COST- AND QUALITY-AWARE PRUNING OF MULTI-AGENT LLM SYSTEMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-agent large language model systems often incur substantial inference costs, even when additional agents contribute little to the final answer. We study which agents can be removed from a deployed system while maintaining a specified quality requirement. We apply a novel extension of the classical random-walk model in which link traversals carry costs and rewards, using closed-form first-passage calculations to evaluate sequential execution. The process ends when the current answer is returned by a reporting manager. Traversal costs represent the accounted model-call cost, while quality increments represent changes in answer correctness whose sum equals final correctness. We first calibrate agent behavior, then evaluate candidate subsets, and construct a pruning sequence using predicted quality and cost changes. We evaluate candidate subsets using closed-form matrix calculations, thereby avoiding expensive trajectory simulations and additional model calls. Finally, we deploy the least costly candidate design whose bootstrap screening percentile meets the quality target. In controlled over-provisioned systems, the pruning method reduces accounted API cost by 73% on competition mathematics, 15% on code repair, and 10% on heterogeneous code repair. Additional mathematics experiments with Qwen, Kimi, and GLM yield cost reductions of 34.8%–73.5% at a quality target of .80. The selected designs achieve held-out accuracies at least as high as the corresponding full systems.

1 INTRODUCTION

A common response to difficult tasks in contemporary AI is to add capacity. In multi-agent large language model (LLM) systems, this often means adding solvers, critics, or rounds of debate (Wang et al., 2023; Du et al., 2024; Li et al., 2024). Even simple panel designs multiply calls by the number of agents and rounds: a five-agent panel running three rounds issues fifteen model calls to produce one answer. These calls continue to incur cost when several agents repeat information already supplied by their peers. Improvements in the underlying model can also make parts of an existing orchestration unnecessary. The result is a deployed system whose cost reflects its full configuration even though some components contribute negligibly to answer quality.

We study removal under a specified quality threshold in sequential worker–manager systems, where one worker acts at a time and a reporting manager returns the current answer. Parallel panels and branching orchestrations motivate the broader efficiency problem but require additional execution models. Our quantity of interest is the change in expected system quality and cost after deleting an agent. Individual accuracy alone does not describe how an agent repairs or damages the current answer, while direct call price omits downstream invocations. These marginal effects must also be estimated from limited calibration data.

The call bill can be represented as an accumulated first-passage cost. Each worker invocation contributes the cost of its model calls to the execution path, and the mean weighted first-passage time gives expected total cost. A second weighting assigns changes in answer correctness, whose sum is final correctness. We apply the closed forms of Le et al. (2026) to evaluate both expectations, including revisits and downstream routing, without enumerating or simulating the trajectories of each candidate design.

To transfer this representation across designs, we track the last active worker and the correctness of the current answer. Calibration estimates each agent's solve, repair, and damage rates, from which

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.

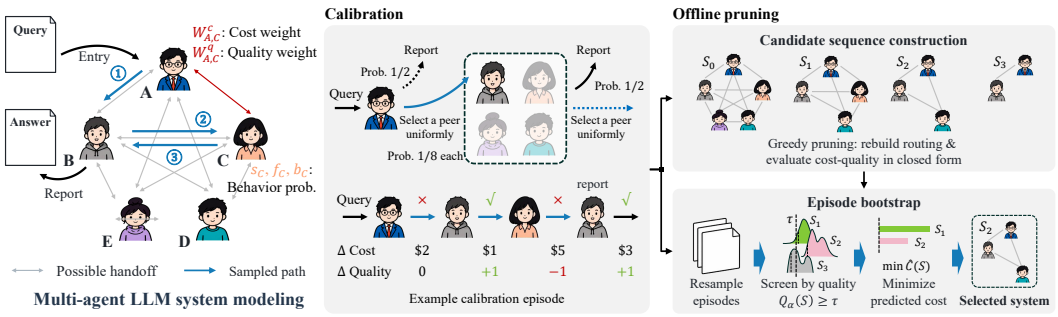


Figure 1: Overview of the proposed workflow. Calibration episodes provide agent behavior and cost estimates for closed-form evaluation and greedy pruning. Episode bootstrap screens the resulting candidates against a quality target, and the passing candidate with the lowest predicted cost is selected.

we construct candidate reduced chains under the specified routing policy. The contribution is a shared cost-and-quality reward construction and its use in a quality-constrained pruning workflow. Its predictions assume that the fitted conditional rates and mean call costs remain stable after pruning. A pointwise bootstrap screen describes calibration uncertainty, with limitations discussed below. For seven-agent systems, exact quality Shapley values provide an additional consistency check within the fitted model. Figure 1 provides an overview of our framework for cost- and quality-aware pruning of multi-agent LLM systems.

In this paper, we develop a framework for pruning sequential multi-agent LLM systems to reduce inference cost under a specified quality target. Our contributions are as follows:

- **Joint cost-and-quality evaluation:** We construct a shared reward model and apply existing first-passage closed forms to predict the cost and quality of candidate agent subsets.
- **Quality-constrained pruning:** We combine calibrated removal effects with greedy pruning and pointwise bootstrap screening to select reduced designs without additional model calls during candidate evaluation.
- **Controlled empirical evaluation:** We evaluate prediction transfer and compare pruning orders in three deliberately over-provisioned systems using real API responses. Marginal-efficiency selections reduce accounted deployment cost by 73% on mathematics, 15% on code repair, and 10% on heterogeneous code repair, with held-out accuracies at least as high as their full-system references. Additional mathematics experiments with Qwen, Kimi, and GLM extend the evaluation to other provider configurations.

The paper is organized as follows. Section 2 reviews related work. Section 3 defines the execution model and its cost and quality rewards, and Section 4 develops their first-passage evaluation. Section 5 presents the pruning and screening procedure. Section 6 reports the empirical evaluation, and Section 7 concludes. The appendices provide derivations, experimental protocols, additional results, and limitations.

2 Related Work

Inference-time efficiency is commonly addressed by choosing among complete models or complete systems. FrugalGPT learns cascades that escalate difficult queries (Chen et al., 2024), while RouteLLM learns query-level routing between models of different cost and quality (Ong et al., 2025). Our method targets savings within a sequential multi-agent system and can be combined with routing or cascading.

Another line of work learns the communication structure. GPTSwarm optimizes language-agent computational graphs (Zhuge et al., 2024), G-Designer generates task-dependent topologies (Zhang et al., 2025b), and OMAC jointly optimizes agent function and collaboration structure (Li et al., 2026). MAGE uses score-based diffusion to generate communication graphs (Zhao et al., 2026), while RADAR explicitly targets communication redundancy through diffusion-based graph generation

(Zhang et al., 2026). AgentPrune prunes communication edges in a spatial-temporal message-passing graph (Zhang et al., 2025a), while AgentDropout learns round-specific agent elimination and communication-edge pruning (Wang et al., 2025). Our method uses one calibrated model to compare removals and select the candidate with the lowest predicted cost that passes a quality screen.

Removal-based attribution is a close comparison. Lu et al. (2026) define agent contributions relative to a coalition distribution, removal protocol, and target metric, compare leave-one-out and coalition-based scores, and use attribution to guide cost-reducing interventions. They also examine LLM-judge estimates that avoid rerunning ablations but can disagree with ablation-based values. Node-contribution backpropagation addresses corrupted agents (Wu et al., 2026). Reliability estimation in classical aggregation and dependence modeling for LLM judges offer complementary perspectives (Dawid & Skene, 1979; Whitehill et al., 2009; Balasubramanian et al., 2026). Studies of portfolio optimization also show that estimation error can affect decisions (Michaud, 1989; Chopra & Ziemba, 1993; DeMiguel et al., 2009). We use a bootstrap screen to summarize uncertainty in estimated quality, with coverage and transfer limitations. After calibration, our method evaluates removals analytically without further system rollouts or LLM-judge calls.

3 Problem Setting

3.1 State and transition model

Let $\mathbb{V} = \{1, \dots, n\}$ be the worker set and $r^\star$ the reporting state, where a manager returns the current answer without modifying it. For a retained set $S \subseteq \mathbb{V}$, define the worker-state space $\mathcal{W}_S$ and the full state space $\mathcal{X}_S$ by

$$\mathcal{W}_S = S \times \{0, 1\}, \qquad \mathcal{X}_S = \mathcal{W}_S \cup \{r^\star\}.$$

We model execution by a discrete-time Markov chain $\{J_t : t \geq 0\}$ on $\mathcal{X}_S$. A worker state $J_t = (j, z)$ records that worker j has just acted and that the resulting answer has correctness z, with $z = 1$ for correct and $z = 0$ for incorrect. An operational episode ends when the answer is reported. Correctness is assessed offline; the manager neither inspects it nor requests further revision.

For worker j, the solve rate s_j is the probability of a correct initial answer. The repair rate f_j is the probability of a correct revision given an incorrect input, and the damage rate b_j is the probability of an incorrect revision given a correct input. Its revision kernel is

$$\boldsymbol{K}_j = \begin{pmatrix} 1 - f_j & f_j \\ b_j & 1 - b_j \end{pmatrix},$$

with rows and columns indexing incoming and outgoing correctness in the order $(0, 1)$.

Let $\mathcal{A} \subseteq \mathbb{V}$ contain the workers eligible for the initial call. An admissible design retains at least one such worker, so $S \cap \mathcal{A} \neq \varnothing$. Define

$$\nu_S(j) = \frac{\mathbf{1}\{j \in S \cap \mathcal{A}\}}{|S \cap \mathcal{A}|}, \qquad \boldsymbol{\alpha}_S(j, z) = \nu_S(j) \begin{cases} 1 - s_j, & z = 0, \\ s_j, & z = 1. \end{cases} \tag{1}$$

Thus the column vector $\boldsymbol{\alpha}_S$ describes the state after the initial call. The solve rate s_j is conditional on selecting worker j.

After an answer is produced, the system reports with probability $p_S = 1/2$ when $|S| > 1$. Otherwise it forwards uniformly to a surviving peer, which revises the answer. A singleton reports immediately, so $p_S = 1$ when $|S| = 1$. The worker transitions are

$$\begin{aligned} \boldsymbol{P}_S\big((j, z), (k, z')\big) &= \frac{1 - p_S}{|S| - 1} \boldsymbol{K}_k(z, z'), \quad |S| > 1,\ k \neq j, \\ \boldsymbol{P}_S\big((j, z), r^\star\big) &= p_S, \qquad\qquad\qquad\quad j \in S,\ z \in \{0, 1\}. \end{aligned} \tag{2}$$

Singletons have no peer transitions. To analyze successive episodes, we give the reporting state the reset law

$$\boldsymbol{P}_S\big(r^\star, (j, z)\big) = \boldsymbol{\alpha}_S(j, z). \tag{3}$$

This transition discards the preceding answer and represents the initial call of an independent new episode. All unspecified probabilities are zero. Upon deletion, the corresponding worker states are removed and the eligible-entry and peer distributions are renormalized. Appendix A gives the full matrix.

3.2 SEQUENTIAL COST AND QUALITY REWARDS

For $a \in \{\mathrm{c}, \mathrm{q}\}$, write $X_t^a = X_t^a(J_{t-1}, J_t)$ for the reward on a transition. The cost reward $X_t^{\mathrm{c}}(J_{t-1}, J_t)$ is the accounted cost incurred on that transition. An entry transition charges the initial call, a worker-to-worker transition charges the revision, and a reporting transition costs zero. All constituent model calls are charged, including multiple samples. Input and output tokens are priced separately, and completion usage includes reasoning tokens. Appendix E.6 specifies the accounted costs and imputed usage.

Quality rewards use an empty-answer baseline at each reset:

$$X^{\mathrm{q}}\big(r^\star, (j,z)\big) = z, \quad X^{\mathrm{q}}\big((j,z), r^\star\big) = 0, \quad X^{\mathrm{q}}\big((j,z),(k,z')\big) = z' - z. \tag{4}$$

Thus repair contributes $+1$, damage contributes -1, and preservation contributes zero. Starting at $J_0 = r^\star$, define the first positive return

$$T_{r^\star}^+ = \inf\{t \geq 1 : J_t = r^\star\}. \tag{5}$$

This cycle contains the initial call and every revision of one episode. Its total quality reward and call cost are

$$Y_{\mathrm{q}} = \sum_{t=1}^{T_{r^\star}^+} X_t^{\mathrm{q}} \in \{0,1\}, \qquad Y_{\mathrm{c}} = \sum_{t=1}^{T_{r^\star}^+} X_t^{\mathrm{c}}. \tag{6}$$

To see why the quality reward equals reported correctness, write $J_t = (j_t, z_t)$ for the worker states before reporting. Then $Y_{\mathrm{q}} = z_1 + \sum_{t=2}^{T_{r^\star}^+ - 1}(z_t - z_{t-1}) = z_{T_{r^\star}^+ - 1}$. For a singleton, the revision sum is empty and $Y_{\mathrm{q}} = z_1$. The quality increments therefore telescope to final correctness, while the costs account for every call exactly once. These identities hold on every execution path; predicting their expectations requires the fitted transition probabilities and mean call costs. The design objectives are

$$Q(S) = \mathbb{E}_{r^\star}[Y_{\mathrm{q}}], \qquad C(S) = \mathbb{E}_{r^\star}[Y_{\mathrm{c}}].$$

4 FIRST-PASSAGE ANALYSIS

4.1 EXPECTED COST AS A MEAN WEIGHTED FIRST-PASSAGE TIME

The expressions $Q(S)$ and $C(S)$ can be interpreted as the accumulated weights of a random walk on the agent set until reaching reporting node r^*, where the random walker incurs cost/reward $X_t^a = X^a(J_{t-1}, J_t)$ when traversing from J_{t-1} to J_t. Following the framework in (Le et al., 2026), $Q(S)$ and $C(S)$ can be modeled as mean weighted first-passage times from r^* to r^*. More specifically, we can introduce the mean accumulated weight until first reaching y from x, denote by $\boldsymbol{M}_{\mathrm{q}}$ and $\boldsymbol{M}_{\mathrm{c}}$, respectively, where the weights along a random walk are accumulated as in (6). As shown in Le et al. (2026), $\boldsymbol{M}_{\mathrm{q}}$ and $\boldsymbol{M}_{\mathrm{c}}$ can be computed from the Markov transition matrix of J_t and the matrix of expected weights

$$\boldsymbol{W}^a(x,y) = \mathbb{E}[X_t^a(J_{t-1}, J_t) \mid J_{t-1} = x, J_t = y], \qquad a \in \{\mathrm{c}, \mathrm{q}\}, \tag{7}$$

with zero assigned to impossible transitions; see Appendix B for details. Solving $\boldsymbol{M}_{\mathrm{q}}$ and $\boldsymbol{M}_{\mathrm{c}}$ provides

$$Q(S) = \boldsymbol{M}_{\mathrm{q}}(r^\star, r^\star), \qquad C(S) = \boldsymbol{M}_{\mathrm{c}}(r^\star, r^\star). \tag{8}$$

Appendix B gives the full-matrix expression, using the formulas from Le et al. (2026), and equivalent stationary identities.

We construct the two weight matrices separately. The quality weights are determined by the states: on every possible transition, $\boldsymbol{W}^{\mathrm{q}}(x,y) = X^{\mathrm{q}}(x,y)$, with X^{q} given in (4). Thus $\boldsymbol{W}^{\mathrm{q}}$ requires no separate estimation.

For the cost matrix, we average the accounted call costs observed during calibration for each receiving worker and call type. Let $\widehat{c}_j^{\,\mathrm{entry}}$ denote worker j's mean initial-call cost and $\widehat{c}_{j,z}^{\,\mathrm{rev}}$ its mean revision cost conditional on incoming correctness z. We use these sample means to estimate the cost weights in (7) as

$$\widehat{\boldsymbol{W}}^{\mathrm{c}}\big(r^\star, (j,z)\big) = \widehat{c}_j^{\,\mathrm{entry}}, \quad \widehat{\boldsymbol{W}}^{\mathrm{c}}\big((j,z),(k,z')\big) = \widehat{c}_{k,z}^{\,\mathrm{rev}}, \quad \widehat{\boldsymbol{W}}^{\mathrm{c}}\big((j,z), r^\star\big) = 0,$$

for possible transitions, with zero assigned to impossible transitions. This specification pools costs across outgoing correctness outcomes and, for revisions, across preceding workers. It therefore approximates the transition-conditional means in (7) by receiving-worker and call-type means.

The transition matrix $\boldsymbol{P}_S$ is fitted separately from the correctness outcomes. Each solve, repair, and damage rate is Laplace-smoothed: a successes in N relevant calibration events give $(a+1)/(N+2)$. These fitted rates and the routing above make the analytic chain finite and irreducible, so it has a unique stationary distribution. Correctness labels are measured offline and are never sent to a model.

We keep the fitted rates and cost means fixed when evaluating retained subsets, while renormalizing the routing probabilities. Predictions therefore assume that these conditional rates and mean costs transfer after pruning. Changes in the problem mix or the retained design may violate this assumption.

4.2 Dependence of cost on routing

The distinction between direct and routed cost is structural. An inexpensive agent can increase total cost by forwarding to an expensive agent, while an agent with the same local price can reduce total cost by reporting early.

Proposition 1. *Across general sequential routing policies, no criterion based only on the retained agents and their per-call prices can always rank designs correctly by expected cost. There are two designs on the same agents, with the same per-call prices and expected reported quality, whose expected total costs differ by more than a factor of two.*

The proof in Appendix C changes the probability of visiting an expensive worker while fixing the roster, call prices, and quality behavior. A criterion using only those fixed inputs cannot distinguish the two expected bills. This counterexample concerns general routing policies, beyond the uniform policy evaluated here. It does not establish optimality of a particular pruning ranking or identify the cause of an empirical difference between retained subsets.

5 Pruning under Calibration Uncertainty

5.1 Quality-constrained selection

Let $\widehat{Q}(S)$ and $\widehat{C}(S)$ be the first-passage expectations evaluated at the calibrated rates and prices. The deployment specifies a quality target τ and a lower-tail probability $\alpha \in (0,1)$, with $\alpha = .05$ in the experiments. A greedy removal sequence $\widehat{\mathcal{P}} = (S_0, S_1, \ldots, S_\ell)$ is constructed from the point estimates, with $S_0 = \mathbb{V}$ and $S_r \cap \mathcal{A} \neq \varnothing$ throughout. For each member of this fixed sequence, calibration episodes are resampled and the quality rates and $\widehat{Q}(S)$ are recomputed. Denote their pointwise α-quantile by $\underline{Q}_\alpha(S)$, termed the screening bound. We select

$$\widehat{S} = \arg\min_{S \in \widehat{\mathcal{P}}} \widehat{C}(S) \quad \text{subject to} \quad \underline{Q}_\alpha(S) \geq \tau. \tag{9}$$

If no candidate passes the screen, the implementation returns S_0 as a fallback without a passing bound. The percentile is a heuristic summary of refitting variation along the estimated sequence, not a confidence guarantee for the selected design. Simultaneous or post-selection coverage requires additional control. Behavioral changes after pruning and shared error identities omitted from the state also lie outside the screen. The held-out split is excluded from subset selection and reserved for evaluating transfer.

The constrained form in Equation (9) reflects how the deployment requirement is specified. An operator can usually state a minimum acceptable accuracy, while a scalar exchange rate between accuracy and tokens is harder to justify. A Lagrangian relaxation uses $\widehat{C}(S) - \lambda(\underline{Q}_\alpha(S) - \tau)$, with $\lambda \geq 0$ measured in cost per unit of the quality bound. Discrete candidate sets can admit several trade-off slopes, so we report the selections at several quality targets directly.

All candidates share the same episode resamples, preserving within-episode dependence and pairing their comparisons, while costs remain fixed. The two episodes from one problem are separate resampling units, so dependence at the problem level is not captured. A rate with no observations stays at its smoothing prior, and resampling an all-success sample cannot represent unseen failures. These limitations can make the screening bound too optimistic even before subset selection.

5.2 Constructing a tractable candidate sequence

Starting from the full set, we remove one worker at a time while retaining an entry worker. Define $\Delta Q_j = \widehat{Q}(S) - \widehat{Q}(S \setminus \{j\})$ and $\Delta C_j = \widehat{C}(S) - \widehat{C}(S \setminus \{j\})$. For positive ΔC_j, the ratio $\eta_j(S) = \Delta Q_j / \Delta C_j$ measures quality sacrificed per unit cost saved. The implemented heuristic first gives equal priority to all candidates with $\Delta Q_j \leq 0$, regardless of cost change. If none exist, it maximizes $\Delta C_j / \max(\Delta Q_j, 10^{-6})$. Ties follow roster order, and the quantities are recomputed after deletion. Consequently a step may increase predicted cost; the later minimum-cost selection evaluates the entire sequence. Algorithm 1 in Appendix D summarizes the complete greedy construction, bootstrap screening, and final selection procedure.

The entire sequence is evaluated because feasibility can change nonmonotonically. Deleting a harmful agent may raise quality after an earlier deletion lowered it. Separating ranking from selection also makes comparisons interpretable. Competing rules propose different sequences, while all of them use the same lower bound and the same minimum-cost selection criterion. The heuristic searches only the greedy sequence. The straightforward implementation performs $O(n^2)$ candidate evaluations, each cubic in the number of retained states, giving an $O(n^5)$ worst-case arithmetic cost for the full greedy sequence.

All candidate evaluations after calibration use matrix computations. For seven agents we also compute exact quality Shapley values over all coalitions as a consistency check on attribution in the calibrated model (Shapley, 1953; Lu et al., 2026).

6 Experiments

6.1 Experimental setting

We study how well the calibrated first-passage model transfers from a full system to reduced designs and whether quality-constrained selection lowers accounted deployment cost. The experiments use three seven-worker systems with controlled redundancy. Across these rosters, controlled redundancy includes near-duplicate workers, intentionally harmful revisers, and higher-priced model tiers. These features provide known opportunities for pruning.

We estimate solve, repair, damage, and call-cost quantities from the full roster. Four ranking rules then construct candidate removal sequences. For each sequence, Equation (9) selects the lowest-cost candidate whose pointwise lower percentile reaches the target. We run the selected designs and the full system on a held-out split reserved from model fitting and design selection. Every system uses 60 calibration problems with two episodes per problem and 40 held-out problems. Episodes begin at an eligible entry worker. For designs with more than one worker, each action is followed by reporting with probability one half or forwarding to a uniformly sampled retained peer. Singletons report immediately. A report is forced after five revisions. The analytic model omits this depth cap, so its predictions approximate the executed protocol.

The mathematics pool contains equal numbers of MATH level-5 problems (Hendrycks et al., 2021) and AIME problems drawn from the 1983–2023 competitions (Mathematical Association of America, 2026). The two code systems use the same 100 stdin-judged problems from LiveCodeBench v6 (Jain et al., 2025) under a local grading protocol. Code repair varies worker roles. The heterogeneous system varies reasoning configuration, sampling, and model tier among its entry workers. Table 1 summarizes the common design. Appendix E gives the rosters, prompts, grading, failure handling, and accounting rules.

Table 1: Experimental setting. Every system uses 120 calibration episodes and 40 held-out problems. The entry and revision columns give the numbers of entry workers and revision-only workers.

System	Task pool	Entry / revision	Token cap	Targets τ
Mathematics	MATH level 5 + AIME	4 / 3	28k	.70, .80, .90
Code repair	LiveCodeBench v6	4 / 3	64k	.40, .50, .60
Heterogeneous	LiveCodeBench v6	5 / 2	64k	.40, .50, .60

Table 2: Fit on the full calibration designs. Cost is accounted dollars per 100 problems.

System	Predicted Q	Calibration accuracy	Predicted cost	Calibration cost
Mathematics	.869	.908	.266	.237
Code repair	.811	.825	1.666	1.424
Heterogeneous	.775	.792	1.742	1.486

6.2 Comparison rules and reported quantities

The proposed marginal-efficiency rule recomputes the removal effects after each deletion, using the priority for nonpositive quality loss and the stabilized cost-to-quality ratio specified in Section 5. We compare it with three alternatives. *Individual accuracy* removes the worker with the lowest Laplace-smoothed fraction of correct outputs pooled across its observed entry and revision roles. *Direct cost* removes the worker with the largest overall mean call price. *Random* follows one fixed-seed removal order. All four rules use the same route-aware quality and cost evaluator, pointwise bootstrap screen, and final minimum-cost selection rule. The comparison varies the removal order with evaluation and selection fixed.

For every deployed design we report fitted quality $\widehat{Q}$, the pointwise 5% screening percentile, held-out accuracy, and accounted held-out cost. The first two quantities use calibration data. Accuracy and realized cost come from the 40 held-out executions. Savings are measured relative to the held-out cost of the corresponding full system. Appendix F reports the complete target-dependent selections.

6.3 Prediction transfer

On the full calibration designs, fitted quality is within 3.9, 1.4, and 1.6 percentage points of calibration accuracy for mathematics, code repair, and heterogeneous code, as shown in Table 2. We compare fitted predictions with held-out outcomes and check whether the observed accuracies of screen-passing designs reach the specified targets. The algebra, code-repair pro, and heterogeneous pro designs have screening percentiles of .958, .758, and .696. Their held-out accuracies are .950, .850, and .850. Each selected design exceeds the highest tested target for its system and the held-out accuracy of its full-system reference, which is .900, .775, and .750, respectively.

The fitted cost model ranks each selected design below its full-system reference. Held-out execution preserves this pairwise ordering in all three systems. The corresponding realized cost reductions are 73%, 15%, and 10%, calculated from held-out costs. The quality screen and the within-system cost comparison therefore agree with the deployment outcomes for the three main selections. Figure 2 summarizes calibration-to-deployment transfer for all executed configurations, including the comparison-rule deployments analyzed in the following subsection and Appendix F. Nominal 95% Wilson intervals show the sampling uncertainty in each 40-problem held-out accuracy (Wilson, 1927).

6.4 Pruning outcomes

Table 3 summarizes deployments selected by the four removal orders; Appendix F gives additional selections. A target range indicates that the same design is selected at every listed target. Among the rows in this table, each reduced design contains a single entry worker except the heterogeneous direct-cost selection at $\tau = .60$.

Figure 3 shows the candidates evaluated along the four removal sequences. Different rankings may revisit the same subset or converge to the same singleton. The highlighted points identify designs selected by the target-specific screen and minimum-cost rule. The crosses give the held-out outcomes of executed configurations.

In mathematics, marginal efficiency and the individual-accuracy and direct-cost orders select the algebra solver. It answers 38 of 40 held-out problems correctly, compared with 36 for the full system. Accounted cost falls from $\$0.482$ to $\$0.132$ per 100 problems.

In code repair, marginal efficiency retains the pro solver, attains .850 accuracy, and saves 15%. The individual-accuracy and random orders retain cheaper singletons and save 58% and 62%, respectively.

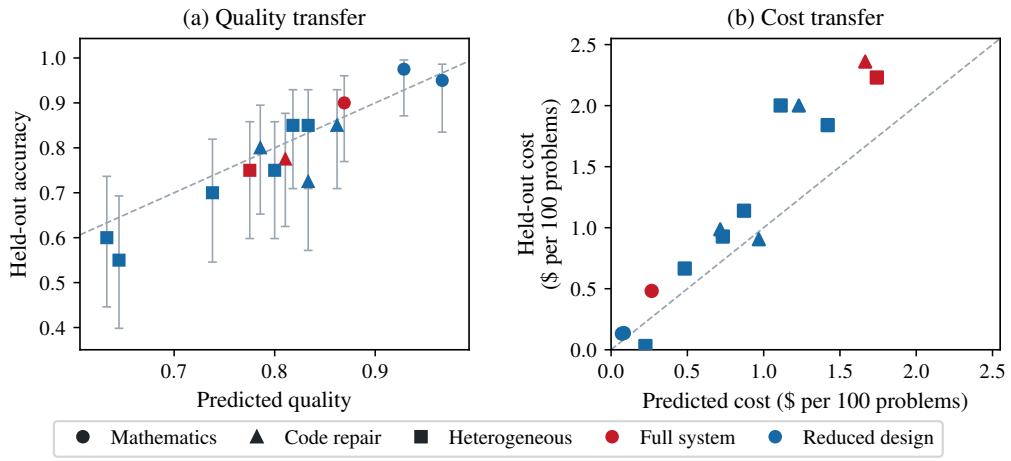


Figure 2: Prediction and held-out outcomes for all executed configurations. The left panel shows fitted quality and held-out accuracy with nominal 95% Wilson intervals. The right panel shows predicted and held-out accounted cost in dollars per 100 problems. Dashed lines indicate equality. Red markers identify full systems; the remaining markers include the marginal-efficiency selections and comparison-rule deployments.

Table 3: Held-out deployments. LCB is the pointwise 5% calibration screening percentile. Accuracy is measured on 40 held-out problems; cost is accounted dollars per 100 problems.

System	Rule	Design	Target(s)	LCB	Accuracy	Cost	Saving
Mathematics	full	full roster	–	–	.900	.482	–
	marginal efficiency	algebra	.70–.90	.958	.950	.132	73%
Code repair	full	full roster	–	–	.775	2.361	–
	marginal efficiency	pro	.40–.60	.758	.850	2.001	15%
	individual accuracy	edge	.40–.60	.727	.725	.987	58%
	random	duplicate	.40–.60	.666	.800	.906	62%
Heterogeneous	full	full roster	–	–	.750	2.230	–
	marginal efficiency	pro	.40–.60	.696	.850	2.001	10%
	individual accuracy	full reasoning	.40–.60	.667	.750	.928	58%
	random	duo	.40–.60	.696	.850	1.840	17%
	direct cost	lite, full, trace, saboteur	.60	.664	.700	1.139	49%

Both designs pass all three tested targets. Marginal efficiency selects the highest-accuracy singleton among those deployed, but the edge and duplicate designs have lower predicted and held-out costs and pass the same screens. Thus the greedy sequence can miss cheaper feasible designs under Equation (9); accuracy above the target receives no additional reward in that objective.

In the heterogeneous system, marginal efficiency retains pro and attains .850 accuracy with a 10% saving. The random order retains duo with the same observed accuracy and a 17% saving. Individual accuracy retains the full-reasoning solver, attaining .750 at $0.928. At $\tau = .60$, direct cost retains four workers, including the saboteur, and attains .700 at $1.139. On this split, the individual-accuracy design is cheaper and more accurate. These comparisons do not establish superiority of marginal efficiency under the specified quality-constrained cost objective. Appendix F reports the remaining target-dependent selections.

The estimated break-even horizons for the one-time calibration expense are approximately 81, 475, and 779 deployment problems for the mathematics, code-repair, and heterogeneous marginal-efficiency selections, respectively, using their observed held-out cost differences. The code-repair random selection and the heterogeneous individual-accuracy selection reach the same point after 117 and 137 problems. Appendix F gives the calculation and the full table.

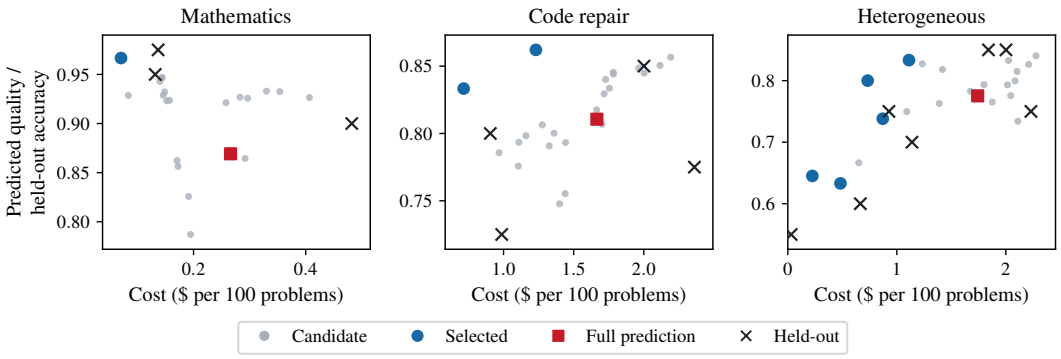


Figure 3: Predicted quality and cost for candidates on the four removal sequences. Blue circles denote selected reduced designs, red squares denote full-design predictions, and crosses denote held-out outcomes for executed configurations. Cost is accounted dollars per 100 problems.

6.5 Cross-provider robustness

We repeat the mathematics experiment with Qwen, Kimi, and GLM configurations while keeping the problem split, seven functional roles, routing, grading, calibration size, held-out set, and removal rules fixed. Table 4 reports the marginal-efficiency selections at $\tau = .80$. Each selected design passes the calibration screen and attains held-out accuracy at least as high as its full-system reference. The held-out cost reductions range from 34.8% to 73.5%.

Table 4: Cross-provider mathematics results at $\tau = .80$. Accuracy is measured on the same 40 held-out problems. Savings use the held-out costs recorded for each provider configuration.

Provider	Selected design	LCB	Full acc.	Selected acc.	Saving
Qwen	pro	.956	.850	1.000	34.8%
Kimi	algebra	.875	.950	.950	73.5%
GLM	algebra	.875	.725	.800	56.7%

Qwen and GLM reach the same endpoint as direct selection among passing entry singletons. For Kimi, the removal sequence selects algebra, with held-out accuracy .950 and cost $1.677 per 100 problems. Direct singleton selection chooses the duplicate solver, with accuracy .900 and cost $2.255. Removing the deliberately harmful reviser also lowers observed cost for all three configurations and raises accuracy for Qwen and GLM while preserving Kimi's accuracy. Appendix G reports the model configurations, absolute costs, target sensitivity, and candidate frontiers.

7 Conclusion

We presented a framework for quality-constrained pruning of sequential multi-agent LLM systems. A calibrated network model of sequential agent execution evaluates expected cost and quality using existing first-passage closed forms. Greedy removal and pointwise bootstrap screening select reduced designs using closed from evaluation on candidate pruned set, without the need for additional model calls.

In three controlled systems, marginal-efficiency selections save 73% on mathematics, 15% on code repair, and 10% on heterogeneous code repair, with held-out accuracies at least as high as their full-system references. Mathematics experiments with Qwen, Kimi, and GLM yield savings of 34.8%–73.5% at $\tau = .80$, again matching or exceeding full-system held-out accuracy.

AI USE STATEMENT

In preparing the final manuscript, we used generative AI tools only for language polishing and improving readability. For the final version, all research tasks beyond language polishing were independently completed by the authors without generative AI assistance. We reviewed all AI assisted language revisions to ensure that they preserve the intended meaning and technical accuracy. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

Krishnakumar Balasubramanian, Aleksandr Podkopaev, and Shiva Prasad Kasiviswanathan. Dependence-aware label aggregation for LLM-as-a-judge via ising models. In *International Conference on Machine Learning*, 2026.

Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to use large language models while reducing cost and improving performance. *Transactions on Machine Learning Research*, 2024.

Vijay K. Chopra and William T. Ziemba. The effect of errors in means, variances, and covariances on optimal portfolio choice. *The Journal of Portfolio Management*, 19(2):6–11, 1993. doi: 10.3905/jpm.1993.409440.

A. P. Dawid and A. M. Skene. Maximum likelihood estimation of observer error-rates using the EM algorithm. *Applied Statistics*, 28(1):20–28, 1979. doi: 10.2307/2346806.

Victor DeMiguel, Lorenzo Garlappi, and Raman Uppal. Optimal versus naive diversification: How inefficient is the $1/n$ portfolio strategy? *The Review of Financial Studies*, 22(5):1915–1953, 2009. doi: 10.1093/rfs/hhm075.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 11733–11763. PMLR, 2024.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 58791–58831, 2025.

Thao Le, Robbert van der Burg, Bernd Heidergott, Ines Lindner, and Alessandro Zocca. Random walks with traversal costs: Variance-aware performance analysis and network optimization. arXiv preprint arXiv:2603.26414, 2026.

Junyou Li, Qin Zhang, Yangbin Yu, Qiang Fu, and Deheng Ye. More agents is all you need. *Transactions on Machine Learning Research*, 2024.

Shijun Li, Hilaf Hasson, and Joydeep Ghosh. OMAC: A holistic optimization framework for LLM-based multi-agent collaboration. In *International Conference on Machine Learning*, 2026. Oral presentation.

Mingyu Lu, Yushan Huang, Chris Lin, and Su-In Lee. Agents that matter: Optimizing multi-agent LLMs via removal-based attribution. arXiv preprint arXiv:2605.27621, 2026.

Mathematical Association of America. American Invitational Mathematics Examination (AIME). MAA Invitational Competitions, 2026. URL `https://maa.org/maa-invitational-competitions/`. Accessed September 26, 2026.

Richard O. Michaud. The Markowitz optimization enigma: Is "optimized" optimal? *Financial Analysts Journal*, 45(1):31–42, 1989. doi: 10.2469/faj.v45.n1.31.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M. Waleed Kadous, and Ion Stoica. RouteLLM: Learning to route LLMs with preference data. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 34433–34448, 2025.

Lloyd S. Shapley. A value for n-person games. In Harold W. Kuhn and Albert W. Tucker (eds.), *Contributions to the Theory of Games II*, volume 28 of *Annals of Mathematics Studies*, pp. 307–318. Princeton University Press, Princeton, NJ, 1953. doi: 10.1515/9781400881970-018.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

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 *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 24013–24035. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.1170.

Jacob Whitehill, Paul Ruvolo, Tingfan Wu, Jacob Bergsma, and Javier Movellan. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *Advances in Neural Information Processing Systems*, volume 22, pp. 2035–2043, 2009.

Edwin B. Wilson. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158):209–212, 1927. doi: 10.1080/01621459.1927.10502953.

Chengcan Wu, Zhixin Zhang, Mingqian Xu, Zeming Wei, and Meng Sun. Securing multi-agent systems against corruptions via node contribution backpropagation. In *International Conference on Machine Learning*, 2026.

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. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 75389–75428, 2025a.

Guibin Zhang, Yanwei Yue, Xiangguo Sun, Guancheng Wan, Miao Yu, Junfeng Fang, Kun Wang, Tianlong Chen, and Dawei Cheng. G-designer: Architecting multi-agent communication topologies via graph neural networks. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 76678–76692. PMLR, 2025b.

Zhen Zhang, Wanjing Zhou, Juncheng Li, Hao Fei, Jun Wen, and Wei Ji. RADAR: Redundancy-aware diffusion for multi-agent communication structure generation. In *International Conference on Machine Learning*, 2026.

Guanhao Zhao, Wenbo Lu, Cheng Cheng, Zhenya Huang, Wei Song, Zhiding Liu, Runze Wu, and Enhong Chen. Navigating the energy landscape of collaboration: Multi-agent communication graph generation via score-based diffusion. In *International Conference on Machine Learning*, 2026.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. GPTSwarm: Language agents as optimizable graphs. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 62743–62767. PMLR, 2024.

A Overall Transition Probability Matrix

Fix an admissible retained set S with $m = |S|$, and order the states as $(\mathcal{W}_S, r^\star)$. The worker block is

$$\boldsymbol{R}_S\big((j,z),(k,z')\big) = \begin{cases} \dfrac{1-p_S}{m-1}\boldsymbol{K}_k(z,z'), & m>1,\ k\neq j, \\ 0, & m=1 \text{ or } k=j. \end{cases}$$

With $\boldsymbol{\alpha}_S$ from Equation (1) and $\mathbf{1}_{2m}$ the vector of ones, the analytic transition matrix is

$$\boldsymbol{P}_S = \begin{pmatrix} \boldsymbol{R}_S & p_S\mathbf{1}_{2m} \\ \boldsymbol{\alpha}_S^\top & 0 \end{pmatrix}. \tag{10}$$

Every row sums to one. The last row begins a fresh episode and carries its initial-call rewards. An operational episode stops on arrival at $r^\star$; the analytic reset is used to evaluate its first-return rewards.

The entry restriction affects only $\boldsymbol{\alpha}_S$, so the same block matrix covers entry and revision-only workers. When every retained worker is eligible for entry, $S \subseteq \mathcal{A}$, the expression simplifies to $\nu_S(j) = 1/m$. This special case can be used for illustration. The experiments use the general restriction: the full mathematics and code rosters have four eligible entry workers, and the heterogeneous roster has five.

B Full First-Passage Expression

For a fixed design write $\boldsymbol{P} = \boldsymbol{P}_S$. Let $\boldsymbol{\pi}$ be its stationary row vector, satisfying $\boldsymbol{\pi}\boldsymbol{P} = \boldsymbol{\pi}$ and $\boldsymbol{\pi}\mathbf{1} = 1$. Define $\Pi = \mathbf{1}\boldsymbol{\pi}$, $\Xi = \mathrm{dg}(\boldsymbol{\pi})$, and the Kemeny fundamental matrix $\boldsymbol{Z} = (\boldsymbol{I} - \boldsymbol{P} + \Pi)^{-1}$. The group inverse of $\boldsymbol{I} - \boldsymbol{P}$ is $\boldsymbol{Z} - \Pi$. With $[\cdot]_{\mathrm{dg}}$ zeroing off-diagonal entries, the full mean weighted first-passage matrix is

$$\boldsymbol{M}[\boldsymbol{W}] = \Big(\boldsymbol{Z}(\boldsymbol{P}\circ\boldsymbol{W})\Pi - \mathbf{1}\mathbf{1}^\top[\boldsymbol{Z}(\boldsymbol{P}\circ\boldsymbol{W})\Pi]_{\mathrm{dg}} + \boldsymbol{\pi}(\boldsymbol{P}\circ\boldsymbol{W})\mathbf{1}(\boldsymbol{I} - \boldsymbol{Z} + \mathbf{1}\mathbf{1}^\top[\boldsymbol{Z}]_{\mathrm{dg}})\Big)\Xi^{-1}.$$

Its diagonal $(r^\star, r^\star)$ entry is the first positive return in Equation (8), including the initial-call weight. The finite irreducible chain has this representation even when it is periodic. The implementation uses an additional source state with a zero-reward manager-to-source reset; its source-to-manager passage accumulates the same episode rewards.

The merged reporting state also permits a stationary characterization of quality. Write $\boldsymbol{\pi}_S$ for the stationary row distribution of $\boldsymbol{P}_S$. The frequency of correct reports is $p_S \sum_{j\in S}\boldsymbol{\pi}_S(j,1)$, and the frequency of all reports is $\boldsymbol{\pi}_S(r^\star)$. Hence

$$Q(S) = \frac{p_S \sum_{j\in S}\boldsymbol{\pi}_S(j,1)}{\boldsymbol{\pi}_S(r^\star)}. \tag{11}$$

C Proof of Proposition 1

Fix workers $\{1,2,3\}$ and reporting state $r^\star$. Their call costs are $c_1 = c_2 = 1$ and $c_3 = 10$ in both designs. On departure from $r^\star$, the new episode calls worker 1, which produces a correct answer. Workers 2 and 3 always preserve it. Worker 1 forwards to 3 with probability $\theta \in (0,1)$ and otherwise to 2, after which the receiving worker reports.

The reachable four-state chain $\{1,2,3,r^\star\}$ is irreducible for every such θ. Correctness is deterministic here, so no separate incorrect-answer states are needed. The same workers with the same behavior and call prices are present in both designs. The quality reward is one on the initial transition from $r^\star$ and zero thereafter, so $Q(\boldsymbol{P}_\theta) = 1$. Expected first-return cost is $C(\boldsymbol{P}_\theta) = c_1 + (1-\theta)c_2 + \theta c_3 = 2 + 9\theta$. Choosing $\theta = 1/4$ and $\theta = 3/4$ gives different expected costs, 4.25 and 8.75, while leaving every per-worker cost and the delivered quality unchanged. Only visit probabilities differ. The two designs present the same agents at the same per-call prices, so any criterion of those inputs alone assigns them one value, while their bills differ by a factor of $8.75/4.25 > 2$.

D PRUNING PROCEDURE

Algorithm 1 states the marginal-efficiency procedure used in the experiments. The bootstrap draws are shared across candidates, and only the fitted behavior rates are resampled; the estimated call-cost means remain fixed. The other removal rules replace the greedy choice while using the same screening and selection steps.

Algorithm 1 Quality-constrained greedy pruning

Require: Fitted objectives quality $\widehat{Q}$, cost $\widehat{C}$, full set $\mathbb{V}$, entry set $\mathcal{A}$, calibration episodes $\mathcal{D}$, target τ, and lower-tail probability α

$S \leftarrow \mathbb{V}$, $\widehat{\mathcal{P}} \leftarrow (S)$
while $|S| > 1$ **do**
  $R \leftarrow \{j \in S : (S \setminus \{j\}) \cap \mathcal{A} \neq \varnothing\}$
  Compute ΔQ_j and ΔC_j for $j \in R$
  $H \leftarrow \{j \in R : \Delta Q_j \leq 0\}$
  if $H \neq \varnothing$ **then**
    $j^\star \leftarrow$ first member of H in roster order
  else
    $j^\star \leftarrow \arg\max_{j \in R} \dfrac{\Delta C_j}{\max(\Delta Q_j, 10^{-6})}$, breaking ties by roster order
  end if
  $S \leftarrow S \setminus \{j^\star\}$; append S to $\widehat{\mathcal{P}}$
end while
Use shared episode resamples of $\mathcal{D}$ to compute $\underline{Q}_\alpha(S)$ for every $S \in \widehat{\mathcal{P}}$
$\mathcal{F} \leftarrow \{S \in \widehat{\mathcal{P}} : \underline{Q}_\alpha(S) \geq \tau\}$
if $\mathcal{F} \neq \varnothing$ **then**
  return $\arg\min_{S \in \mathcal{F}} \widehat{C}(S)$
else
  return $\mathbb{V}$ ▷ fallback without a passing bound
end if

E EXPERIMENTAL PROTOCOL

E.1 BENCHMARKS, SPLITS, AND GRADING

The mathematics calibration and held-out splits each contain equal numbers of MATH level-5 and AIME problems. The sampled AIME problems span competitions held from 1983 through 2023 (Mathematical Association of America, 2026). Both code systems use the same seeded sample of stdin-judged LiveCodeBench v6 problems dated January 4 to April 6, 2025 (Jain et al., 2025). The 60-problem code calibration split contains 16 easy, 13 medium, and 31 hard problems. The 40-problem code held-out split contains 9 easy, 9 medium, and 22 hard problems. Every comparison within a system uses the same problem split.

A code submission is correct when it passes every selected private test within a five-second timeout per test after textual output normalization. The local protocol selects up to 15 private tests deterministically for each problem. Three calibration problems have four available tests, and all other problems have 15. All code workers receive the problem statement and examples. The trace critic also receives execution feedback. For both entry and revision calls, duo uses up to three public tests to select between its two samples.

E.2 ROUTING, CALIBRATION, AND SCREENING

When more than one worker is retained, the active worker reports after each action with probability $1/2$; otherwise it forwards uniformly to a retained peer. A singleton reports immediately after its entry call. Experimental episodes force a report after five revisions. The analytic model uses the same local transition rule without a revision-depth state and consequently allows longer paths. Problem-pool sampling uses seed 0, calibration routing uses seeds 10 and 11, and held-out routing uses seed 77 through per-problem seeding.

For each candidate design, the lower screening percentile is obtained from 200 bootstrap replicates with seed 0, resampling complete calibration episodes and refitting the model. If no candidate on a removal sequence, including the full design, passes the screen, the procedure returns the full design as a fallback without a passing bound. The individual-accuracy comparator pools entry and revision outputs and applies the same Laplace smoothing as the fitted worker probabilities.

E.3 Agent rosters and generation settings

Table 5: Agent rosters. Entry workers may initiate an episode; revision-only workers act only after another worker.

System	Entry workers	Revision-only workers
Mathematics	CoT, duplicate, algebra, pro	verifier, step critic, saboteur
Code repair	standard, duplicate, edge, pro	trace critic, reviewer, saboteur
Heterogeneous	lite, mid, full, duo, pro	trace critic, saboteur

In the three core experiments, all inexpensive workers use the historical model alias `deepseek-v4-flash`; pro uses `deepseek-v4-pro`. The default temperature is .3. In the heterogeneous roster, lite requests reasoning disabled, mid requests low reasoning effort, and full uses the provider-default reasoning configuration. These names refer to the requested API settings. For both entry and revision calls, duo draws two samples at temperatures .2 and .8 and retains the sample passing more of up to three available public tests. Ties retain the first sample. Both calls contribute to cost. Code calls use a 64,000-token output cap, and mathematics calls use 28,000.

E.4 Prompt and revision protocol

Table 6 states the task information and requested output for each worker class. Role instructions vary across workers, and private-test outcomes remain hidden. Mathematics revisions receive the current extracted answer. Code revisions receive the current complete program. If parsing a revision fails, the previous answer or program remains the episode state and the active-worker index advances. An unparseable entry output is scored incorrect.

Table 6: Summary of prompt roles. Current output denotes the extracted answer or complete program produced earlier in the episode.

Worker class	Information supplied	Requested output
Mathematics entry	Problem statement	Solve the problem and give the final answer in the required boxed format.
Mathematics revision	Problem and current extracted answer	Check the solution, correct it when needed, and return a final boxed answer.
Code entry	Problem statement and examples	Produce a complete Python stdin/stdout program in one fenced code block.
Code reviewer	Problem and current program	Inspect the program and return a complete corrected program.
Trace critic	Reviewer input plus public-test execution feedback	Diagnose the observed failure and return a complete corrected program.
Saboteur	The corresponding revision input	Introduce one subtle boundary error and describe the change as an improvement.
Duo entry / revision	Corresponding entry or revision input, sampled twice	Return the sample with the higher public-test pass count; break ties in favor of the first.

E.5 Calibration support and truncation

Wrong-input revision support ranges from zero to two observations per mathematics worker and from one to three observations per code worker. Truncation flags are counted at the worker-event level;

Table 7: Calibration event counts. A repair is a revision whose input is incorrect. A code event is flagged as truncated when at least one constituent call has no response-cache entry; this is an inferred indicator rather than a verified finish reason. Event-level truncation indicators are unavailable for mathematics.

System	Events	Repairs	Truncated events	Truncated repairs
Mathematics	221	6	n/a	n/a
Code repair	221	10	14	4
Heterogeneous	221	15	20	4

duo requires two generations for one event. The harness does not cache responses that reach the token limit, but cache-write failures can also produce missing entries. An unparseable response with a cache entry does not receive this flag. These counts explain why the conditional repair estimates and their downstream screening percentiles are sensitive to a small number of observations.

E.6 Cost accounting

For the three core experiments, accounted cost uses the historical August 2026 evaluation constants. Flash input and output prices are $\$0.14$ and $\$0.28$ per million tokens. Pro input and output prices are $\$0.435$ and $\$0.87$. All input tokens are charged at the full input rate, and retained calls are priced as deployment calls. These constants apply to the DeepSeek experiments; the additional provider configurations require their own price schedules. Completion usage includes reasoning tokens. Local test execution and orchestration overhead are excluded.

For completed calls, cost is computed from recorded input and output token counts. When a code call has no response-cache entry, input tokens are approximated as prompt characters divided by 3.5 and output tokens are charged at the configured cap. When mathematics usage is unavailable, the calculation assigns 1,100 output tokens. These conventions define the reported dollar amounts for the three core experiments. Expected subset cost uses separate per-worker means for entry calls, revisions of incorrect inputs, and revisions of correct inputs. An overall per-worker mean is used when a conditional type is unobserved.

F Additional Experimental Results

F.1 Target-dependent selections

Table 8 supplements Table 3 with selections that change across targets. The mathematics random sequence has no candidate passing the $\tau = .90$ screen and therefore returns the full system. In heterogeneous code, the lite singleton has fitted quality $.645$ and a screening percentile of $.499$, so it is selected at $\tau = .40$ and excluded at the two higher targets.

Table 8: Additional target-dependent selections. Outcomes use 40 held-out problems; cost is accounted dollars per 100 problems.

System	τ	Rule	Design	Accuracy	Cost
Mathematics	.70, .80	random	CoT duplicate	.975	.137
	.90	random	full system	.900	.482
Heterogeneous	.40	direct cost	lite	.550	.031
	.50	direct cost	lite, saboteur	.600	.666
	.60	direct cost	lite, full, trace, saboteur	.700	1.139

F.2 Calibration-cost amortization

For a selected reduced design $\widehat{S}$, the sample-based break-even count is

$$T_{\mathrm{be}} = \frac{C_{\mathrm{cal}}}{C(V) - C(\widehat{S})},$$

where C_{cal} is the accounted cost of the calibration calls and the denominator uses observed held-out cost per problem. Table 9 reports this descriptive amortization calculation using realized held-out costs. Development and local-test overhead are excluded.

Table 9: Sample-based numbers of deployment problems required to offset calibration cost.

System	Rule	Selected design	Break-even problems
Mathematics	marginal efficiency	algebra	81
Code repair	marginal efficiency	pro	475
Code repair	random	duplicate	117
Heterogeneous	marginal efficiency	pro	779
Heterogeneous	individual accuracy	full reasoning	137

G Provider Ablation on Mathematics

We repeat the mathematics experiment with Qwen, Kimi, and GLM configurations. The problem identifiers and their order, seven functional roles, routing rule, grading procedure, calibration size, held-out split, and four removal rules are fixed across the three runs. Each run uses 120 calibration episodes over the same 60 problems and evaluates its frozen designs on the same 40 held-out problems. Candidate screening uses 200 bootstrap repetitions at targets .70, .80, and .90.

Qwen assigns `qwen3.5-flash-2026-02-23` to six roles and `qwen3.5-plus-2026-02-15` to the pro solver. Kimi uses `kimi-k2.6` for all roles, with thinking enabled for the pro solver. GLM uses `glm-5.1` for all roles, with high effort for six roles and xhigh effort for the pro solver. These configurations retain the same role definitions used in the main mathematics experiment.

Section 6 summarizes the cross-provider comparison. Table 10 adds the absolute held-out costs for the marginal-efficiency selection at $\tau = .80$.

Table 10: Provider ablation at $\tau = .80$. Accuracy is measured on the same 40 held-out problems. Costs are dollars per 100 problems.

Provider	Selected design	LCB	Full acc.	Selected acc.	Full cost	Selected cost	Saving
Qwen	pro	.956	.850	1.000	1.271	.828	34.8%
Kimi	algebra	.875	.950	.950	6.336	1.677	73.5%
GLM	algebra	.875	.725	.800	10.299	4.461	56.7%

We also compare the marginal-efficiency endpoint with the entry singleton with the lowest predicted cost among those passing the same calibration screen. Qwen and GLM reach the same endpoint under both rules. For Kimi, marginal efficiency selects algebra, with held-out accuracy .950 and cost $1.677, whereas direct singleton selection chooses the duplicate solver, with accuracy .900 and cost $2.255. On this held-out split, algebra has higher accuracy and lower cost than the singleton baseline. Since both endpoints report immediately, this comparison does not isolate a benefit from routing information.

Table 11: Marginal-efficiency and direct-singleton endpoints at $\tau = .80$. Each accuracy and cost entry is reported as held-out accuracy / dollars per 100 problems.

Provider	Marginal design	Singleton design	Marginal acc. / cost	Singleton acc. / cost
Qwen	pro	pro	1.000 / .828	1.000 / .828
Kimi	algebra	CoT duplicate	.950 / 1.677	.900 / 2.255
GLM	algebra	algebra	.800 / 4.461	.800 / 4.461

Removing the deliberately harmful reviser also gives a consistent control across configurations. Held-out accuracy rises by 10 percentage points for Qwen and 12.5 points for GLM, and remains unchanged for Kimi. Observed cost falls in every case.

Table 12: Removal of the saboteur. Accuracy and cost are measured on the same 40 held-out problems; costs are dollars per 100 problems.

Provider	Full acc.	Without sab. acc.	Accuracy change	Full cost	Without sab. cost
Qwen	.850	.950	+.100	1.271	1.258
Kimi	.950	.950	+.000	6.336	5.295
GLM	.725	.850	+.125	10.299	9.953

Figure 4 shows the fitted candidate frontiers and held-out deployments. The marginal-efficiency endpoint remains pro for Qwen and algebra for Kimi and GLM at targets .70 and .80. At $\tau = .90$, Qwen retains pro, whereas the Kimi and GLM marginal-efficiency paths return the full system; no reduced candidate on those paths passes the screen.

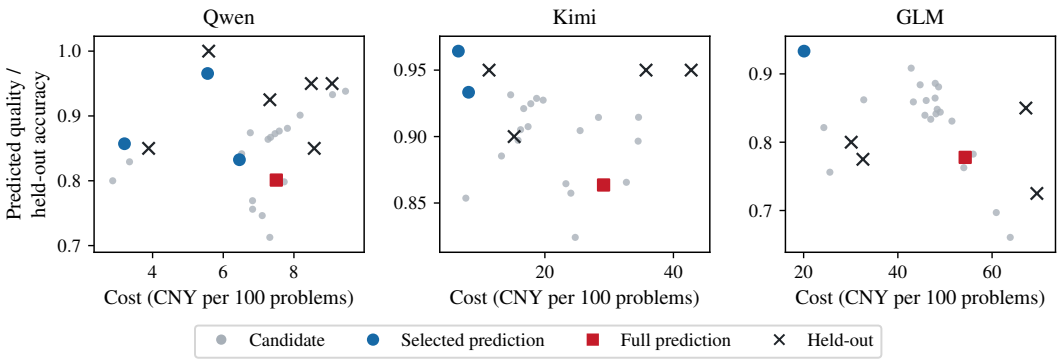


Figure 4: Predicted quality and cost for candidate designs under the Qwen, Kimi, and GLM mathematics configurations. Crosses show held-out outcomes for executed configurations. Costs are dollars per 100 problems.

H Exact Shapley Control

As a fitted-model attribution check, we evaluate all 2^7 coalitions and recompute exact quality Shapley values after each greedy deletion. A nonempty coalition with no original entry worker receives a uniform entry distribution over its members, and the empty coalition has value zero. Revision-only workers have no observed entry calls, so their initial-solve probability in these extended coalitions takes the Laplace-smoothed prior value .5. Exact refers to full enumeration within this fitted coalition game.

The Shapley values sum numerically to full-design fitted quality. As shown in Table 13, Shapley deletion first removes the deliberately harmful reviser in every system and reaches the same final worker as marginal efficiency. The calculation provides a second attribution rule within the fitted model.

Table 13: Endpoints of exact quality-Shapley deletion in the fitted coalition game.

System	First worker removed	Final retained worker
Mathematics	saboteur	algebra
Code repair	saboteur	pro
Heterogeneous	saboteur	pro