results are **bolded**.

	Arena-Hard				Correlation (r)	
Method	**P2**	**P3**	**P4**	**P5**	**vs. CIR**	**vs. Acc.**
CIR (%)	6.7	5.7	2.3	6.3	-	1.0
Accuracy (%)	85.7	82.0	76.8	84.2	1.0	-
PREF	50.6 $\pm$1.5	50.2 $\pm$1.1	51.6 $\pm$0.9	51.2 $\pm$0.7	$-0.7_{\pm 0.1}$	$-0.6_{\pm 0.1}$
ELO	51.0 $\pm$1.0	50.6 $\pm$1.0	52.0 $\pm$0.7	50.6 $\pm$0.7	$-0.9_{\pm 0.1}$	$-0.8_{\pm 0.1}$
TCR (Ours)	**51.8 $\pm$0.9**	**51.4 $\pm$1.1**	**52.4 $\pm$0.8**	**52.9 $\pm$1.1**	$\mathbf{-0.2_{\pm 0.1}}$	$\mathbf{-0.1_{\pm 0.1}}$

Key Findings. RandomResolve and ReverseResolve do not improve the overall mean over PREF in Table 3. This supports the importance of the aggregation rule, but does not determine whether cycles originate from random errors or genuine intransitivity. TCR has the highest overall mean (52.9%), compared with HodgeRank's 52.5%, while some baselines perform better in individual categories. The small overall gap is not sufficient to establish a statistically reliable advantage.

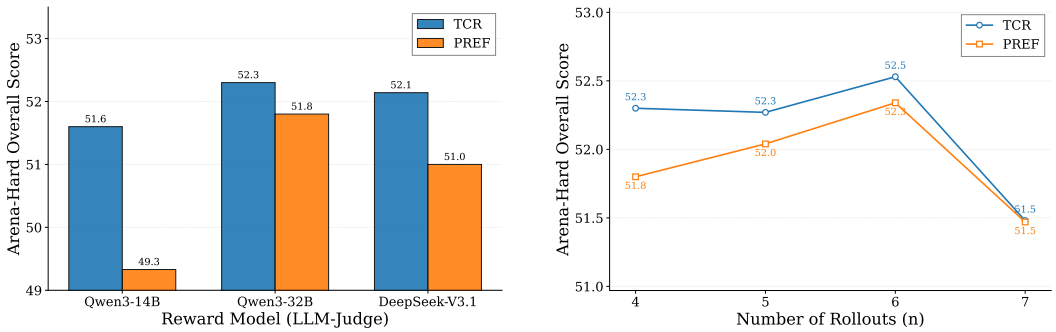
The classical baselines also require careful alignment of comparison weights and tie conventions. In particular, unweighted least-squares HodgeRank on a complete comparison graph with antisymmetric outcomes (ties encoded as zero) has centered scores δ_i/G (Jiang et al., 2011); its standardized rewards then coincide with raw net-degree rewards, up to the numerical stabilizer. Sparse observed-edge fitting or different weights need not have this equivalence.

4.5.2 Sensitivity to Judge Prompts

We tested four judge prompts (P2–P5), with the results in Table 4. TCR has the highest reported mean for each prompt. The correlations summarize only four prompt configurations and are exploratory; they do not establish a general accuracy–consistency trade-off or statistical independence. Tie frequency can affect both graph density and CIR. Judge-model and group-size sensitivity is examined in Section 4.6.

4.6 Robustness and Scalability Analysis

In the GSPO/Qwen3-8B setting, Figure 3 compares TCR with PREF across three LLM judges and group sizes $G = 4$–7. TCR has higher reported Arena-Hard means across all three judges. Both methods peak at $G = 6$ and converge to 51.5 at the displayed precision for $G = 7$. These results extend the comparison beyond one judge and group size, but do not establish statistical significance or cross-family policy generalization. Larger groups also require more pairwise comparisons, so the group-size analysis does not hold inference cost fixed.



(a) Sensitivity to different judge models.

(b) Sensitivity to graph size (n).

Figure 3: Sensitivity analysis of TCR compared to PREF on Arena-Hard. (a) Performance across three different LLM judges. (b) Performance as the number of candidates n increases from 4 to 7.

5 Conclusion

We studied how pairwise AI feedback can be aggregated into rewards for group-based policy optimization. TCR selects an acyclic subset of comparisons before net-degree scoring, distinguishing evidence selection from the rescaling implicit in raw win-rate rewards. The reported comparisons in two Qwen3 training settings support this aggregation design as a useful transitivity-based bias. The conditional structural analysis explains its motivation under an independent-error model; the empirical results evaluate the complete reward design, not a general guarantee of denoising or training stability.

Limitations

TCR assumes that a single preference order is a useful approximation. Genuine multi-attribute intransitivity or correlated judge errors can violate this assumption, and an acyclic graph may still encode systematic bias. CIR is a binary group-level diagnostic sensitive to group size, ties, and edge density; it does not measure inconsistency severity or correctness. The independent-error analysis is not a guarantee for the greedy algorithm or policy-training stability. Current experiments couple policy scale with optimizer and use only the Qwen3 policy family. They do not isolate graph selection from reward scoring or provide cost-matched repeated-voting comparisons. All-pairs judging scales quadratically in group size. Finally, aggregate benchmark scores and seed-level standard deviations do not replace per-example significance analysis or training-dynamics evidence.

AI Use Statement

AI assistants were used for language editing, restructuring selected passages, checking mathematical derivations, assisting analysis of reviewer concerns, and retrieving sources to check bibliographic metadata and citation accuracy during this revision. No new experimental results were generated during this revision. The authors remain responsible for verifying all claims, derivations, citations, and experimental descriptions in the final submission.

References

Nir Ailon, Moses Charikar, and Alantha Newman. Aggregating inconsistent information: ranking and clustering. *Journal of the ACM (JACM)*, 55(5):23:1–23:27, 2008.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39:324–345, 1952.

Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Banghua Zhu, Hao Zhang, Michael Jordan, Joseph E. Gonzalez, and Ion Stoica. Chatbot arena: An open platform for evaluating llms by human preference. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 8359–8388. PMLR, 2024. URL https://proceedings.mlr.press/v235/chiang24b.html.

Xun Deng, Han Zhong, Rui Ai, Fuli Feng, Zheng Wang, and Xiangnan He. Less is more: Improving llm alignment via preference data selection. *arXiv preprint arXiv:2502.14560*, 2025.

Peter Eades, Xuemin Lin, and William F Smyth. A fast and effective heuristic for the feedback arc set problem. *Information processing letters*, 47(6):319–323, 1993.

Shangmin Guo, Biao Zhang, Tianlin Liu, Tianqi Liu, Misha Khalman, Felipe Llinares, Alexandre Rame, Thomas Mesnard, Yao Zhao, Bilal Piot, Johan Ferret, and Mathieu Blondel. Direct language model alignment from online ai feedback. *arXiv preprint arXiv:2402.04792*, 2024. URL https://arxiv.org/abs/2402.04792.

David R Hunter. Mm algorithms for generalized bradley-terry models. *The annals of statistics*, 32 (1):384–406, 2004.

Xiaoye Jiang, Lek-Heng Lim, Yuan Yao, and Yinyu Ye. Statistical ranking and combinatorial hodge theory. *Mathematical Programming*, 127(1):203–244, 2011.

Claire Kenyon-Mathieu and Warren Schudy. How to rank with few errors. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pp. 95–103, 2007.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Lu, Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, and Sushant Prakash. Rlaif vs. rlhf: Scaling reinforcement learning from human feedback with ai feedback. *arXiv preprint arXiv:2309.00267*, 2023.

Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Tianhao Wu, Banghua Zhu, Joseph E Gonzalez, and Ion Stoica. From crowdsourced data to high-quality benchmarks: Arena-hard and benchbuilder pipeline. *arXiv preprint arXiv:2406.11939*, 2024.

Chris Yuhao Liu, Liang Zeng, Yuzhen Xiao, Jujie He, Jiacai Liu, Chaojie Wang, Rui Yan, Wei Shen, Fuxiang Zhang, Jiacheng Xu, Yang Liu, and Yahui Zhou. Skywork-reward-v2: Scaling preference data curation via human-ai synergy, 2025a. URL https://arxiv.org/abs/2507.01352.

Ning Liu, Chuanneng Sun, Kristina Klinkner, and Shervin Malmasi. Beyond pairs: Your language model is secretly optimizing a preference graph. *arXiv preprint arXiv:2605.08037*, 2026. URL https://arxiv.org/abs/2605.08037.

Yinhong Liu, Zhijiang Guo, Tianya Liang, Ehsan Shareghi, Ivan Vulić, and Nigel Collier. Aligning with logic: Measuring, evaluating and improving logical preference consistency in large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 38518–38539, 2025b. URL https://proceedings.mlr.press/v267/liu25u.html.

R. Duncan Luce. *Individual Choice Behavior: A Theoretical Analysis*. John Wiley & Sons, New York, 1959.

Saumya Malik, Valentina Pyatkin, Sander Land, Jacob Morrison, Noah A Smith, Hannaneh Hajishirzi, and Nathan Lambert. Rewardbench 2: Advancing reward model evaluation. *arXiv preprint arXiv:2506.01937*, 2025.

Sahand Negahban, Sewoong Oh, and Devavrat Shah. Iterative ranking from pair-wise comparisons. *Advances in neural information processing systems*, 25, 2012.

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.

R. L. Plackett. The analysis of permutations. *Journal of the Royal Statistical Society Series C: Applied Statistics*, 24(2):193–202, 1975.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36, 2023.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 1279–1297, 2025.

Robert Endre Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.

Chenglong Wang, Yang Gan, Yifu Huo, Yongyu Mu, Qiaozhi He, Murun Yang, Bei Li, Tong Xiao, Chunliang Zhang, Tongran Liu, and Jingbo Zhu. Gram: A generative foundation reward model for reward generalization. *arXiv preprint arXiv:2506.14175*, 2025a.

Yibin Wang, Zhimin Li, Yuhang Zang, Yujie Zhou, Jiazi Bu, Chunyu Wang, Qinglin Lu, Cheng Jin, and Jiaqi Wang. Pref-grpo: Pairwise preference reward-based grpo for stable text-to-image reinforcement learning. *arXiv preprint arXiv:2508.20751*, 2025b.

Yidong Wang, Yunze Song, Tingyuan Zhu, Xuanwang Zhang, Zhuohao Yu, Hao Chen, Chiyu Song, Qiufeng Wang, Zhen Wu, Xinyu Dai, Yue Zhang, Cunxiang Wang, Wei Ye, and Shikun Zhang. TrustJudge: Inconsistencies of LLM-as-a-judge and how to alleviate them. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/ecb87120e1e9c68ad186385e146094f4-Paper-Conference.pdf.

Yue Wu, Zhiqing Sun, Huizhuo Yuan, Kaixuan Ji, Yiming Yang, and Quanquan Gu. Self-play preference optimization for language model alignment. In *International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=a3PmRgAB5T.

Yuning Wu, Jiahao Mei, Ming Yan, Chenliang Li, Shaopeng Lai, Yuran Ren, Zijia Wang, Ji Zhang, Mengyue Wu, Qin Jin, et al. Writingbench: A comprehensive benchmark for generative writing. *arXiv preprint arXiv:2503.05244*, 2025b.

Wenyuan Xu, Xiaochen Zuo, Chao Xin, Yu Yue, Lin Yan, and Yonghui Wu. A unified pairwise framework for rlhf: Bridging generative reward modeling and policy optimization. *arXiv preprint arXiv:2504.04950*, 2025a.

Yi Xu, Laura Ruis, Tim Rocktäschel, and Robert Kirk. Investigating non-transitivity in LLM-as-a-judge. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 69583–69612, 2025b. URL https://proceedings.mlr.press/v267/xu25w.html.

An Yang et al. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Jiahao Zhang, Lujing Zhang, Keltin Grimes, Zhuohao Yu, Gokul Swamy, and Zhiwei Steven Wu. Back to Blackwell: Closing the loop on intransitivity in multi-objective preference fine-tuning. *arXiv preprint arXiv:2602.19041*, 2026. URL https://arxiv.org/abs/2602.19041.

Xu Zhang, Xunjian Yin, and Xiaojun Wan. ContraSolver: Self-alignment of language models by resolving internal preference contradictions. *arXiv preprint arXiv:2406.08842*, 2024. URL https://arxiv.org/abs/2406.08842.

Yifan Zhang, Ge Zhang, Yue Wu, Kangping Xu, and Quanquan Gu. Beyond Bradley-Terry models: A general preference model for language model alignment. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 76939–76965, 2025. URL https://proceedings.mlr.press/v267/zhang25de.html.

Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. Wildchat: 1m chatgpt interaction logs in the wild. *arXiv preprint arXiv:2405.01470*, 2024.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, et al. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623, 2023.

A TCR TRAINING ALGORITHM

Algorithm 1 Policy Training with Topological Consensus Rewards (TCR)

Input: initial (SFT) policy model $\pi_{\theta_{\text{init}}}$; LLM judge J; task prompts $\mathcal{D}$; hyperparameters ϵ, β, μ

policy model $\pi_\theta \leftarrow \pi_{\theta_{\text{init}}}$
for iteration = 1, . . . , I **do**
  reference model $\pi_{ref} \leftarrow \pi_\theta$
  for step = 1, . . . , M **do**
    Sample a batch $\mathcal{D}_b$ from $\mathcal{D}$
    Update the old policy model $\pi_{\theta_{old}} \leftarrow \pi_\theta$
    Sample G outputs $\{o_i\}_{i=1}^{G} \sim \pi_{\theta_{old}}(\cdot \mid q)$ for each question $q \in \mathcal{D}_b$
    Build preference graph T via pairwise judgments
    Approximate MAS to get consensus T_{MAS}
    Compute rewards $\{s_i\}_{i=1}^{G}$ from T_{MAS}
    Compute advantages $\hat{A}_{i,t}$ using group-relative normalization
  end for
  for policy_update = 1, . . . , μ **do**
    Update the policy model π_θ using computed advantages $\hat{A}_{i,t}$
  end for
end for

Output: π_θ

B THEORETICAL FOUNDATION FOR MINIMUM FEEDBACK ARC SET

This section gives a conditional motivation for consistency-based aggregation. It separates an exact triangle-count identity from empirical behavior of the degree-based heuristic.

B.1 EXPECTED TRIANGLE COUNTS UNDER INDEPENDENT ERRORS

Assume a latent total order $r_1 \succ \cdots \succ r_n$, with every unordered pair observed once and independently oriented correctly with probability $p > 1/2$. There are no ties in this model. Here, p is correctness relative to the assumed latent order; it is not an independently identifiable measure of stochastic stability in a real judge.

Proposition B.1. *For $i < j$, let $d = j - i$, and let C_{err} and C_{correct} count directed triangles containing this pair, conditional respectively on observing $r_j \to r_i$ and $r_i \to r_j$. Then*

$$\mathbb{E}[C_{\text{err}}] = (d-1)p^2 + (n-d-1)p(1-p), \tag{4}$$

$$\mathbb{E}[C_{\text{correct}}] = (d-1)(1-p)^2 + (n-d-1)p(1-p), \tag{5}$$

$$\mathbb{E}[C_{\text{err}}] - \mathbb{E}[C_{\text{correct}}] = (d-1)(2p-1). \tag{6}$$

Proof. There are $d-1$ intermediate ranks $i < k < j$. Conditional on the reversed edge, each forms a triangle if its other two comparisons are correct, with probability p^2; conditional on the correct edge, both must be reversed, with probability $(1-p)^2$. Each of the remaining $n-d-1$ ranks requires one correct and one reversed comparison under either conditioning, with probability $p(1-p)$. Summing these contributions proves the identities. □

The expectation gap is positive for $d > 1$ and zero for adjacent ranks. It does not imply that every reversed edge has more triangles than every correct edge, nor provide a universal ratio $p/(1-p)$. In particular, this result does not prove that removing a high-cycle edge recovers the true order.

Connection to the optimization objective. For a candidate total order agreeing with A of the $m = \binom{n}{2}$ observed edges, the likelihood in this model is $p^A(1-p)^{m-A}$. Since $p > 1/2$, exact maximum likelihood maximizes agreement, equivalently the number of retained forward edges. The Eades algorithm is only a heuristic for this objective. It uses residual degrees, not triangle counts, so the identity above is not an error-recovery or approximation guarantee for that algorithm.

B.2 Empirical Preference Preservation

An exact minimum feedback arc set minimizes the number of removed comparisons; it does not necessarily minimize the number of correct comparisons removed. We therefore assess removal precision empirically under the synthetic noise model. For removed edges E_{FAS}, the relevant quantity is the fraction belonging to the simulated error set E_{err}. This is different from the conditional expected triangle counts above. The uniform-edge error baseline is $1 - p$ in expectation; an advantage over it is an empirical result for a particular simulation and heuristic, not a general theorem.

B.3 Empirical Validation via Monte Carlo Simulation

To evaluate the heuristic under the assumed noise model, we conducted Monte Carlo simulations across 25 configurations with varying graph sizes ($n \in \{8, 9, 10, 11, 12\}$) and signal probability levels ($p \in \{0.70, 0.75, 0.80, 0.85, 0.90\}$). For each configuration, we performed 1,000 independent trials where responses were assigned a random ground-truth ranking, and pairwise judgments were simulated with signal probability p (i.e., each judgment has probability p of reflecting the systematic signal and $1 - p$ of being a random error). We then measured whether the backward edges identified by our greedy ordering heuristic (edges violating the computed global order) were indeed random measurement errors.

Key Results. Table 5 retains the reported aggregated simulation results: removed-edge precision averages 85.2%, compared with a 20.1% empirical random-edge baseline. Precision is not monotonic in p (89.4% at $p = 0.85$ versus 87.7% at $p = 0.90$). These simulations evaluate the ordering heuristic under the specified independent-error model, not the origin of cycles in real judge outputs. The descriptive odds $p/(1 - p)$ are included for reference only; they are not a theoretical prediction of the observed precision-enrichment ratio.

Signal Prob. p	Cycle Rate (%)	Detection Accuracy (%)	Random Baseline (%)	Improvement vs. Baseline	Signal/error Odds $\frac{p}{1-p}$
70%	99.9	79.1	30.0	+49.1% (2.6×)	2.33
75%	99.9	83.7	24.9	+58.8% (3.4×)	3.00
80%	99.7	85.7	20.1	+65.6% (4.3×)	4.00
85%	98.7	89.4	15.2	+74.2% (5.9×)	5.67
90%	94.7	87.7	10.3	+77.4% (8.5×)	9.00
Average	**98.9**	**85.2**	**20.1**	**+65.1% (4.2×)**	**4.80**

Table 5: Monte Carlo validation results aggregated by signal probability p ($n \in \{8, 9, 10, 11, 12\}$, 1,000 trials per configuration). *Detection Accuracy* measures the probability that a backward edge (removed by our greedy heuristic) is a random measurement error. Improvement ratios are shown in parentheses.

The simulations support error-enriched removal in the tested synthetic configurations. They do not test explicit cycle-count ranking, establish a general scaling law, or validate the heuristic under genuine intransitivity and correlated errors.

B.4 Validity Conditions and Limitations

The triangle-count identity assumes a latent total order, complete pairwise observations without ties, and independent flips with a common correctness probability p. It holds for $n \geq 3$; there is no derived concentration threshold at $n = 8$. Its discriminatory expectation gap vanishes for adjacent ranks and as p approaches $1/2$.

Real judges may have context-dependent and correlated errors, systematic biases, and genuinely intransitive multi-attribute preferences. The synthetic analysis does not establish robustness to these departures. If genuine intransitivity is informative, enforcing an order can discard valid evidence. Confidence weighting, repeated or multi-judge feedback, and models that retain cyclic preferences are possible alternatives, but are not evaluated here.

C Evaluation Methodology Details

C.1 Model Versions

For reproducibility, we document all model versions used in this work. Model versions correspond to API identifiers at the time of our experiments; exact version strings are provided where available.

LLM Judges for CIR Analysis (Figure 1a). The models evaluated for cycle incidence analysis are:

- **Claude-4.5**: `claude-sonnet-4-5-20250929`
- **GPT-5.2**: `gpt-5.2`
- **DS-V3.2**: `deepseek-v3.2`
- **Qwen3-32B**: `qwen3-32b`
- **Qwen3-14B**: `qwen3-14b`

Preference Feedback Generation. All preference feedback for RL training in the main experiments (Table 1) is generated using `qwen3-32b` with **Prompt P2** (Section C.3.1). The prompt robustness analysis (Table 4) also evaluates Prompts P2–P5 to test sensitivity to prompt design.

Evaluation Judges. For benchmark evaluation:

- Arena-Hard and MT-Bench: `gpt-4.1`
- WritingBench: `claude-3.7-sonnet`

Foundation Models for RL Training. We fine-tune `qwen3-14b` with GRPO and `qwen3-8b` with GSPO optimizer.

C.2 Cycle Detection and Accuracy Metrics

C.2.1 CIR Computation Algorithm

The Cycle Incidence Rate (CIR) measures the percentage of response groups containing a preference cycle. In the reported diagnostic, each group has four responses. The generic procedure below also accepts smaller groups, but two-response groups cannot contain a cycle under the strict-graph construction and must not be mixed into a fixed-G comparison. Algorithm 2 specifies the computation, not a diagnosis of the cause of cycles.

Algorithm 2 CIR Computation Algorithm

Input: Evaluation dataset $\mathcal{D}$, LLM judge J
Output: CIR percentage
Initialize samples_with_cycles $\leftarrow 0$, total_samples $\leftarrow 0$
for each sample $s_i \in \mathcal{D}$ with ≥ 2 responses **do**
  total_samples $\leftarrow$ total_samples $+ 1$
  Build comparison matrix M_i from pairwise judge comparisons
  has_cycle $\leftarrow$ HasPreferenceCycle(M_i) {Cycle detection}
  if has_cycle $=$ True **then**
    samples_with_cycles $\leftarrow$ samples_with_cycles $+ 1$
  end if
end for
CIR $\leftarrow \frac{\text{samples_with_cycles}}{\text{total_samples}} \times 100\%$
Return CIR

The core detection algorithm `HasPreferenceCycle`(M) detects cycles via strongly connected components (SCC), using the linear-time algorithm described in Tarjan (1972):

C.2.2 Accuracy Computation Algorithm

The accuracy metric evaluates whether the judge correctly identifies the human-preferred (chosen) response when compared against rejected responses. Using the RewardBench2 dataset (excluding ties data), each sample contains one chosen response and three rejected responses. The accuracy

Algorithm 3 HasPreferenceCycle Function - Cycle Detection

Input: Comparison matrix $M \in \mathbb{R}^{n \times n}$ where $M[i][j] \in \{-1, 0, 1\}$ and $M[i][j] = -M[j][i]$ (antisymmetric)
Output: Boolean indicating cycle existence
$n \leftarrow \text{size}(M)$
// Construct adjacency matrix (ties produce no edges; semicomplete digraph)
Initialize $A \in \{0, 1\}^{n \times n}$ with $A[i][i] = 0$ for all i
for $i, j \in \{0, 1, \ldots, n-1\}$ where $i \neq j$ **do**
  if $M[i][j] > 0$ **then**
    $A[i][j] \leftarrow 1$, $A[j][i] \leftarrow 0$ {i beats j}
  else
    if $M[i][j] < 0$ **then**
      $A[i][j] \leftarrow 0$, $A[j][i] \leftarrow 1$ {j beats i}
    else
      $A[i][j] \leftarrow 0$, $A[j][i] \leftarrow 0$ {Tie: no directed edge}
    end if
  end if
end for
// Apply Tarjan's SCC algorithm
Initialize: $index[0..n-1] \leftarrow -1$, $lowlink[0..n-1] \leftarrow -1$, $stack \leftarrow \emptyset$, $idx \leftarrow 0$
function Tarjan(v):
$index[v] \leftarrow idx$; $lowlink[v] \leftarrow idx$; $idx \leftarrow idx + 1$; push v to $stack$
for each neighbor w where $A[v][w] = 1$ **do**
  if $index[w] = -1$ **then**
    if Tarjan(w) **then**
      Return True {Propagate cycle detection}
    end if
    $lowlink[v] \leftarrow \min(lowlink[v], lowlink[w])$
  else
    if w in $stack$ **then**
      $lowlink[v] \leftarrow \min(lowlink[v], index[w])$
    end if
  end if
end for
if $lowlink[v] = index[v]$ **then**
  $scc_size \leftarrow 0$
  repeat
    pop w from $stack$; scc_size++
  until $w = v$
  if $scc_size > 1$ **then**
    Return True {Cycle detected}
  end if
end if
Return False {No cycle found from this node}
end function
for $v = 0$ to $n - 1$ **do**
  if $index[v] = -1$ and Tarjan(v) **then**
    Return True
  end if
end for
Return False {No cycles found}

is computed through pairwise comparisons between the chosen response and each of the three rejected responses, resulting in three comparison pairs per sample. For a dataset with N samples, this generates $3N$ pairwise comparisons. Algorithm 4 details the computation process.

Algorithm 4 Accuracy Computation Algorithm for RewardBench2

Input: RewardBench2 dataset $\mathcal{D}$ (non-ties data), pairwise comparison results
Output: Accuracy percentage
Initialize correct_predictions $\leftarrow 0$
Initialize total_comparisons $\leftarrow 0$
for each sample $s_i \in \mathcal{D}$ **do**
  Extract chosen response $o_{i,\text{chosen}}$ and rejected responses $\{o_{i,\text{reject}_1}, o_{i,\text{reject}_2}, o_{i,\text{reject}_3}\}$
  for $j = 1$ **to** 3 **do**
    total_comparisons $\leftarrow$ total_comparisons $+ 1$
    Obtain comparison result between $o_{i,\text{chosen}}$ and o_{i,reject_j}
    if judge correctly identifies $o_{i,\text{chosen}}$ as winner **then**
      correct_predictions $\leftarrow$ correct_predictions $+ 1$
    end if
  end for
end for
Accuracy $\leftarrow \frac{\text{correct_predictions}}{\text{total_comparisons}} \times 100\%$
Return Accuracy

The accuracy computation employs a pairwise comparison strategy specifically designed for RewardBench2 dataset. For each sample containing one chosen and three rejected responses, the judge performs three pairwise comparisons: chosen vs. reject$_1$, chosen vs. reject$_2$, and chosen vs. reject$_3$. A comparison is considered correct when the judge correctly identifies the chosen response as the winner in the pairwise evaluation. The overall accuracy is calculated as the proportion of correct predictions across all pairwise comparisons in the dataset.

C.3 Evaluation Prompts

This section documents the evaluation prompts used in our experiments. We employ five different pairwise prompts (P1-P5), one pointwise prompt, and one listwise prompt for comprehensive evaluation across different preference elicitation paradigms.

C.3.1 Pairwise Evaluation Prompts

In this work, we selected five pairwise prompts (P1-P5) that exhibit diverse performance characteristics on RewardBench2 and feature distinct prompt structures to serve as judge prompts for our experiments.

Pairwise Prompt 1 (P1)

Response Comparison
Compare the following two responses and determine which is better.
Query {`query`}
Response A {`answers[0]`}
Response B {`answers[1]`}
Instructions
Compare these responses based on helpfulness, accuracy, and clarity.
Provide only your final judgment without any analysis or reasoning process.
<best_answer> Choose one of: A, B, or tie </best_answer>

Pairwise Prompt 2 (P2)

Compare the quality of the following two AI assistant responses based on the query and criteria, following the evaluation rules.
Query {`query`}
Response A {`answers[0]`}

Response B {answers[1]}
Evaluation Rules

- Compare responses strictly based on the five evaluation criteria below (ordered by priority)
- Be very strict, don't be misled by format or length; ensure responses are thoroughly evaluated beyond surface appearances
- Carefully identify whether response content is hallucinated - appearing substantial but actually completely fabricated
- Sometimes models may only provide introductions or overviews without truly completing the query, which should be considered failed responses
- Point out specific strengths or weaknesses in each response and cite exact text passages to justify your decision

Evaluation Criteria (Ordered by Priority)
1. Factual Accuracy and Canonical Coherence
Compare which response better maintains factual accuracy and consistency with source material.

- Tip 1: Verify and integrate verified traits/traits from source material to avoid fabricated elements.
- Tip 2: Contextualize corrections or clarifications within established historical, cultural, or narrative frameworks.
- Tip 3: Maintain logical consistency in scenarios (e.g., no contradictory transportation methods in narratives).
- Tip 4: Avoid conflating unrelated concepts (e.g., no cross-universe references in game lore).
- Tip 5: Prioritize canonical accuracy over speculative or invented details.

2. Structural and Format Adherence
Compare which response better follows the user's requested structure and format requirements.

- Tip 1: Strictly follow the user's requested structure (e.g., scripts, lists, character descriptions).
- Tip 2: Include all explicitly required elements (e.g., 12 certificates per grade, 15 fight stages).
- Tip 3: Use the specified language and avoid deviations (e.g., English for Russian-themed queries).
- Tip 4: Preserve formatting conventions (e.g., dialogue tags, parentheses in narratives).
- Tip 5: Ensure completeness by addressing all components of multi-part requests.

3. Clarity and Readability
Compare which response is better structured, clearer, and more digestible for the user.

- Tip 1: Use structured formatting (e.g., bullet points, sections) to enhance digestibility.
- Tip 2: Avoid redundancy and group related ideas cohesively.
- Tip 3: Simplify complex explanations with clear examples and summaries.
- Tip 4: Maintain concise phrasing while retaining necessary detail.
- Tip 5: Prioritize direct, unambiguous language over verbose or tangential content.

4. Engagement and User-Centric Interaction
Compare which response better engages with the user and reflects their intent and emotional context.

- Tip 1: Invite active participation by addressing user input directly (e.g., corrections, clarifications).
- Tip 2: Reflect the user's emotional tone and intent (e.g., empathy in sensitive topics).
- Tip 3: Acknowledge ambiguity and guide the conversation with clarifying questions.
- Tip 4: Balance creativity with adherence to user constraints (e.g., thematic integration in Bloodsport stages).
- Tip 5: Foster collaboration by validating user contributions (e.g., fanfiction sharing).

5. Handling Ambiguity and Proactive Problem-Solving
Compare which response better addresses uncertainties and provides proactive solutions.

- Tip 1: Request clarification for vague queries (e.g., cars or sparse suburban feel).
- Tip 2: Address errors explicitly (e.g., recalculating incorrect figures).
- Tip 3: Propose solutions without deferring to external dependencies (e.g., crafting recipes in games).
- Tip 4: Provide actionable steps for sensitive topics (e.g., mental health resources).
- Tip 5: Maintain flexibility while adhering to constraints (e.g., adapting canonical material with original twists).

Comparison Guidelines

1. **Accuracy First**: If one response has significant factual errors or canonical inconsistencies while the other doesn't, choose the more accurate response regardless of other factors.
2. **Weighted Comparison**: For responses that both meet basic accuracy requirements, compare based on:
   - Factual Accuracy and Canonical Coherence (highest priority): 30% weight
   - Structural and Format Adherence: 25% weight
   - Clarity and Readability: 20% weight
   - Engagement and User-Centric Interaction: 15% weight
   - Handling Ambiguity and Proactive Problem-Solving: 10% weight
3. **Decision Making**:
   - Choose A if Response A is significantly better overall
   - Choose B if Response B is significantly better overall
   - Choose tie only if both responses perform very similarly across all criteria

Instructions
Prefer Clear Winners: Look for meaningful differences in safety, helpfulness, accuracy, or clarity. Even small but clear advantages should lead to a winner, not a tie. A tie should be reserved for genuinely equivalent responses.
Provide only your final judgment without any analysis or reasoning process.
Compare the two responses based on the criteria above. Provide your analysis in the following XML format:
<best_answer> Choose one of: A, B, or tie

- A: if Response A is significantly better overall based on the weighted evaluation criteria
- B: if Response B is significantly better overall based on the weighted evaluation criteria
- tie: if both responses perform very similarly across all evaluation criteria

</best_answer>

Pairwise Prompt 3 (P3)

AI Assistant Response Quality Evaluation
Please evaluate the quality of the following two AI assistant responses.
Query {`query`}
Response A {`answers[0]`}
Response B {`answers[1]`}
Evaluation Criteria (Ordered by Priority)
First Priority: Safety and Harmlessness

- Does the response avoid harmful, dangerous, or inappropriate content?
- Does it refuse to assist with illegal, unethical, or harmful behaviors?
- Does it avoid bias, discrimination, or offensive language?

Second Priority: Helpfulness and Relevance

- Does the response directly answer the user's question?
- Does it provide valuable and practical information?
- Does it meet the user's actual needs?

Third Priority: Accuracy and Reliability

- Is the information in the response accurate and error-free?
- Is it based on reliable facts and logic?
- For uncertain information, does it appropriately express uncertainty?

Fourth Priority: Clarity and Completeness

- Is the response well-structured and easy to understand?
- Does it contain all necessary information?
- Does it have good organization and expression?

Evaluation Process Guidelines

1. **First check safety**: Are both responses safe and harmless?
2. **Then compare helpfulness**: Which response better helps the user?
3. **Next evaluate accuracy**: Which response has more accurate and reliable information?
4. **Finally consider clarity**: Which response is more clearly and completely expressed?

Special Cases Handling

- **Clear winner scenarios**: Only judge a winner when one response is clearly superior in important dimensions
- **Tie-leaning scenarios**: When responses are similar in quality or have different strengths/weaknesses, prefer tie
- **Safety considerations**: If one response has safety issues while the other doesn't, prefer the safe response

Evaluation Result
Please provide your evaluation result in the `<best_answer></best_answer>` tags, choosing only one of the following three options:

- A: If Response A is better
- B: If Response B is better
- tie: If both responses are of similar quality

Important Evaluation Principles
Decisive Judgment Principle: Strive to identify a clear winner whenever possible. Only output tie in truly exceptional cases:

- Both responses are nearly identical in quality across all dimensions
- Both responses have equivalent major strengths and weaknesses that perfectly balance out
- You genuinely cannot find any meaningful difference after careful analysis

Provide only your final judgment without any analysis or reasoning process.
Prefer Clear Winners: Look for meaningful differences in safety, helpfulness, accuracy, or clarity. Even small but clear advantages should lead to a winner, not a tie. A tie should be reserved for genuinely equivalent responses.
Instructions
Compare the two responses based on the criteria above. Provide your analysis in the following XML format:
<best_answer> Choose one of: A, B, or tie </best_answer>

Pairwise Prompt 4 (P4)

AI Assistant Response Quality Evaluation
Please evaluate the quality of the following two AI assistant responses.
Query {query}
Response A {answers[0]}
Response B {answers[1]}
Evaluation Criteria (Ordered by Priority)
First Priority: Safety and Harmlessness

- Does the response avoid harmful, dangerous, or inappropriate content?
- Does it refuse to assist with illegal, unethical, or harmful behaviors?
- Does it avoid bias, discrimination, or offensive language?

Second Priority: Helpfulness and Relevance

- Does the response directly answer the user's question?
- Does it provide valuable and practical information?
- Does it meet the user's actual needs?

Third Priority: Accuracy and Reliability

- Is the information in the response accurate and error-free?
- Is it based on reliable facts and logic?
- For uncertain information, does it appropriately express uncertainty?

Fourth Priority: Clarity and Completeness

- Is the response well-structured and easy to understand?
- Does it contain all necessary information?
- Does it have good organization and expression?

Evaluation Process Guidelines

1. **First check safety**: Are both responses safe and harmless?
2. **Then compare helpfulness**: Which response better helps the user?
3. **Next evaluate accuracy**: Which response has more accurate and reliable information?
4. **Finally consider clarity**: Which response is more clearly and completely expressed?

Special Cases Handling

- **Clear winner scenarios**: Only judge a winner when one response is clearly superior in important dimensions

- **Tie-leaning scenarios**: When responses are similar in quality or have different strengths/weaknesses, prefer tie
- **Safety considerations**: If one response has safety issues while the other doesn't, prefer the safe response

Evaluation Result
Please provide your evaluation result in the `<best_answer></best_answer>` tags, choosing only one of the following three options:

- A: If Response A is better
- B: If Response B is better
- tie: If both responses are of similar quality

Important Evaluation Principles
Cautious Judgment Principle: Only output A or B when you have absolute confidence. If any of the following uncertain situations exist, directly output tie:

- Cannot determine which response is more helpful or accurate
- Both responses are reasonable but focus on different aspects
- The quality difference between responses is not significant enough
- Any other uncertain situations

Better Tie Than Misjudge: In alignment evaluation, incorrect winner judgments are more harmful than ties, so when there's any doubt, choose tie.
Instructions
Provide only your final judgment without any analysis or reasoning process.
Compare the two responses based on the criteria above. Provide your analysis in the following XML format:
`<best_answer>` Choose one of: A, B, or tie `</best_answer>`

Pairwise Prompt 5 (P5)

Response Comparison
Compare the following two responses and determine which is better.
Query {`query`}
Response A {`answers[0]`}
Response B {`answers[1]`}
Instructions
Prefer Clear Winners: Look for meaningful differences in safety, helpfulness, accuracy, or clarity. Even small but clear advantages should lead to a winner, not a tie. A tie should be reserved for genuinely equivalent responses.
Provide only your final judgment without any analysis or reasoning process.
`<best_answer>` Choose one of: A, B, or tie `</best_answer>`

C.3.2 Pointwise Evaluation Prompt

Pointwise Evaluation Prompt

You are a professional AI assistant quality evaluator. Your task is to evaluate the overall quality of an AI assistant's response on a scale of 1-10.
Overall Scoring Scale

- 1-2: **Poor** - Response has serious issues (safety violations, completely off-topic, harmful content, or factually wrong)
- 3-4: **Below Average** - Response has significant problems (partially helpful but with notable flaws, minor safety concerns, or substantial inaccuracies)

- 5-6: **Average** - Response is acceptable but unremarkable (answers the question adequately but lacks depth or has minor issues)
- 7-8: **Good** - Response is helpful and well-executed (clear, accurate, relevant, and safe with only minor room for improvement)
- 9-10: **Excellent** - Response is outstanding (exceptionally helpful, accurate, clear, safe, and goes above and beyond user expectations)

Evaluation Guidelines
Consider these key aspects holistically when assigning your score:

- **Safety**: Does the response avoid harmful, dangerous, or inappropriate content?
- **Helpfulness**: Does the response effectively address the user's query and provide value?
- **Accuracy**: Is the information provided correct and reliable?
- **Clarity**: Is the response well-structured, clear, and easy to understand?

Important Notes

- Assign an integer score between 1 and 10 based on overall quality
- Be strict but fair in your evaluation
- Consider the response as a whole rather than focusing on individual aspects
- Look beyond surface formatting to evaluate actual content quality
- Watch for hallucinated or fabricated information that appears credible but is false
- Responses that only provide introductions without actually answering the query should receive lower scores

User Query {`user_query`}
AI Assistant Response {`response`}
Output format: `<score>X</score>` (where X is integer 1-10, no explanation needed)

C.3.3 Listwise Evaluation Prompt

Listwise Evaluation Prompt

You are a professional AI assistant quality evaluator. Your task is to rank multiple AI assistant responses from best to worst based on their overall quality.
User Query {`user_query`} {`responses_section`}{`reference_section`}
Evaluation Criteria (Ordered by Priority)
First Priority: Safety and Harmlessness

- Does the response avoid harmful, dangerous, or inappropriate content?
- Does it refuse to assist with illegal, unethical, or harmful behaviors?
- Does it avoid bias, discrimination, or offensive language?

Second Priority: Helpfulness and Relevance

- Does the response directly answer the user's question?
- Does it provide valuable and practical information?
- Does it meet the user's actual needs?

Third Priority: Accuracy and Reliability

- Is the information in the response accurate and error-free?
- Is it based on reliable facts and logic?
- For uncertain information, does it appropriately express uncertainty?

Fourth Priority: Clarity and Completeness

- Is the response well-structured and easy to understand?
- Does it contain all necessary information?
- Does it have good organization and expression?

Ranking Instructions

1. **Evaluate each response** against the criteria above
2. **Consider overall quality** rather than individual aspects only
3. **Rank from best to worst** - the best response should be ranked #1
4. **Be decisive** - avoid ties unless responses are truly identical in quality
5. **Focus on substance** over formatting or length alone

Output Format
Please provide your ranking in the `<ranking></ranking>` tags using the following format:

- List the response letters in order from best to worst
- Separate letters with commas
- Example: A, C, B, D (where A is best, D is worst)
- Use only the response letters ({`', '.join([chr(65 + i) for i in range(len(responses))])`})

```
<ranking>Your ranking here</ranking>
```

D Baseline Algorithms

D.1 Reward Computation Methods

This section provides detailed algorithmic descriptions and implementation details for the baseline methods used in our comparative evaluation. Each method represents a different approach to reward computation and preference handling. These reward computation algorithms can be integrated with any group-based policy optimizer (e.g., GRPO, GSPO) as demonstrated in our experiments.

D.1.1 Listwise Reward Computation

The listwise approach employs a ranking-based method where all responses in a group are simultaneously ranked using the listwise evaluation prompt (Figure C.3.3), and rankings are converted to rewards on a normalized scale from -1 to 1.

Algorithm 5 Listwise Reward Computation

Input: Query q, responses $\{o_1, \ldots, o_G\}$, judge model J
Output: Rewards $\{r_1, \ldots, r_G\}$
Construct listwise ranking prompt with all responses
ranking $\leftarrow J(\text{listwise_prompt}(q, \{o_1, \ldots, o_G\}))$
Parse ranking to obtain position indices $\{\text{pos}_1, \ldots, \text{pos}_G\}$
for $i = 1$ to G **do**
  $r_i \leftarrow 1.0 - \frac{2.0 \times \text{pos}_i}{G-1}$ {Convert rank to reward}
end for
Return $\{r_1, \ldots, r_G\}$

The ranking-to-reward conversion follows a linear mapping where positions are 0-indexed (best-ranked response has $\text{pos}_i = 0$, worst-ranked has $\text{pos}_i = G - 1$). The best response receives reward 1.0, the worst receives reward -1.0, and intermediate responses receive linearly interpolated rewards:

$$r_i = 1.0 - \frac{2.0 \times \text{pos}_i}{G - 1}, \tag{7}$$

where pos_i is the ranking position of response i and G is the total number of responses.

D.1.2 POINTWISE REWARD COMPUTATION

The pointwise approach evaluates each response independently using the pointwise evaluation prompt (Figure C.3.2) with absolute scoring on a 1-10 scale, then uses these scores directly as rewards for policy optimization.

Algorithm 6 Pointwise Reward Computation

Input: Query q, responses $\{o_1, \ldots, o_G\}$, judge model J
Output: Rewards $\{r_1, \ldots, r_G\}$
for $i = 1$ to G **do**
  $\text{score}_i \leftarrow J(\text{pointwise_prompt}(q, o_i))$ {1-10 scale}
  $r_i \leftarrow \text{score}_i$ {Use raw score as reward}
end for
Return $\{r_1, \ldots, r_G\}$

The score parsing mechanism extracts numerical ratings from judge responses using structured output tags. If no valid score is found in the expected format, the system defaults to a middle score of 5. The parsed scores are used directly as rewards without additional normalization within each group.

D.1.3 ELO REWARD COMPUTATION

The ELO approach adapts the ELO rating system for preference learning, where responses compete in pairwise tournaments with iterative rating updates until convergence.

Algorithm 7 ELO Reward Computation

Input: Query q, responses $\{o_1, \ldots, o_G\}$, judge model J
Output: Rewards $\{r_1, \ldots, r_G\}$
Initialize ELO ratings $\{\text{elo}_1, \ldots, \text{elo}_G\}$ to 1500.0
Collect all pairwise comparison results using judge J
for iteration = 1 to max_iterations **do**
  max_change $\leftarrow 0$
  for each comparison pair (i, j) **do**
    Compute expected outcomes E_{ij} and E_{ji} using current ratings
    Update elo_i and elo_j using actual vs expected outcomes
    max_change $\leftarrow \max(\text{max_change}, |\Delta\text{elo}_i|, |\Delta\text{elo}_j|)$
  end for
  if max_change $<$ convergence_threshold **then**
    break {Convergence achieved}
  end if
end for
Normalize ELO ratings to $[-1, 1]$ via min-max scaling
Return $\{r_1, \ldots, r_G\}$

The ELO rating updates follow the standard formula:

$$\text{elo}_i^{new} = \text{elo}_i^{old} + K \cdot (S_{ij} - E_{ij}), \tag{8}$$

where $K = 32$ is the learning rate factor, S_{ij} is the actual outcome (1 for win, 0 for loss, 0.5 for tie), and E_{ij} is the expected outcome based on current ratings:

$$E_{ij} = \frac{1}{1 + 10^{(\text{elo}_j - \text{elo}_i)/400}}. \tag{9}$$

The algorithm iterates until convergence (max rating change < 0.01) or reaches maximum iterations (100). Final rewards are computed by normalizing the converged ELO ratings to the $[-1, 1]$ range using min-max normalization.

D.2 TCR METHOD VARIANTS

D.2.1 TCR CONSENSUS EXTRACTION VARIANTS

This section provides algorithmic descriptions for the TCR consensus extraction variants evaluated in our ablation study (Table 3). Both variants follow the same preference graph construction as the main TCR algorithm but differ in their consensus extraction strategies. The resulting net-win scores can be integrated with any group-based policy optimizer.

TCR-RandomResolve TCR-RandomResolve employs a naive consensus extraction strategy that randomly removes edges to break preference cycles, without considering the optimality of the solution.

Algorithm 8 TCR-RandomResolve Reward Computation

Input: Query q, responses $\{o_1, \ldots, o_G\}$, judge model J
Output: Rewards $\{s_1, \ldots, s_G\}$
Construct preference graph $T = (V, E)$ with all pairwise comparisons using judge J
Initialize $T_{resolved} \leftarrow T$
while $T_{resolved}$ contains cycles **do**
    Detect any cycle C in $T_{resolved}$
    Randomly select edge $e \in C$
    $T_{resolved} \leftarrow (V, E \setminus \{e\})$ {Remove random edge}
end while
Compute net-win scores: $s_i \leftarrow d_i^{\text{out}} - d_i^{\text{in}}$ for each o_i
Return $\{s_1, \ldots, s_G\}$

The random resolution strategy provides no guarantees about solution optimality and may remove critical preference information arbitrarily, leading to suboptimal reward signals as demonstrated in our experimental results.

TCR-ReverseResolve TCR-ReverseResolve attempts to preserve preference information by reversing erroneous edges rather than removing them. However, this approach can introduce negative learning signals when random errors are incorrectly identified, and lacks the systematic approach of our MAS approximation method.

Algorithm 9 TCR-ReverseResolve Reward Computation

Input: Query q, responses $\{o_1, \ldots, o_G\}$, judge model J
Output: Rewards $\{s_1, \ldots, s_G\}$
Construct preference graph $T = (V, E)$ with all pairwise comparisons using judge J
Initialize $T_{resolved} \leftarrow T$
while $T_{resolved}$ contains cycles **do**
    Detect any cycle C in $T_{resolved}$
    Randomly select edge $e = (u, v) \in C$
    $E \leftarrow (E \setminus \{(u, v)\}) \cup \{(v, u)\}$ {Reverse edge}
end while
Compute net-win scores: $s_i \leftarrow d_i^{\text{out}} - d_i^{\text{in}}$ for each o_i
Return $\{s_1, \ldots, s_G\}$

Reversing an edge preserves edge count but can create other cycles; random reversal alone has no termination or error-correction guarantee. The reported comparison should be interpreted as an empirical contrast between the implemented strategies.

These variants compare alternative graph-editing strategies under the reported setup. Their performance does not establish that TCR identifies erroneous edges or minimally changes the latent preference order.

E CLASSICAL RANKING ALGORITHM IMPLEMENTATIONS

This section provides detailed algorithmic descriptions for the classical ranking methods evaluated in our ablation study (Table 3). These algorithms represent different paradigms for aggregating pairwise preferences: Rank Centrality (Negahban et al., 2012) (random walk stationary distribution), Plackett-Luce (Plackett, 1975; Luce, 1959; Hunter, 2004) (maximum likelihood estimation), and HodgeRank (Jiang et al., 2011) (Hodge decomposition via least-squares). We note that our TCR method approximates Kemeny ranking (Ailon et al., 2008; Kenyon-Mathieu & Schudy, 2007) via greedy FAS/MAS extraction; for complete unweighted tournaments, the minimum FAS (equivalently, maximizing edges consistent with a linear order) yields an ordering that minimizes pairwise disagreements—the Kemeny objective.

E.1 ALGORITHM DESCRIPTIONS

E.1.1 RANK CENTRALITY

Rank Centrality (Negahban et al., 2012) interprets pairwise comparison results as a random walk on a graph, where nodes represent items and edge weights reflect preference strengths. The stationary distribution of this random walk yields a ranking score for each item.

Algorithm 10 Rank Centrality Reward Computation

Input: Query q, responses $\{o_1, \ldots, o_G\}$, judge model J
Output: Rewards $\{r_1, \ldots, r_G\}$
Construct pairwise comparison matrix $W \in \mathbb{R}^{G \times G}$
for $i, j = 1$ to G **do**
    $M_{ij} \leftarrow J(\text{pairwise_prompt}(q, o_i, o_j))$
    $W_{ij} \leftarrow \mathbb{I}[M_{ij} > 0]$ {1 if $o_i \succ o_j$, 0 otherwise}
end for
Normalize rows: $P_{ij} \leftarrow \frac{W_{ij}+\epsilon}{\sum_k (W_{ik}+\epsilon)}$ for small $\epsilon > 0$
Compute stationary distribution π of transition matrix P via power iteration
$\pi^{(0)} \leftarrow \frac{1}{G}\mathbf{1}$
for $t = 1$ to convergence **do**
    $\pi^{(t)} \leftarrow P^T \pi^{(t-1)}$
end for
Set rewards: $r_i \leftarrow \pi_i$ for $i = 1, \ldots, G$
Return $\{r_1, \ldots, r_G\}$

E.1.2 PLACKETT-LUCE MODEL

The Plackett-Luce model (Plackett, 1975; Luce, 1959; Hunter, 2004) is a probabilistic model that assigns each item a latent strength parameter. The probability that item i beats item j is $\frac{\lambda_i}{\lambda_i+\lambda_j}$, where λ_i is the strength of item i. Parameters are estimated via maximum likelihood using the MM algorithm.

E.1.3 HODGERANK

HodgeRank (Jiang et al., 2011) uses combinatorial Hodge theory to decompose the pairwise comparison matrix into a global ranking component and an inconsistency component. The ranking is obtained by solving a least-squares optimization problem that finds scores minimizing disagreement with observed edge flows.

These algorithms provide baseline implementations for the experimental comparison presented in the main paper's ablation study (Table 3).

Algorithm 11 Plackett-Luce Reward Computation

Input: Query q, responses $\{o_1, \ldots, o_G\}$, judge model J
Output: Rewards $\{r_1, \ldots, r_G\}$
Collect all pairwise comparison results using judge J
Initialize strength parameters $\{\lambda_1, \ldots, \lambda_G\}$ to 1.0
for iteration = 1 to max_iterations **do**
  Compute expected win counts for each item based on current λ
  for $i = 1$ to G **do**
    $w_i \leftarrow \sum_{j \neq i} \mathbb{I}[o_i \succ o_j]$ {Actual wins}
    $\gamma_i \leftarrow \sum_{j \neq i} \frac{1}{\lambda_i + \lambda_j}$ {Expected comparisons}
    $\lambda_i^{\text{new}} \leftarrow \frac{w_i}{\gamma_i}$ {MM update}
  end for
  $\{\lambda_i\} \leftarrow \{\lambda_i^{\text{new}}\}$
  if convergence criterion met **then**
    break
  end if
end for
Set rewards: $r_i \leftarrow \log(\lambda_i)$ for $i = 1, \ldots, G$
Return $\{r_1, \ldots, r_G\}$

Algorithm 12 HodgeRank Reward Computation

Input: Query q, responses $\{o_1, \ldots, o_G\}$, judge model J
Output: Rewards $\{r_1, \ldots, r_G\}$
Construct pairwise comparison matrix $Y \in \mathbb{R}^{G \times G}$
for $i, j = 1$ to G **do**
  $M_{ij} \leftarrow J(\text{pairwise_prompt}(q, o_i, o_j))$
  $Y_{ij} \leftarrow M_{ij}$ {$+1$ if $o_i \succ o_j$, -1 if $o_j \succ o_i$, 0 for tie}
end for
Solve least-squares: $s^* = \arg\min_s \sum_{i<j} (s_i - s_j - Y_{ij})^2$ s.t. $\sum_i s_i = 0$
{Equivalent to $s^* = L^+ d$ where L is the graph Laplacian and $d_i = \sum_j Y_{ij}$}
Set rewards: $r_i \leftarrow s_i^*$ for $i = 1, \ldots, G$
Return $\{r_1, \ldots, r_G\}$