LEARNING WHEN TO SPAWN AGENTS VIA QUALITY-GUIDED COST OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

A language-model recruitment policy must decide when additional agents justify their execution cost. With Group Relative Policy Optimization (GRPO), adding a cost penalty to the reward can make cost differences determine task-level advantages whenever all sampled trajectories have the same quality. We show why reducing the penalty coefficient may have little effect: normalization approximately cancels its scale when weighted cost dispersion is large relative to the numerical stability term. Quality-Guided Cost Optimization (QGC) uses the group's quality state to choose which cost comparisons to retain. It assigns zero task scores when quality is zero, compares team size alone when quality is identical and positive, and combines separately normalized quality and full estimated cost when quality varies. Positive quality includes partial success. The positive-quality branch therefore favors smaller teams at the same observed progress without penalizing output length. Across three Qwen3-8B training seeds on controlled compositional programming tasks, QGC raises mean success from 44.56% to 46.03% (+1.46 percentage points before rounding) and reduces mean estimated cost by 28.78% relative to Quality-Only. Against Guided Direct Cost, it gains 1.95 percentage points with 1.51% lower cost. The gains vary across seeds and models. Branch comparisons show higher success at slightly higher cost relative to full-cost tie feedback, and lower cost with a small success loss relative to tie suppression. These results support choosing cost comparisons according to observed task progress, while leaving matched-resource branch benefits to be established.

1 INTRODUCTION

Dividing work among language-model agents can help solve complex tasks, but each additional agent adds execution and coordination overhead. A single agent may suffice for a simple task; a task with several subtasks may benefit from a team. A recruitment policy must make this choice for each task and weigh any quality gain against the resources it requires. Prior research jointly optimizes quality and cost through reinforcement learning. For example, MAO-ARAG (Chen et al., 2025) trains an orchestration policy using quality rewards and cost penalties. Group Relative Policy Optimization (GRPO) centers and normalizes rewards sampled for the same task, providing relative advantages as within-group comparison signals (Shao et al., 2024). Related work includes DAPO (Yu et al., 2025), which filters out sample groups in which all responses are correct or all are incorrect and draws additional samples. DRPO (Li et al., 2025) addresses interference between correctness and length rewards within its decoupled optimization framework. MDP-GRPO (Salmani-Zarchi et al., 2026) examines normalization issues under low-dispersion rewards. In agent recruitment, costs arise from both team size and the outputs generated during execution. When repeated executions yield identical quality scores, we must specify which cost comparisons to retain and how they influence recruitment policy learning.

Consider a group of trajectories for the same task, scored by quality q minus estimated cost λc. If every trajectory has the same quality, centering removes the quality term. When weighted cost dispersion $\lambda\sigma_c$ is large relative to the numerical stability term, normalization also approximately cancels the scale of λ. Cheaper failed trajectories can then receive positive task-level advantages even though no trajectory made measured progress. Reducing the cost coefficient need not weaken this signal in proportion.

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.

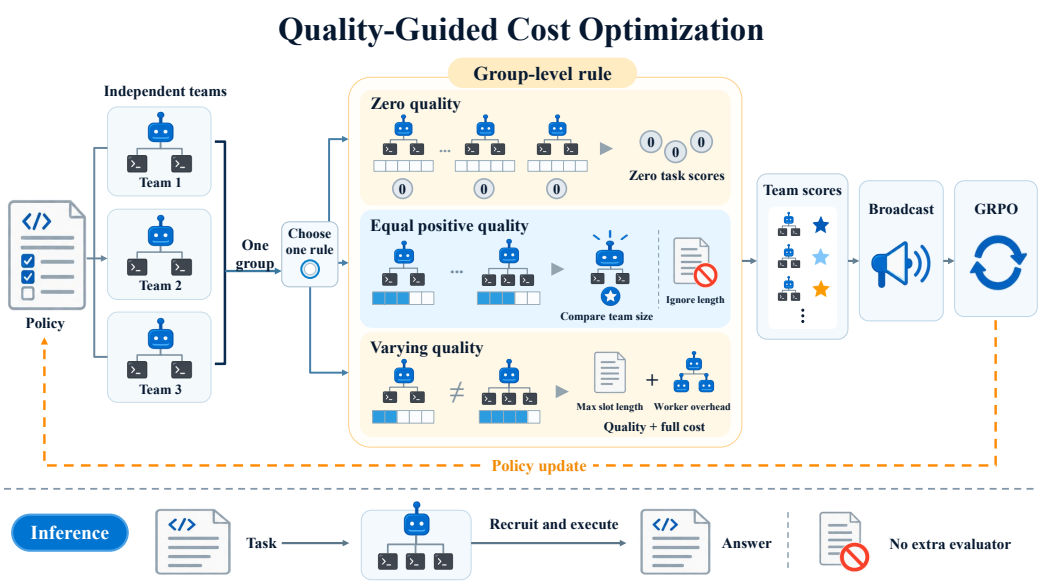


Figure 1: QGC overview. Independent teams solving one task share a rule: zero task scores for zero quality; team-size comparisons for equal positive quality; otherwise, separately normalized quality and full cost. Scores feed GRPO; inference needs no additional quality evaluator.

The appropriate comparison depends on what the group achieved. An all-zero-quality group offers no successful progress to preserve. A group with identical positive quality, including partial success, lets us compare how many agents were needed to reach that level. Recruiting fewer agents and shortening their outputs both reduce cost, but they impose different preferences on execution. QGC uses this distinction to select training scores: zero scores for zero-quality groups, team-size comparisons for positive-quality ties, and separately normalized quality and full cost when quality varies. The positive-tie branch retains the preference for smaller teams while excluding output-length penalties. Figure 1 shows how these scores enter the existing GRPO update. The method reuses trajectory quality, team size and cost statistics; inference needs no additional quality evaluator.

We evaluate QGC on controlled compositional programming tasks, measuring success and cost alongside recruitment and output behavior. In the Qwen3-8B comparison across training seeds 0, 1, and 2, QGC raises mean success from 44.56% to 46.03% relative to Quality-Only, a gain of 1.46 percentage points before rounding, while reducing mean estimated cost by 28.78%. Against Guided Direct Cost, the gain is 1.95 percentage points with 1.51% lower cost. One seed reverses the success gain, so the average does not describe every run. Comparisons with other models and branch rules also reveal quality–cost trade-offs. We therefore report both outcomes for each configuration and distinguish the current aggregate comparisons from the historical recruitment diagnostics. The latter show how team size varies with task structure; they do not explain the success gains in the current main table.

Our main contributions are as follows:

1. We propose Quality-Guided Cost Optimization for agent recruitment. The method selects cost comparisons based on within-group quality feedback. For groups with identical positive quality scores, it retains team-size comparisons while excluding output-length penalties from this branch.

2. We derive the condition under which normalization approximately cancels the cost coefficient in a quality-tied group. Component comparisons and recruitment diagnostics then examine how the choice of cost signal relates to success, team size and output behavior, including configurations with quality–cost trade-offs.

2 Related Work

Multi-agent systems reduce execution overhead by adapting workflows, participating agents, and communication structures. MaAS (Zhang et al., 2025) samples query-dependent workflows from an agentic supernet, whereas AOrchestra (Ruan et al., 2026) constructs sub-agents according to task requirements and uses in-context learning to optimize cost-aware model routing. AgentPrune (Zhang et al., 2024) performs one-shot pruning of spatiotemporal message-passing graphs, while AgentDropout (Wang et al., 2025) removes redundant agents and connections by optimizing communication graphs. Learning-based orchestration incorporates quality and cost into training. MAO-ARAG (Chen et al., 2025) trains a planner using quality rewards and cost penalties. xRouter (Qian et al., 2025) learns whether to answer directly or invoke external models. ToolOrchestra (Su et al., 2025) trains an orchestration policy for models and tools using outcome, efficiency, and preference rewards. Both methods condition cost or efficiency rewards on the success of individual trajectories, providing direct precedents for incorporating quality feedback into cost optimization. The recent study *Rethinking the Evaluation of Efficiency Methods for Multi-Agent Systems (Zhang et al., 2026)* evaluates efficiency methods using shared models, agent configurations, and a common runtime framework. It reports that some gains depend on the initial structure, prompts, and whether the task requires collaboration. Our focus is the cost comparison made within a group of trajectories for the same task. Their quality scores determine whether cost contributes to the training signal and whether that signal compares team size or also penalizes output length. Recruitment and task completion help interpret the resulting cost changes.

Another line of research examines within-group advantages and multiple-reward combination. DeepSeekMath (Shao et al., 2024) introduces GRPO, which computes relative advantages from rewards for outputs sampled for the same problem. M-GRPO (Hong et al., 2025) supports joint training of main agents and sub-agents through hierarchical advantages and trajectory alignment. Dr. GRPO (Liu et al., 2025) analyzes optimization biases introduced by response-length normalization and normalization by the within-problem reward standard deviation. MDP-GRPO (Salmani-Zarchi et al., 2026) addresses low-dispersion rewards using multi-temperature sampling, dual-anchor advantages, and bounded advantage transformations. DAPO (Yu et al., 2025) filters out sample groups in which all responses are correct or all are incorrect and draws additional samples. DRPO (Li et al., 2025) separates positive and negative samples within the DisCO framework and introduces length-based weights among correct samples. MO-GRPO (Ichihara et al., 2025) and GDPO (Liu et al., 2026) are closer to our score-combination design. MO-GRPO normalizes each reward separately before combining them, mitigating dominance by objectives with higher reward variance. GDPO similarly normalizes each reward separately within groups, combines the resulting advantages, and then normalizes them across the batch. This design preserves distinctions between different reward combinations. Separate reward normalization thus has clear precedents. QGC adds a decision about which cost comparisons remain meaningful after observing the group's quality. It distinguishes an all-zero-quality group from one whose trajectories reach the same positive quality, and retains team-size comparisons only in the latter. This applies to partial success as well as complete solutions. The distinction concerns the content of the cost signal; separate normalization handles its combination with quality when quality varies.

3 Method

Quality-Guided Cost Optimization selects the cost components in training scores based on quality feedback from multiple team trajectories for the same task. Figure 1 summarizes the training and inference flow.

3.1 Tasks and Metrics

Given a task x, the policy π_θ generates a recruitment and execution trajectory τ. The controller determines the number of workers, $m(\tau) \in \{0, \ldots, M\}$, giving a team size of $T(\tau) = 1 + m(\tau)$; $m = 0$ denotes independent execution. We sample G independent team trajectories for the same task to form a comparison group.

A compositional programming task contains $K_x \geq 1$ subtasks. Let $n_{\text{pass}}(\tau)$ denote the number that pass evaluation. Training quality and full-task success are defined as

$$q(\tau) = \frac{n_{\text{pass}}(\tau)}{K_x}, \qquad y(\tau) = \mathbf{1}[n_{\text{pass}}(\tau) = K_x]\,. \tag{1}$$

Positive quality therefore includes partial completion, whereas evaluation success requires all subtasks to pass. Estimated cost is

$$c(\tau) = P(\tau) + \alpha m(\tau), \qquad P(\tau) = \max_{j \in \mathcal{S}} L_j(\tau), \tag{2}$$

where $\mathcal{S}$ is the set of logical execution slots, L_j is the cumulative text-length estimate for slot j, and $\alpha = 50$. Each action's character count is divided by four, rounded down, and assigned a minimum value of one before accumulation within its slot. This proxy adds maximum slot length to recruitment overhead; it is not the actual total token count, latency, or billed cost.

3.2 Within-Group Normalization and Cost

GRPO constructs advantages from relative rewards within a task group (Shao et al., 2024). Consider the direct cost score $s_i = q_i - \lambda c_i$, with $\lambda > 0$, and its normalized advantage:

$$A_i = \frac{s_i - \bar{s}}{\sigma_s + \epsilon_{\text{out}}}, \tag{3}$$

where $\bar{s}$ and σ_s are the within-group mean and standard deviation, and $\epsilon_{\text{out}} > 0$ is a stability term. This score is used to analyze cost penalties; the experimental Quality-Only baseline uses only q_i.

Observation 1: With identical quality scores, normalization may approximately cancel the scaling effect of the cost coefficient. If all $q_i = q_0$, $\sigma_c > 0$, and standard deviations are defined consistently, then

$$A_i = -\frac{\lambda(c_i - \bar{c})}{\lambda\sigma_c + \epsilon_{\text{out}}} = -\frac{\lambda\sigma_c}{\lambda\sigma_c + \epsilon_{\text{out}}} \frac{c_i - \bar{c}}{\sigma_c}. \tag{4}$$

Centering removes the common quality term, and the score standard deviation is $\sigma_s = \lambda\sigma_c$. When $\lambda\sigma_c \gg \epsilon_{\text{out}}$, relative costs largely determine the advantages; when it is comparable to or below the stability term, reducing λ can still weaken them. In particular, failed trajectories with below-average costs receive positive task-level advantages in all-zero-quality groups. This motivates cost-component selection without implying that training must deteriorate. Appendix A.3 gives the finite-epsilon analysis; Figure 2(a) illustrates the resulting signal in a fixed zero-quality group.

3.3 Quality-Guided Cost Optimization

We define within-group normalization using the population standard deviation and $\epsilon = 10^{-6}$:

$$\sigma_v = \sqrt{\frac{1}{G}\sum_{i=1}^{G}(v_i - \bar{v})^2}, \qquad z_\epsilon(v_i) = \begin{cases} \dfrac{v_i - \bar{v}}{\sigma_v + \epsilon}, & \sigma_v > \epsilon, \\ 0, & \sigma_v \le \epsilon. \end{cases} \tag{5}$$

Given quality, estimated cost, and team size q_i, c_i, T_i, QGC constructs

$$s_i^\star = \begin{cases} 0, & \sigma_q \le \epsilon,\ \bar{q} \le \epsilon, \\ -\lambda z_\epsilon(T_i), & \sigma_q \le \epsilon,\ \bar{q} > \epsilon, \\ z_\epsilon(q_i) - \lambda z_\epsilon(c_i), & \sigma_q > \epsilon. \end{cases} \tag{6}$$

Zero-quality groups provide no cost-only task-level advantages. Groups with identical positive quality compare team size alone, yielding zero scores if team sizes are also identical. Groups with varying quality combine separately normalized quality and full estimated cost. The positive-quality

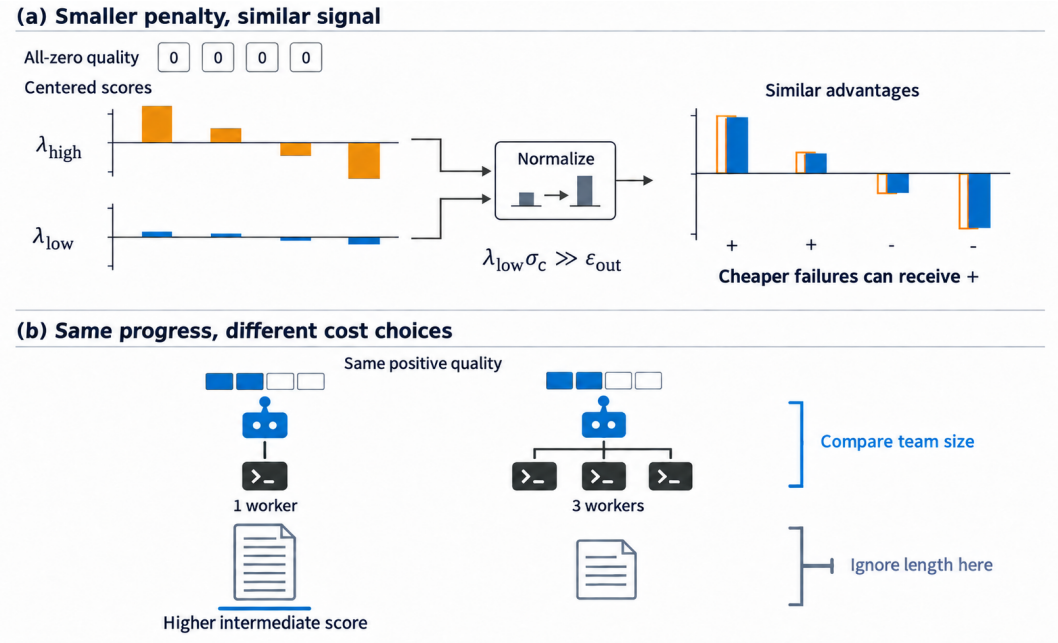


Figure 2: QGC motivation (schematic). (a) With equal zero quality, a smaller direct-cost coefficient can yield similar normalized advantages when weighted cost dispersion dominates the stabilizer. QGC instead sets zero-quality task scores to zero. (b) Both teams complete two of four subtasks. QGC gives the smaller team a higher intermediate score despite its longer slot-text estimate: this branch excludes length.

branch keeps a preference for smaller teams at equal observed progress. Suppressing all quality-tied groups would discard this comparison. Figure 2(b) illustrates the team-size preference at equal positive quality.

The quality state determines how a cost difference enters training. At equal positive quality, a smaller team reaches the same measured progress with fewer participants, while a shorter output may reflect a different execution strategy. QGC keeps recruitment feedback in this state and excludes the output-length preference. This also applies when every team completes only part of the task, as in Figure 2(b). When quality varies, the rule compares progress together with full cost. The switch uses measured group statistics and requires no additional learned selector.

3.4 Training and Inference

Scores are constructed over independent teams; slot records from the same team do not increase the sample count. After assigning team scores to the corresponding sample rows, the existing GRPO trainer computes advantages and updates the policy:

$$\widehat{A} = \mathcal{N}_{\mathrm{GRPO}}(\mathrm{Broadcast}(s^\star)) \,, \tag{7}$$

Equation (6) defines intermediate scores. Downstream normalization may also cancel the overall scale of λ in the positive-quality-tie branch, so λ primarily controls the relative weights in groups with varying quality. Zero task scores do not preclude updates from other losses such as KL regularization. Removing the length term does not isolate recruitment and execution gradients. Inference directly uses the learned recruitment policy without an additional quality evaluator. Pseudocode is provided in Appendix A.4.

4 EXPERIMENTS

We examine how the choice of training score affects success, cost and recruitment, and whether the observed trade-offs persist across models.

4.1 EXPERIMENTAL SETUP

Tasks and configuration. We use single-function Anchor tasks, Trap tasks with collaboration cues, and compositions of two to four subtasks (k2–k4). A controller executes alone or recruits up to four workers. Qwen3-8B is the primary model; supplementary models are Qwen3-4B-Instruct-2507, Llama-3.1-8B-Instruct, and Llama-3.2-3B-Instruct. Table 1 reports training seeds 0, 1, and 2, with $\lambda = 0.03$ for QGC and $\lambda = 0.1$ for GDC and NC; the branch comparison fixes $\lambda = 0.1$.

Appendix B describes the historical benchmark of 194 tasks with three decoding repetitions, together with its model sources and runtime settings. These records are separate from the aggregate comparisons reported here. Appendix C defines the aggregation and rounding rules and explains the evaluation scope.

Metrics and aggregation. Quality-Only (QO) rewards quality alone; QGC follows Equation (6). We report full-task success and mean proxy cost $c = P + 50m$ (Equation (2)), including failures. Team size includes the controller; tokenizer-counted total output is separate. Table 1 reports equal-weight means and sample SD across three training seeds. Tables 2–3 and Figure 3 present aggregate outcomes. These comparisons are descriptive; Appendix C explains their evaluation and uncertainty limits.

4.2 MAIN RESULTS

Table 1: Qwen3-8B results. Values are equal-weight mean ± sample SD across training seeds 0, 1, and 2; SD is descriptive training-seed variation, not a confidence interval. Cost is $c = P + 50m$ in raw proxy units. Per-seed values appear in Appendix Table C.1.

Method	λ	**Success (%)** ↑	**Mean cost** ↓
QO	—	44.56 ± 0.75	1011.1 ± 24.7
GDC	0.1	44.08 ± 0.84	731.2 ± 8.4
NC	0.1	41.02 ± 0.72	691.7 ± 16.4
QGC (Ours)	0.03	46.03 ± 1.29	720.1 ± 19.1

QGC increases mean success from 44.56% to 46.03% relative to QO: +1.46 percentage points before rounding, with 28.78% lower mean cost. Relative to GDC, QGC gains 1.95 pp at 1.51% lower cost; relative to NC, it gains 5.00 pp at 4.11% higher cost. The gain varies by seed: on seed 1, QGC success is lower than both QO and GDC. Appendix C.3 gives the per-seed summaries and explains the uncertainty measures.

Figure 3(b) displays these training-seed differences; its configurations match the $0.75\times$ operating-point aggregates in panel (a) after rounding.

The seed summaries separate cost reduction from quality variation. Every QGC run has lower mean cost than the corresponding QO run, whereas success improves on seeds 0 and 2 and decreases on seed 1. The comparison with GDC places the mean success gain alongside a much smaller cost difference. NC reaches the lowest mean cost in Table 1, with lower success. Thus the four methods occupy different quality–cost positions; the mean improvement is informative together with the individual runs.

4.3 COMPONENTS AND DIRECT ALTERNATIVES

The underlying 2×2 design varies quality guidance and score combination. Direct Cost (DC) uses $q_i - \lambda c_i/50$ in every group; Normalized Cost (NC) uses $z_\epsilon(q_i) - \lambda z_\epsilon(c_i)$. Guided Direct Cost (GDC) shares both quality-tied branches with QGC but uses the DC score when quality varies (Ap-

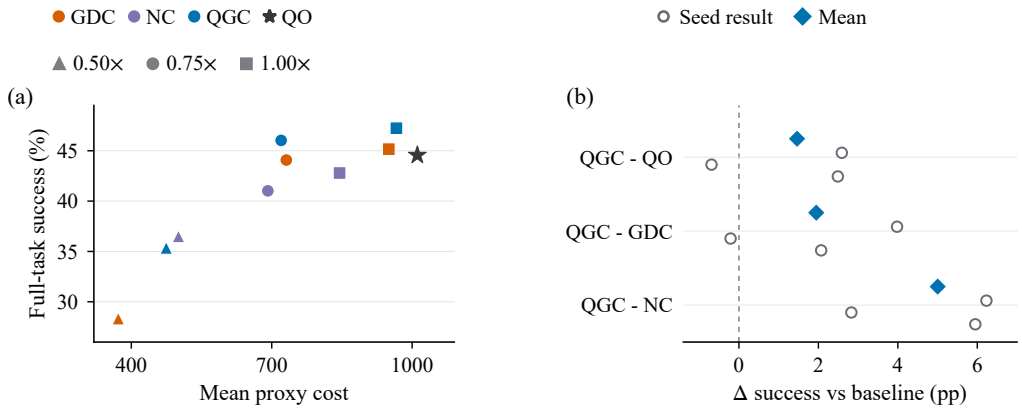


Figure 3: Qwen3-8B outcomes. (a) Success–cost points; shapes denote dev budgets relative to QO, not matched realized costs. The QO star is the Table 1 mean. (b) QGC-minus-baseline success differences for the three Table 1 training seeds (hollow dots) and their means (diamonds); vertical offsets separate marks.

pendix A.2). Table 2 compares QO, GDC, NC and QGC. A corresponding DC result is unavailable for this set of experiments.

Table 2: Supplementary-model results. Panel (a) gives success (%) / mean cost in raw proxy units. Panel (b) retains the separately reported QGC–QO contrasts. Cells and contrasts are rounded separately; see Appendix C.2.

(a) Absolute outcomes.

Model	**QO**	**GDC**	**NC**	**QGC (Ours)**
Qwen3-4B	36.93 / 328.6	36.83 / 277.6	32.41 / 247.9	38.21 / 258.5
Llama-3.1-8B	29.40 / 3188.3	27.96 / 1094.8	23.49 / 877.2	28.36 / 985.5
Llama-3.2-3B	22.37 / 904.8	22.94 / 583.7	18.54 / 489.6	23.92 / 536.0

(b) QGC relative to QO.

Model	**QO success**	**QGC success**	**ΔSuccess**	**Cost change**
Qwen3-4B	36.93%	38.21%	+1.28 pp	−21.35%
Llama-3.1-8B	29.40%	28.36%	−1.04 pp	−69.09%
Llama-3.2-3B	22.37%	23.92%	+1.55 pp	−40.75%

Across all three models in Table 2(a), QGC has higher success and lower cost than GDC, and higher success at higher cost than NC. The comparison covers the reported configurations; sensitivity to tuning and variation across training seeds remain unmeasured for these models.

Figure 3 shows budget-dependent trade-offs. At $0.75\times$ QO, QGC has higher success and lower cost than GDC. At $0.50\times$, NC has higher success than QGC at higher cost; at $1.00\times$, QGC has higher success and higher cost than GDC. Each plotted point describes a complete configuration. The horizontal coordinate gives its realized mean cost, while the marker identifies its development-budget constraint. Comparing both coordinates is therefore necessary even for methods with the same marker. Panel (b) shows seed variation for the central operating point. The selection and evaluation scope is stated in Appendix C.2.

Appendix A.5 establishes conditional distinctions from DC and GDC with fixed coefficients; these do not guarantee performance gains. The QGC–NC comparison changes both tied-quality branches, preventing separate attribution to the positive-quality rule from that comparison alone. The dedicated controls in Section 4.5 examine these rule choices separately.

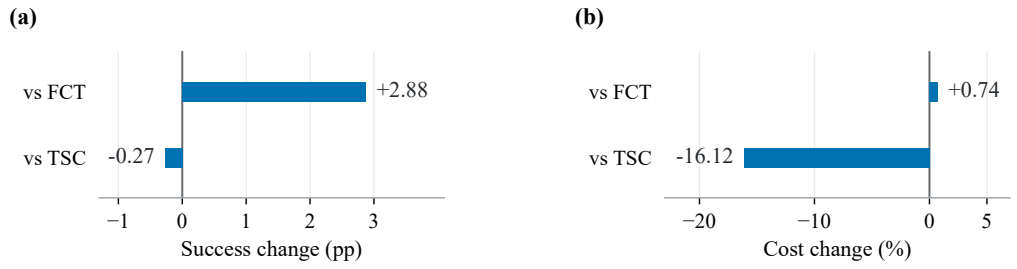


Figure 4: Qwen3-8B branch contrasts at $\lambda = 0.1$. Bars show QGC relative to each baseline: (a) success difference in percentage points and (b) percentage change in mean proxy cost. Values retain the separately reported rounding (Appendix C.2). Absolute outcomes, team size and output tokens remain in Table 3.

4.4 Recruitment and Output

The branch summaries in Table 3 report fewer team members for QGC than FCT or TSC. QGC retains more total output than FCT and similar total output to TSC. Fewer team members can therefore coexist with longer total output.

Subtracting the controller from the team means in Table 3 gives 1.327 workers for QGC, 1.707 for FCT and 1.930 for TSC (Equation (A.1)). These worker counts are calculated from the same summaries, rather than measured separately.

The resource columns answer different questions. Recruitment overhead grows with the number of workers; the length term uses the longest accumulated slot, whereas total output tokens sum generated text. Fewer workers can therefore coexist with longer output or a similar proxy cost. The QGC–FCT comparison shows this pattern: QGC uses a smaller team and produces more output at slightly higher cost. Against TSC, QGC uses fewer workers with similar total output and lower cost. These distinctions connect the branch rules to the resources they score.

Historical Qwen3-8B recruitment decreases mainly on single-function tasks versus QO; QGC retains more workers on complex tasks versus GDC (Appendix D.1, Figure 5). This neither establishes optimal teams nor explains current success gains.

4.5 Quality States and Branch Diagnostics

Table 3: Qwen3-8B branch summaries at $\lambda = 0.1$, distinct from the $\lambda = 0.03$ main QGC configuration. FCT uses full cost in positive-quality ties; TSC suppresses all quality ties. Team size includes the controller. Total output tokens are reported separately from proxy cost.

Method	Success (%)	Mean cost	Mean team size	Total tokens
QGC	45.48	681.2	2.327	1861.1
FCT	42.60	676.1	2.707	1605.6
TSC	45.76	812.1	2.930	1865.1

Full-Cost Tie (FCT) compares full cost in positive-quality ties; Tie-Suppressed Cost (TSC) zeros all quality-tied groups. At $\lambda = 0.1$, the reported QGC–FCT contrast is +2.88 pp success with +0.74% cost. The reported QGC–TSC contrast is -0.27 pp with 16.12% lower cost. These comparisons suggest different effects of output-length penalties and team-size comparisons, but do not establish a matched-cost gain or quality noninferiority. Separately rounded contrasts can differ slightly from arithmetic on displayed cells (Appendix C.2). TSC does not implement DAPO's dynamic additional sampling.

Figure 4 displays the reported success and cost contrasts; absolute outcomes, team size and output tokens remain in Table 3.

Positive-quality ties account for 47.78%, 45.15%, and 48.10% of groups for QGC, FCT, and TSC; zero-quality groups account for 20.95%, 25.02%, and 20.23% (Appendix C.3). These frequencies describe how often each rule applies; their statistical interpretation is specified in Appendix C.3.

A rule applying to roughly half of the groups can affect many updates, but its frequency alone does not determine their magnitude or direction. Within a positive-quality tie, equal team sizes also give zero recruitment scores. Counting positive ties and counting nonzero team-size feedback therefore measure different aspects of training. A common quality state can still produce different recruitment feedback across groups.

Among 320 groups from two historical Qwen3-8B QO runs, 82.2% have identical quality, including 55.6% of all groups with identical positive quality (Appendix D.1). These frequent states motivate the tied-group rules; they are neither QGC training frequencies nor contributions to success.

An exploratory Qwen3-4B comparison at $\lambda = 0.3$, seed 0, also found higher logged success and higher slot-weighted cost with team-only feedback. This single-run result cannot establish a matched-cost branch benefit (Appendix D.2).

In a separate historical Qwen3-4B QGC–TSC comparison ($\lambda = 0.1$, steps 221–250), success differences on the more comparable seeds 1 and 2 are -0.99 and -0.63 pp, with cost-diagnostic reductions of 27.72% and 4.54%. Seed 0 differs in configuration and shows +2.19 pp and -8.86% cost. Success does not consistently improve, indicating a quality–resource trade-off in team-size comparisons. These historical controls remain separate from the current Qwen3-8B comparison; Appendix D.2 records their configurations.

4.6 Cross-Model Results and Scope

Table 2(b) reports joint gains over QO on Qwen3-4B (+1.28 pp success, 21.35% lower cost) and Llama-3.2-3B (+1.55 pp, 40.75% lower cost). Llama-3.1-8B trades 1.04 pp of success for 69.09% lower cost. These aggregates support model-dependent efficiency gains and trade-offs, without establishing universal improvement, statistical significance, or actual billing savings.

Absolute proxy costs differ substantially across models, so the relative changes are computed against each model's own QO baseline. Llama-3.1-8B has both the largest proportional cost reduction and a success decrease; its outcome is a trade-off. The Qwen3-4B and Llama-3.2-3B points improve both measures. Reporting these cases separately preserves the distinction between saving resources and improving task completion, which an average across models would obscure.

5 Conclusion

QGC uses group quality to decide which cost comparisons enter GRPO recruitment training. The analysis shows why lowering a scalar cost coefficient can leave normalized cost advantages nearly unchanged when quality is tied and weighted cost dispersion dominates the stabilizer. QGC removes the cost-only signal at zero quality, retains team-size comparisons at equal positive quality, and combines separately normalized quality and full cost when quality varies. Partial success counts as positive quality. These rules change the training scores while reusing the existing policy update and inference procedure.

Across three Qwen3-8B training seeds on controlled compositional programming tasks, QGC raises mean success from 44.56% to 46.03% (+1.46 pp before rounding) with 28.78% lower mean proxy cost than QO; versus GDC, it gains 1.95 pp at 1.51% lower cost. Branch summaries show higher success than FCT at slightly higher cost and lower cost than TSC with a small success decrease. Supplementary results show joint gains on Qwen3-4B and Llama-3.2-3B, and a trade-off on Llama-3.1-8B.

Historical recruitment diagnostics show the controller working alone on single-function tasks and teams of three, four and five on two-, three- and four-subtask tasks. This is evidence of task-dependent recruitment, without establishing optimal team sizes or explaining the main-table success gains. The reported cost is an execution proxy, so savings should not be read as measured billing reductions. Testing the branch rules at matched resources and on broader task distributions remains necessary to determine where this approach is useful.

AI USE STATEMENT

Generative AI assistants were used for coding support, experiment monitoring, and figure-layout iteration. The manuscript was primarily written by the authors. AI assistance in writing covered language polishing, editorial revision, and formatting. The authors designed the research questions and protocols and fixed the scientific evaluation criteria. They verified artifact hashes and numerical claims against the raw outputs and take full responsibility for the content of the paper.

ETHICS STATEMENT

This study concerns agent recruitment in a controlled programming-task environment. Cost is reported as an explicitly defined proxy, and the results do not establish reductions in deployment expenditure or environmental impact. Use or redistribution of the underlying models and benchmark materials remains subject to their respective terms. The method does not establish safe autonomous behavior in open-ended deployment settings.

REPRODUCIBILITY STATEMENT

Appendix A provides score definitions, proofs, and training pseudocode. Appendix B records benchmark construction, cleaning, models and settings; the source package includes the benchmark records. Appendix C defines metrics and reports seed summaries, budget labels, rounding, and evaluation scope. Appendix D contains behavioral and branch diagnostics. Appendix C also states the scope of checkpoint evaluation and statistical comparisons.

REFERENCES

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program Synthesis with Large Language Models, 2021. URL https://arxiv.org/abs/2108.07732. arXiv:2108.07732.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating Large Language Models Trained on Code, 2021. URL https://arxiv.org/abs/2107.03374. arXiv:2107.03374.

Yiqun Chen, Erhan Zhang, Lingyong Yan, Shuaiqiang Wang, Jizhou Huang, Dawei Yin, and Jiaxin Mao. MAO-ARAG: Multi-Agent Orchestration for Adaptive Retrieval-Augmented Generation, 2025. URL https://arxiv.org/abs/2508.01005. arXiv:2508.01005.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The Llama 3 Herd of Models, 2024. URL https://arxiv.org/abs/2407.21783. arXiv:2407.21783.

Haoyang Hong, Jiajun Yin, Yuan Wang, Jingnan Liu, Zhe Chen, Ailing Yu, Ji Li, Zhiling Ye, Hansong Xiao, Yefei Chen, Hualei Zhou, Yun Yue, Minghui Yang, Chunxiao Guo, Junwei Liu, Peng Wei, and Jinjie Gu. Multi-Agent Deep Research: Training Multi-Agent Systems with M-GRPO, 2025. URL https://arxiv.org/abs/2511.13288. arXiv:2511.13288.

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, 2025. URL https://arxiv.org/abs/2509.22047. arXiv:2509.22047.

Gang Li, Yan Chen, Ming Lin, and Tianbao Yang. DRPO: Efficient Reasoning via Decoupled Reward Policy Optimization, 2025. URL https://arxiv.org/abs/2510.04474. arXiv:2510.04474.

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, Yejin Choi, Jan Kautz, and Pavlo Molchanov. GDPO: Group reward-Decoupled Normalization Policy Optimization for Multi-reward RL Optimization, 2026. URL https://arxiv.org/abs/2601.05242. arXiv:2601.05242.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding R1-Zero-Like Training: A Critical Perspective, 2025. URL https://arxiv.org/abs/2503.20783. arXiv:2503.20783.

Cheng Qian, Zuxin Liu, Shirley Kokane, Akshara Prabhakar, Jielin Qiu, Haolin Chen, Zhiwei Liu, Heng Ji, Weiran Yao, Shelby Heinecke, Silvio Savarese, Caiming Xiong, and Huan Wang. xRouter: Training Cost-Aware LLMs Orchestration System via Reinforcement Learning, 2025. URL https://arxiv.org/abs/2510.08439. arXiv:2510.08439.

Jianhao Ruan, Zhihao Xu, Yiran Peng, Fashen Ren, Zhaoyang Yu, Xinbing Liang, Jinyu Xiang, Yongru Chen, Bang Liu, Chenglin Wu, Yuyu Luo, and Jiayi Zhang. AOrchestra: Automating Sub-Agent Creation for Agentic Orchestration, 2026. URL https://arxiv.org/abs/2602.03786. arXiv:2602.03786.

Mohammad Mahdi Salmani-Zarchi, Zahra Rahimi, Heshaam Faili, and Mohammad Javad Dousti. MDP-GRPO: Stabilized Group Relative Policy Optimization for Multi-Constraint Instruction Following, 2026. URL https://arxiv.org/abs/2606.06058. arXiv:2606.06058.

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, 2024. URL https://arxiv.org/abs/2402.03300. arXiv:2402.03300.

Hongjin Su, Shizhe Diao, Ximing Lu, Mingjie Liu, Jiacheng Xu, Xin Dong, Yonggan Fu, Peter Belcak, Hanrong Ye, Hongxu Yin, Yi Dong, Evelina Bakhturina, Tao Yu, Yejin Choi, Jan Kautz, and Pavlo Molchanov. ToolOrchestra: Elevating Intelligence via Efficient Model and Tool Orchestration, 2025. URL https://arxiv.org/abs/2511.21689. arXiv:2511.21689.

Zhexuan Wang, Yutong Wang, Xuebo Liu, Liang Ding, Miao Zhang, Jie Liu, and Min Zhang. AgentDropout: Dynamic agent elimination for token-efficient and high-performance LLM-based multi-agent collaboration. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 24013–24035, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1170. URL https://aclanthology.org/2025.acl-long.1170/.

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, 2025. URL https://arxiv.org/abs/2505.09388. arXiv:2505.09388.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. DAPO: An Open-Source LLM Reinforcement Learning System at Scale, 2025. URL https://arxiv.org/abs/2503.14476. arXiv:2503.14476.

Guibin Zhang, Yanwei Yue, Zhixun Li, Sukwon Yun, Guancheng Wan, Kun Wang, Dawei Cheng, Jeffrey Xu Yu, and Tianlong Chen. Cut the Crap: An Economical Communication Pipeline for LLM-based Multi-Agent Systems, 2024. URL https://arxiv.org/abs/2410.02506. arXiv:2410.02506.

Guibin Zhang, Luyang Niu, Junfeng Fang, Kun Wang, Lei Bai, and Xiang Wang. Multi-agent Architecture Search via Agentic Supernet, 2025. URL https://arxiv.org/abs/2502.04180. arXiv:2502.04180.

Jiamu Zhang, Lingxi Zhang, Pengjun Lu, Qiyue Zhang, Yu-Neng Chuang, Zhengchen Li, Shuai Xu, Vipin Chaudhary, and Hanjie Chen. Rethinking the Evaluation of Efficiency Methods for Multi-Agent Systems, 2026. URL https://arxiv.org/abs/2609.05933. arXiv:2609.05933.

A Score Construction and Normalization Analysis

A.1 Team-level quantities

For a trajectory completing $n_{\text{pass},i}$ of K_x subtasks with $m_i \le 4$ workers,

$$q_i = \frac{n_{\text{pass},i}}{K_x}, \qquad y_i = \mathbf{1}[n_{\text{pass},i} = K_x], \qquad T_i = 1 + m_i. \tag{A.1}$$

The controller is included in T_i; positive quality may be partial success. For slot actions $\mathcal{A}_{ij}$, the cost proxy is

$$\begin{aligned} L_{ij} &= \sum_{a \in \mathcal{A}_{ij}} \max\left(1, \left\lfloor \frac{|\operatorname{str}(a)|}{4} \right\rfloor\right), \qquad P_i = \max_j L_{ij}, \\ c_i &= P_i + \alpha m_i, \qquad \alpha = 50. \end{aligned} \tag{A.2}$$

This character-length proxy is distinct from actual tokens, latency or monetary cost.

A.2 Method definitions and controls

Let $\bar{v}$ and σ_v be the within-task mean and population SD. Define

$$z_\epsilon(v_i) = \begin{cases} (v_i - \bar{v})/(\sigma_v + \epsilon), & \sigma_v > \epsilon, \\ 0, & \sigma_v \le \epsilon, \end{cases} \qquad \epsilon = 10^{-6}. \tag{A.3}$$

The states are zero-quality ties ($\sigma_q \le \epsilon, \bar{q} \le \epsilon$), positive-quality ties ($\sigma_q \le \epsilon, \bar{q} > \epsilon$), and varying quality. In Table A.1, $u_i = q_i - \lambda c_i/50$, $d_i = z_\epsilon(q_i) - \lambda z_\epsilon(c_i)$, and λ follows the reported experiment.

Table A.1: Intermediate scores used by each method. A quality tie is determined by the numerical criterion above. The downstream GRPO advantage calculation is retained.

Method	**Zero-quality tie**	**Positive-quality tie**	**Varying quality**
Quality-Only (QO)	q_i	q_i	q_i
Direct Cost (DC)	u_i	u_i	u_i
Guided Direct Cost (GDC)	0	$-\lambda z_\epsilon(T_i)$	u_i
Normalized Cost (NC)	d_i	d_i	d_i
Quality-Guided Cost (QGC; Ours)	0	$-\lambda z_\epsilon(T_i)$	d_i
Tie-Suppressed Cost (TSC)	0	0	d_i

Guidance changes both tied-quality branches; GDC–QGC isolates score combination at the configured coefficients, not matched effective penalties. TSC is a local control, not DAPO's complete resampling recipe (Yu et al., 2025).

A.3 Finite-epsilon analysis

For $s_i = q_i - \beta c_i$, $\beta > 0$, equal quality and $\sigma_c > 0$, centering gives $s_i - \bar{s} = -\beta(c_i - \bar{c})$ and $\sigma_s = \beta\sigma_c$. Hence GRPO-style normalization (Shao et al., 2024) gives

$$A_i = -\frac{\beta(c_i - \bar{c})}{\beta\sigma_c + \epsilon_{\text{out}}} = -\rho\frac{c_i - \bar{c}}{\sigma_c}, \qquad \rho = \frac{\beta\sigma_c}{\beta\sigma_c + \epsilon_{\text{out}}}. \tag{A.4}$$

$$1 - \rho = \frac{\epsilon_{\text{out}}}{\beta\sigma_c + \epsilon_{\text{out}}}. \tag{A.5}$$

For $\delta \in (0,1)$, $\beta\sigma_c \ge (1-\delta)\epsilon_{\text{out}}/\delta$ gives $1-\rho \le \delta$; constant cost instead gives zero centered scores. The result concerns fixed-group task advantages, without implying training collapse or invariance of complete policy updates.

A.4 Training integration

Final scores are computed once per independent team and broadcast before GRPO normalization. With $G = 8$, 40 slot rows represent eight teams. Section 3.4 explains downstream scaling, other losses and inference.

Algorithm 1 Quality-Guided Score Construction for a Task Group

Input: Final quality scores q, estimated costs c, and team sizes T for G independent team trajectories
Parameters: $\lambda > 0$, $\epsilon = 10^{-6}$
if $\mathrm{std}_{\mathrm{population}}(q) \le \epsilon$ **then**
  if $\mathrm{mean}(q) \le \epsilon$ **then**
    Set all team scores to 0
  else
    Set scores to $-\lambda z_\epsilon(T)$
  end if
else
  Set scores to $z_\epsilon(q) - \lambda z_\epsilon(c)$
end if
Assign team scores to the corresponding sample rows.
Run the existing GRPO advantage computation and policy update.

A.5 Distinctions from fixed scalar penalties

Quality-state selection. For a zero-quality group with nonconstant cost, DC with $\beta > 0$ gives

$$A_i^{\mathrm{DC}} = -\frac{\beta(c_i - \bar{c})}{\beta\sigma_c + \epsilon_{\mathrm{out}}}, \tag{A.6}$$

which is nonzero for some trajectories, whereas QGC gives zero task advantages. With $\beta = 0$, DC cannot reproduce QGC's nonconstant $-\lambda z_\epsilon(T_i)$ in a positive-quality tie with $\sigma_T > \epsilon$. Thus no fixed nonnegative DC coefficient reproduces both cases.

Separate normalization. For varying-quality group g, let $\widetilde{q}_i = q_i - \bar{q}_g$, $\widetilde{c}_i = c_i - \bar{c}_g$, and $\sigma_{q,g}, \sigma_{c,g} > \epsilon$. Exactly,

$$s_i^{\mathrm{QGC}} = \frac{\widetilde{q}_i - \beta_g \widetilde{c}_i}{\sigma_{q,g} + \epsilon}, \qquad \beta_g = \lambda_{\mathrm{QGC}} \frac{\sigma_{q,g} + \epsilon}{\sigma_{c,g} + \epsilon}. \tag{A.7}$$

GDC instead uses $\widetilde{q}_i - \beta_{\mathrm{GDC}} \widetilde{c}_i$, with fixed $\beta_{\mathrm{GDC}} = \lambda_{\mathrm{GDC}}/50$. For linearly independent centered vectors, equality up to positive scaling fixes the quality coefficient and then requires $\beta_{\mathrm{GDC}} = \beta_g$. One fixed coefficient cannot match two such groups with different β_g. Positive normalization preserves direction; collinearity or equal effective ratios can remove this distinction. These are conditional score-rule distinctions, not a performance guarantee over tuned baselines.

B Benchmark and Recorded Settings

B.1 Task construction and cleaning

The diagnostic benchmark bundles MBPP (Austin et al., 2021) and HumanEval (Chen et al., 2021) subtasks. Source IDs are shuffled with seed 20260817, with 15% reserved for validation before packing 2,000 training and 200 validation bundles. Cleaning leaves 948 training and 148 validation source subtasks with disjoint IDs; this excludes neither semantic overlap nor pretraining exposure. These diagnostic counts are separate from Tables 1–3.

Packing weights for Anchor/Trap/k2/k3/k4 are 0.30/0.20/0.20/0.15/0.15. Table B.1 gives the cleaned composition; structural team references are construction labels, not optimal policies.

Table B.1: Composition of the cleaned evaluation set. Each retained bundle has three decoding repetitions.

Stratum	Subtasks per bundle	Bundles	Evaluations	Structural team reference
Anchor	1	60	180	1
Trap	1	38	114	1
Two-subtask	2	40	120	3
Three-subtask	3	30	90	4
Four-subtask	4	26	78	5
Total		194	582	

Composite and Trap descriptions are LLM-rewritten with specifications and tests retained, then checked for format and test markers. Replication uses the frozen descriptions. Removing six duplicate-description bundles independently of outcomes leaves 194 tasks per method; excluded IDs accompany the source package.

B.2 Models and settings

Model variants, current training seeds and coefficients appear in Section 4.1 (Yang et al., 2025; Grattafiori et al., 2024). The diagnostic runs use eight teams per task, eight task groups per batch, up to four workers, $\alpha = \alpha_{\text{ref}} = 50$, $\epsilon = 10^{-6}$, and $\lambda = 0.1$. Evaluation uses step 250, decoding seeds 101/102/103, temperature 1.0, prompt/response limits of 1,024/1,536 tokens, task batches of eight and one team per task per decoding seed. These settings describe the diagnostic cohort; branch exceptions and evaluation scope appear in Appendices D.2 and C.4.

C Metrics and Supporting Results

C.1 Metrics and aggregation

For full-success indicator $y_{a,s,t,r}$, n_s tasks and R_s decoding repetitions per training seed,

$$\widehat{p}_{a,s} = \frac{1}{n_s R_s} \sum_{t=1}^{n_s} \sum_{r=1}^{R_s} y_{a,s,t,r}, \qquad \overline{p}_a = \frac{1}{S} \sum_{s=1}^{S} \widehat{p}_{a,s}. \tag{C.1}$$

Table 1 uses $S = 3$; mean cost includes failures with equal seed weighting. Success changes are $100(\overline{p}_{\text{QGC}} - \overline{p}_a)$ and cost changes are $100(\overline{c}_{\text{QGC}}/\overline{c}_a - 1)$, without intermediate rounding. The tables do not specify n_s, R_s or integer success counts; the diagnostic 194×3 evaluations form a separate cohort.

C.2 Operating points and reporting

Figure 3's budgets are dev constraints, not matched realized costs; its $0.75\times$ points match Table 1 after rounding. The points are not documented as an independent held-out test of equal-budget tuning.

Marginal means and contrasts are rounded separately: QGC–TSC success gives -0.28 pp from displayed cells versus reported -0.27 pp; Qwen3-4B cost gives -21.33% versus -21.35%. Both differences are compatible with the displayed precision.

C.3 Seed summaries and uncertainty

Sample SD uses denominator $S - 1$. Decoding repeats and shared-subtask bundles are not independent training runs. Paired outcomes, other budget/branch/cross-model repetitions, and quality-state counts/windows are unspecified; confidence intervals, paired significance, noninferiority and state-conditioned causal effects are not established.

Table C.1: Qwen3-8B training-seed summaries. Each cell is success (%) / mean estimated cost in raw proxy units. Table 1 aggregates these seeds with equal weight.

Method	λ	**Seed 0**	**Seed 1**	**Seed 2**
QO	—	43.82 / 987.6	45.31 / 1036.8	44.56 / 1008.9
GDC	0.1	44.24 / 739.2	44.83 / 722.5	43.17 / 731.8
NC	0.1	40.36 / 690.4	41.79 / 675.9	40.92 / 708.7
QGC	0.03	46.31 / 724.8	44.62 / 699.1	47.15 / 736.4

C.4 Evaluation scope

Diagnostic evaluation resumed training at zero learning rate without strictly frozen weights; the outcome effect is unquantified. Common checkpoint, evaluator and unchanged-weight checks are undocumented for the aggregate tables. Cohorts remain separate; aggregate agreement does not establish matched-resource branch benefits.

D Behavioral Diagnostics

D.1 Recruitment and quality states

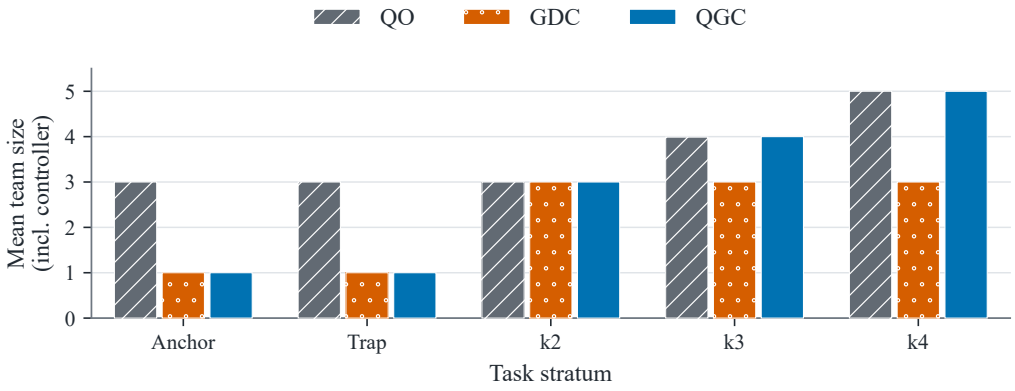


Figure 5: Historical Qwen3-8B mean team size, including the controller. Each stratum compares QO, GDC and QGC from left to right. Each checkpoint is evaluated on 194 tasks with three decoding repetitions (Table B.1). This diagnostic cohort is separate from Table 1.

In 320 groups from two historical Qwen3-8B QO runs, 26.6% have zero-quality ties, 55.6% positive ties and 17.8% varying quality. These frequencies motivate the rules; they are neither QGC frequencies nor causal effects.

D.2 Positive-quality tie controls

Qwen3-4B QGC–TSC diagnostics use $\lambda = 0.1$, steps 221–250. Seed 0 differs in model path, 300-step horizon and spawn hints. Seeds 1/2 match more closely but retain score-construction, coefficient and checkpoint-frequency differences. Section 4.5 reports the trade-offs; complete execution snapshots are unavailable.

At $\lambda = 0.3$, seed 0, over 150 steps, team-only feedback raises Qwen3-4B logged success and slot-weighted cost under matching selected settings. Slot means and dependent training windows describe training behavior, without establishing independent or matched-cost evaluation gains.