QUANTIZE THE TEAM: JOINT QUANTIZATION AND TOPOLOGY ADAPTATION FOR MULTI-AGENT LLM SYSTEMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-agent LLM systems solve tasks by exchanging information across repeated model calls. The large number of calls and long text sequences make efficient model deployment particularly important, with post-training quantization (PTQ) serving as a key approach. However, preserving the model's performance on individual calls may not preserve the benefits of communication: a message that helped another agent before quantization may no longer improve the final answer, even though it is still delivered. We call this loss a contraction of the system's effective topology: the communication graph stays the same, but fewer connections contribute to task performance. We characterize two failure mechanisms: sending agents provide less information beyond what other agents already supply, or receiving agents make less use of useful messages. To address these failures, we propose Quantization-Aware Repair or Reroute (Q-RoR) for agents that share a low-bit model. First, we introduce a unified quantization objective that jointly preserves message generation and use. By bounding the change in each message's edge value, the drop in task score when that message is removed, it grounds their optimization in what messages contribute and avoids manually balancing separate surrogate losses. Second, we propose an activation quantization operator tailored to multi-agent collaboration: it jointly controls quantization errors across roles to preserve complementary information. We pair this operator with attention calibration to restore receivers' use of useful messages. Third, compatibility-guided rerouting goes beyond quantization on a fixed topology or topology search with a fixed model: it jointly adapts the communication topology and quantized model to restore useful communication when repair alone is insufficient. Experiments across three models and five tasks, plus two further precisions on the main model, show that Q-RoR outperforms PTQ and sequential topology-search baselines.

1 INTRODUCTION

Multi-agent LLM systems have emerged as a promising approach to complex problem solving. By assigning agents different roles and allowing them to exchange intermediate results, these systems can divide a task into smaller steps, combine complementary perspectives, and refine solutions through collaboration, as exemplified by MetaGPT (Hong et al., 2024). This collaborative workflow comes at a substantial inference cost: a single task may require many model calls over long communication contexts. For example, OpenAI reports that one multi-agent mathematical research effort used roughly 130 billion output tokens,[1] while Anthropic reports that multi-agent research uses about 15 times more tokens than ordinary chat.[2] Efficient deployment is essential at this scale, making post-training quantization (PTQ) a promising way to reduce the cost of multi-agent systems.

Existing approaches address efficient multi-agent deployment from two largely separate directions. Conventional post-training quantization uses a calibration dataset of text sequences to optimize the quantized model, typically by preserving its outputs or intermediate activations during individual inference calls. This calibration focuses on model-level fidelity, without explicitly accounting for how agents generate and use messages within a workflow. Topology search, in turn, holds the underlying

[1] https://openai.com/index/navier-stokes-solution/

[2] https://www.anthropic.com/engineering/multi-agent-research-system

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.

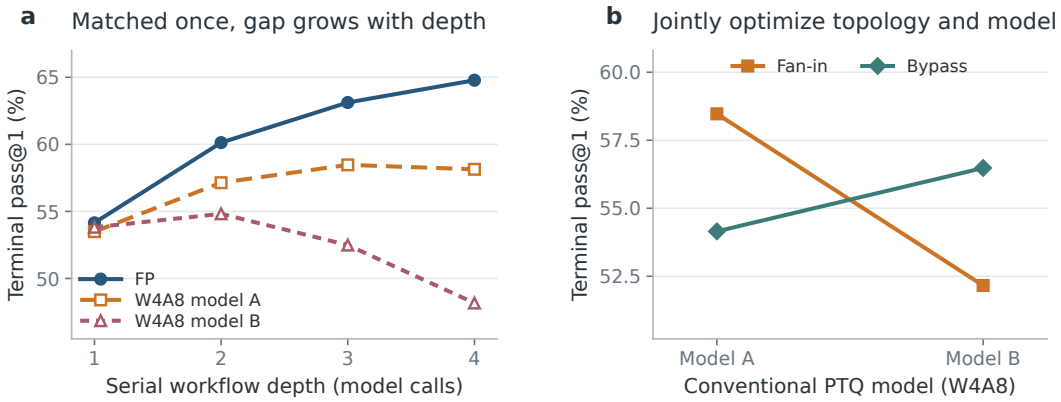

Figure 1: **Separate model and topology evaluation breaks down in multi-agent workflows.** Left: a full-precision (FP) model and two models with 4-bit weights and 8-bit activations (W4A8) differ by at most 0.7 point on one call, but separate increasingly as serial workflow depth grows. Right: two communication topologies, fan-in and bypass, exchange rank between the same two W4A8 models, each held fixed across topologies (setting in Appendix A).

model fixed and evaluates candidate communication structures by their final task performance. This separate optimization does not jointly account for two interacting effects: quantization can change which connections are useful, while topology changes the model's calibration contexts.

Figure 1 illustrates why separate optimization can be suboptimal. First, as shown in Figure 1a, two W4A8 models obtained through conventional PTQ have similar single-call performance but diverge increasingly as serial workflows become deeper, a difference that single-call evaluation misses. Second, Figure 1b shows that the ranking of two topologies reverses across quantization schemes, so choosing a topology without accounting for the quantization scheme can end up with the worse topology for the quantized model.

To understand these performance differences, we characterize two failure mechanisms of quantization calibrated solely to recover single-call performance. First, this quantization reduces the diversity of useful information across sending agents. Even when most of each agent's message and its standalone quality are preserved, the small but valuable portion that it uniquely contributes is lost. Additional agents then bring smaller task gains despite their continued inference cost. Second, receiving agents make less use of useful messages: the same message improves the final answer less after the receiver is quantized. Controlled sender and receiver tests distinguish these mechanisms. Quantization therefore has to preserve what messages contribute within the workflow: complementary information across senders and receivers' ability to use it.

We propose Quantization-Aware Repair or Reroute (Q-RoR), which quantizes the shared model and adapts the communication topology around each message's edge value: the drop in task score when that message is removed. First, calibration matches full-precision and quantized transcripts with and without each message. Since a transcript contains every message and the final answer, one objective covers message generation and use, and a bound ties it to each edge value. Layerwise, it weights reconstruction errors by how strongly they change message and answer predictions. Second, edges that still lose value are repaired on the side that fails, with weights and graph fixed. All sender roles read the same task text, so their activations there differ only slightly, and rounding errors of opposite sign can erase that difference. Our operator gives each role its own offset of the rounding grid, which keeps each role's expected error but changes how errors align across roles. We choose the offsets by how much each message adds to the team. Receiver attention calibration adjusts how much and how sharply each message is attended, restoring response to helpful messages without amplifying mismatched ones. Third, a message its receiver cannot use may still help another. For edges that still lose value, we move the message to a receiver that lacks its information and still uses helpful messages well. Our contributions are (i) a **quantization objective that bounds the change in each message's edge value**; (ii) to our knowledge, the first **activation quantization operator that controls how errors align across agents** sharing one model; and (iii) **joint quantization and topology adaptation**, which reroutes edges that still lose value after repair and judges each graph with its own recalibrated model.

2 Related Work

Post-training quantization. PTQ methods preserve model accuracy through different choices of weights, scales, clipping, and transformations. GPTQ (Frantar et al., 2022) uses approximate second-order information for weight quantization. AWQ (Lin et al., 2024) uses activation statistics to identify and rescale salient weight channels, while SmoothQuant (Xiao et al., 2023) transfers quantization difficulty from activations to weights through equivalent channel scaling. OmniQuant (Shao et al., 2024) learns weight clipping and equivalent transformations. QuaRot (Ashkboos et al., 2024) rotates hidden representations to reduce outliers, and SpinQuant (Liu et al., 2025) learns rotations using the quantized network's calibration loss. SERQ (Park et al., 2026) uses saliency-aware low-rank error reconstruction with static activation flattening and offline weight permutation for W4A8 and W4A4 quantization. These methods primarily target model fidelity within individual calls, without explicitly optimizing how communication contributes to a complete workflow. Our calibration matches both message-present and message-absent executions, giving message generation and use a common edge-value target.

Agentic workflow and topology optimization. Automated agent design optimizes programs, prompts, team composition, and communication patterns. ADAS (Hu et al., 2025) uses a meta agent to iteratively program and evaluate agentic systems. AFlow (Zhang et al., 2025) searches workflows represented as code using Monte Carlo Tree Search and execution feedback. GPTSwarm (Zhuge et al., 2024) optimizes node prompts and graph connectivity, while DyLAN (Liu et al., 2023) selects agent teams and supports query-dependent communication. ADAS and AFlow also study transfer across models. These approaches optimize workflows around a given model. We jointly adapt topology and quantization to sustain effective collaboration under low-bit deployment.

3 Problem Setup

We jointly choose a communication graph and a shared quantized model to maximize final task performance, guided by what each message contributes to that performance.

Agents and the quantized model. We are given a pretrained full-precision (FP) model with weights W, a deployment bit-width configuration β, and agent nodes V with role prompts $\{r_v\}_{v\in V}$. Each agent has a fixed role, receives the task input, and invokes the model once per task; a final agent s, called the sink, produces the task answer. All agents run one quantized model a. It consists of quantized weights shared by every agent, stored as integer codes with group scales and zero points; an activation-quantization policy that may differ across roles; and lightweight attention parameters for incoming messages. The last two serve the repairs in Section 5.2. We optimize all three parts at the prescribed bit width.

Communication graphs and allowed edits. Let $G = (V, E)$ be a directed acyclic graph whose edges carry agent outputs. Source nodes have no incoming messages. Every non-sink node sends its output to exactly one successor, so messages flow along a tree into s. We denote by $\mathcal{G}$ the graphs satisfying these constraints with the given nodes and sources. An allowed local edit either sends one node's message to a different successor or exchanges the successors of two nodes, keeping the graph in $\mathcal{G}$ and the roles fixed, so the number of model calls never changes.

Joint deployment objective. Let $\mathcal{D}$ be the deployment task distribution and $J(G, a)$ the expected task score of graph G executed with model a, averaged over tasks and generation randomness; $J(G, \mathrm{FP})$ denotes execution with W. Let $\mathcal{A}_\beta(G)$ contain the quantized models satisfying the sharing and precision constraints; it depends on G because the attention parameters are attached to each receiver's incoming messages. The ideal objective is

$$\max_{G\in\mathcal{G},\ a\in\mathcal{A}_\beta(G)} J(G, a). \tag{1}$$

We seek a solution that is local with respect to the allowed edits, under an offline search and calibration budget. Calibration tasks come from $\mathcal{D}_{\mathrm{cal}}$, which may differ from the deployment tasks; calibration, candidate-selection development, and final test sets are mutually disjoint.

Measuring a message's edge value. The task score shows how much performance quantization loses, but not which connections stop contributing, so we also measure each message. The *edge value* of e is the difference in task score with and without its message:

$$U_e^a(G) = J(G, a) - J(G - e, a). \tag{2}$$

The superscript names the executing model: FP or a quantized model a. A positive edge value means that removing the message lowers the score. To evaluate $G - e$, we replace only the message on

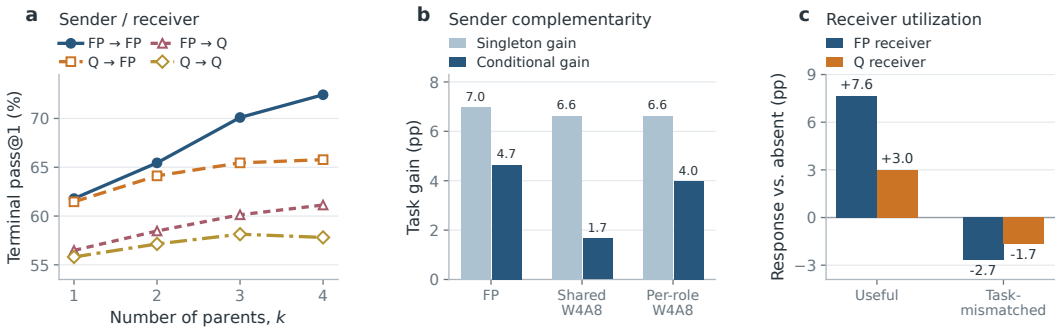

Figure 2: **Quantization reduces sender complementarity and receiver utilization.** (a) FP/Q parents crossed with FP/Q receivers in the star graph; legend arrows denote parents → receiver. (b) An added role's singleton and conditional task gains with an FP receiver. Shared W4A8 preserves singleton gain but reduces conditional task gain; per-role W4A8 stores one model per role. (c) Receiver response to fixed useful or task-mismatched messages, relative to no message.

edge e with `[no message]` and recompute downstream outputs. We keep the prompt template, node calls, and execution settings fixed. We evaluate the same edge in full precision and under quantization, using the same graph, roles, and generation protocol.

Effective topology and its contraction. The *effective topology* keeps only the edges that still contribute. We estimate each edge value from paired runs on the same tasks and count an edge as useful when its value reaches a threshold $\gamma_{\text{use}} > 0$, chosen before evaluation:

$$E^a_{\text{eff}}(G) = \{e \in E : U^a_e \geq \gamma_{\text{use}}\}. \tag{3}$$

Quantization contracts this topology when an edge is useful in full precision but loses its value under quantization. We call such an edge contracted when its quantized value falls below a smaller threshold $0 \leq \gamma_{\text{dead}} < \gamma_{\text{use}}$:

$$\mathcal{C}^a(G) = \{e \in E : U^{\text{FP}}_e \geq \gamma_{\text{use}},\ U^a_e < \gamma_{\text{dead}}\}. \tag{4}$$

An FP-useful edge whose quantized value lies between the two thresholds is weakened but not contracted. Section 4 examines why quantization contracts useful edges.

4 Motivating Diagnostics

To locate where contracted edges lose their value, we separate the messages senders produce from how receivers use them. All diagnostics use Qwen3-8B on 301 MBPP training and validation problems, disjoint from the test sets. In a star graph, $k \in \{1, \ldots, 4\}$ parent agents in specification-checking, algorithm-design, test-writing, and API-checking roles send messages to one solver, which writes the final program. Q denotes W4A8 model A of Figure 1, shared by all quantized agents. Figure 2a crosses FP and quantized senders with FP and quantized receivers. Each parent message is generated once per task and reused across receivers, so changing the receiver changes only how the same text is read. **With both sides quantized, four parents improve pass@1 by only 2.0 points over one parent, compared with 10.6 points in FP, and the fourth parent adds nothing.** The added agents still cost one call each, but their messages contribute little to the answer. Quantizing only the senders reduces the FP receiver's gain to 4.3 points, so the messages themselves carry less value. Quantizing only the receiver lowers the curve by 5.3–11.3 points, but this drop mixes weaker use of messages with weaker standalone solving. Figures 2b and 2c separate the two sides. Both measure an edge value as in equation 2, with the other side held in FP.

On the sender side, the lost value could come from weaker individual messages or from messages that no longer add to what the team already supplies. Figure 2b measures an added role's message with an FP receiver in two ways: its *singleton gain*, when the message is the receiver's only input, and its *conditional task gain*, when it joins a fixed FP team of two parents. **Shared W4A8 keeps the singleton gain within 0.4 point of FP, yet cuts the conditional task gain from 4.7 to 1.7 points.** Per-role calibration, which stores a separate W4A8 model for each role, raises the conditional task gain back to 4.0 points, so most of the loss comes from running all roles through one shared quantized model.

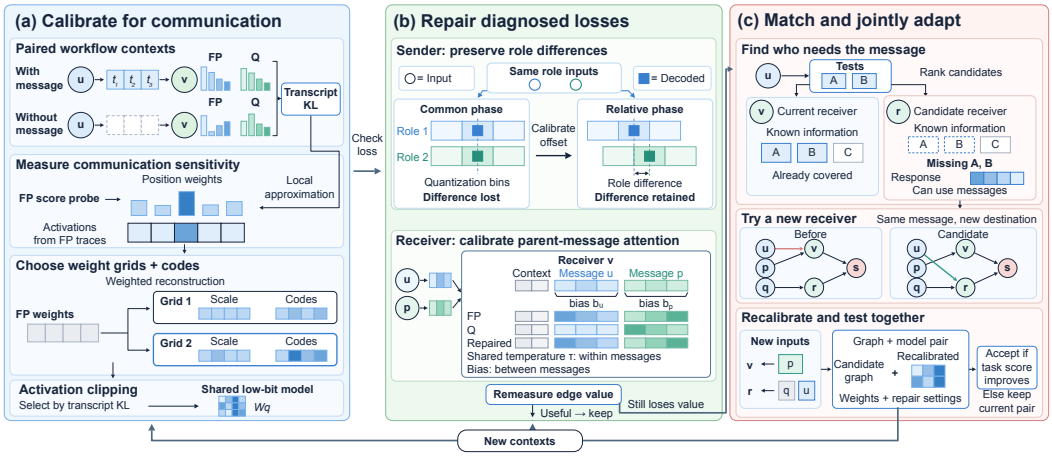

Figure 3: **Overview of Quantization-Aware Repair or Reroute.** (a) Calibrate the shared model to preserve message generation and use. (b) Diagnose residual losses and adjust sender quantization and receiver attention as needed. (c) For edges that still lose value, match messages to suitable receivers and evaluate each candidate graph together with the model adapted to its inputs.

On the receiver side, Figure 2c gives FP and quantized receivers the same useful or task-mismatched message and measures the change from each receiver's no-message baseline (Appendix A). Subtracting this baseline removes differences in standalone performance. **The same useful message improves the FP receiver by 7.6 points but the quantized receiver by only 3.0 points.** The penalty from the mismatched message also shrinks, from 2.7 to 1.7 points: the quantized receiver responds less to incoming messages in both directions.

These findings point each part of the method at a specific loss. An edge value is a difference between executions with and without a message, so calibration matches both executions (Section 5.1). Because the weights must remain shared, sender repair acts on what each role computes separately, coordinating activation quantization errors across roles. Receiver repair adjusts how much and how sharply each message is attended, while keeping the response to mismatched messages within its FP tolerance (Section 5.2). A message that remains unhelpful to its current receiver may still supply information another receiver lacks, so we reconsider its destination when repair is insufficient (Section 5.3).

5 Method: Quantization-Aware Repair or Reroute

We use each message's edge value to guide shared-model quantization and communication routing. Calibration first preserves message generation and use, and targeted sender and receiver adjustments recover residual losses on the current graph. Only edges that still lose value after repair enter receiver matching, where each candidate graph is evaluated with a model recalibrated for its new inputs.

5.1 Calibrate to Preserve Edge Values

We calibrate the shared model to preserve edge values (equation 2; Figure 3a). An edge value compares execution with and without a message, so matching the original workflow alone leaves behavior under message removal unconstrained. We therefore match both $G^{(0)} = G$ and each edge ablation $G^{(e)} = G - e$ to FP. With inter-agent messages M and final answer Y, our objective is

$$\mathcal{L}_{\text{init}}(G, a) = \mathbb{E}_{x\sim\mathcal{D}_{\text{cal}},\, \iota\sim\pi_G} D_{\text{KL}}\Big(P_W(M, Y \mid x, G^{(\iota)}) \,\Big\|\, P_a(M, Y \mid x, G^{(\iota)})\Big). \tag{5}$$

Here P_W and P_a are the FP and quantized joint distributions, and π_G assigns probabilities to the original and ablated executions.

The joint distribution covers message generation and use through successive conditional outputs. For a fixed task and intervention, either model factorizes as

$$P(M, Y) = \underbrace{\prod_{v\in V\setminus\{s\}} \prod_{t} p\big(m_{v,t} \mid r_v, M_{\text{pa}(v)}, m_{v,<t}\big)}_{\text{message generation}} \underbrace{\prod_{t} p\big(y_t \mid r_s, M_{\text{pa}(s)}, y_{<t}\big)}_{\text{response to the messages}}. \tag{6}$$

Here $m_{v,t}$ and y_t are message and answer tokens, r_v is the role prompt, $M_{\mathrm{pa}(v)}$ contains incoming messages, and $< t$ denotes the generated prefix. Each intermediate output reflects both incoming-message use and next-message generation; the final factor describes the sink's response. Let D_0 and D_e be the expected transcript KLs for G and $G - e$. For terminal utility in $[0, 1]$ and sampled execution, with KLs and edge values averaged over the same tasks,

$$|U_e^a - U_e^{\mathrm{FP}}| \le \sqrt{D_0/2} + \sqrt{D_e/2}. \tag{7}$$

Thus, matching both executions controls changes in edge value. The original term enters every edge's bound, and minimizing the largest bound weights it $|E|^{1/2}$ times as much as each ablation in π_G. Generation and response share one KL weighted by the bound, with no separate generation and use losses to balance. The KL chain rule gives a teacher-forced estimate: replay FP parent messages and prefixes, sum next-token KLs over calls and positions, and average over tasks and interventions.

To translate this target into layerwise quantization, we give more weight to reconstruction errors that affect message and answer predictions more strongly. A local second-order expansion of the joint KL gives a Fisher quadratic. Gradient probes on cached FP calls estimate a sensitivity $\omega_{\ell t}$ for each layer and input position. Dropping cross-layer and cross-position terms and approximating output-channel blocks by $\omega_{\ell t} I$ yields

$$\begin{aligned} &\min_{\boldsymbol{\alpha}, Z} \ \mathcal{E}_\ell\Big(\hat{W}_\ell(\boldsymbol{\alpha}, Z)\Big)\,, \\ &\mathcal{E}_\ell(\hat{W}_\ell) = \frac{\sum_t \omega_{\ell t} \|(\hat{W}_\ell - W_\ell) x_{\ell t}\|_2^2}{\sum_t \omega_{\ell t}} = \frac{1}{2}\,\mathrm{tr}\big(\Delta_\ell H_\ell \Delta_\ell^\top\big)\,. \end{aligned} \tag{8}$$

Here t indexes input positions across sampled calls, and $x_{\ell t}$ is the layer input, recollected after upstream projections are quantized. The clipping ratios $\boldsymbol{\alpha}$ determine the affine grids for each output row and weight group, and Z contains their integer codes. With $\Delta_\ell = \hat{W}_\ell - W_\ell$, the matrix form uses $H_\ell = 2\sum_t \omega_{\ell t} x_{\ell t} x_{\ell t}^\top / \sum_t \omega_{\ell t}$. Uniform sensitivities recover uniform layer reconstruction; the weighted metric covers senders and receivers in original and ablated contexts. The sensitivities are channel means of squared layer-output score gradients, from one backward pass per sampled FP call.

This objective selects each weight group's range and integer codes: for each candidate range, block-wise rounding with error compensation (Frantar et al., 2022) produces codes under H_ℓ from the same compensated weights, and we retain the grid and codes with the smallest conditional increase in the regularized weighted objective. With weights fixed, we choose a shared activation-clipping policy by the teacher-forced joint KL, evaluating their combined effect on communication. This yields the initial model a_0. Appendix B.1 gives the proofs, sensitivity estimator, and quantization protocol.

5.2 Recover Complementary Information and Its Use

The local approximation and finite grids can leave residual communication losses. We remeasure edge values under a_0 and diagnose the edges that are useful in FP but not under a_0, $E_{\mathrm{eff}}^{\mathrm{FP}}(G) \backslash E_{\mathrm{eff}}^{a_0}(G)$ in equation 3. Lost conditional task gain, the gain from adding the message to its receiver's other inputs, selects sender repair; reduced response to fixed helpful messages selects receiver repair. These adjustments retain the shared weights and graph. When both are needed, sender repair precedes receiver calibration on the resulting frozen messages.

Control quantization errors across roles. Preserving each role's standalone quality can still lose its additional contribution to a team (Figure 2b). Every agent reads the same task text, so on those tokens their activations differ only through the role prompts. We target such role-dependent activation differences: their distortion depends on the *difference* between roles' quantization errors. Opposite-signed errors can erase a small role difference even when their individual magnitudes are unchanged. Coordinating these errors can preserve role differences while keeping each role's expected squared error fixed. Our operator, *role-phased rounding*, shifts each role's quantization grid: an offset is added before rounding and removed after decoding. A shared random phase couples rounding across roles, and calibrated role offsets adjust their relative grids (Figure 3b, top).

For an activation group x_v of role v, let $\mathcal{Q}_{0,v}$ be the initialized activation quantizer. Set $E_v^0 = \|\mathcal{Q}_{0,v}(x_v) - x_v\|_2^2$ and $\widetilde{\Delta}_v = \sqrt{12 E_v^0 / d_g}$, where d_g is the group width. The shared phase $\xi \sim \mathcal{U}[-1/2, 1/2)$ and role phase ϕ_v give $c_v = \mathrm{wrap}(\xi + \phi_v)$, with wrap mapping to $[-1/2, 1/2)$. With zero point z_v, decoding yields

$$y_v = \widetilde{\Delta}_v \left[\mathrm{round}\left(x_v/\widetilde{\Delta}_v + z_v + c_v\right) - z_v - c_v\right], \tag{9}$$

where each scalar offset applies to the whole group. For $E_v^0 > 0$, no clipping, and exact offset removal, each coordinate's error is uniform on a step-width interval, giving $\mathbb{E}_\xi[\|y_v - x_v\|_2^2 \mid x_v] = E_v^0$. Changing the role offsets keeps these marginals and changes only the joint error pattern.

We screen phases by role-difference distortion after the next linear map, then select adjustments by conditional task gain while preserving each role's standalone quality within its calibration tolerance. Appendix B.2.1 specifies phase sharing, group alignment, selection, and implementation.

Restore receivers' use of helpful messages. Holding helpful messages fixed, receiver repair targets their reduced use (Figure 2c). We control how much attention each message receives and how concentrated it is within the message. A segment bias adjusts attention allocation, and a temperature controls concentration (Figure 3b, bottom). For the logit L_{ij} from query i to key j, we use

$$L'_{ij} = L_{ij}/\tau + b_{\mathrm{seg}(j)}. \tag{10}$$

FP attention statistics define a correction direction. We choose the smallest adjustment that restores the required useful-message response while keeping mismatched-message response within its FP tolerance. We retain the combined repair if its development score is no lower than that of a_0, and denote the retained model by a_R. Appendix B.2.2 gives the calibration rules.

5.3 Match Receivers and Jointly Adapt

After repair, we remeasure edge values under a_R and consider rerouting every FP-useful edge that still loses value, $U_e^{a_R} < U_e^{\mathrm{FP}}$. A message may help a different receiver whose inputs lack its information: test cases already available to a reviewer may still help a solver (Figure 3c, top). Diagnostics for a_R provide S_u, the calibration units where the message generated by sender u under a_R adds value to a fixed FP reader; N_r, the units not covered by receiver r's other inputs; and $h_r \in [0, 1]$, its retained fraction of FP useful-message response. We rank receivers by their compatibility

$$\kappa(u, r) = h_r \frac{|S_u \cap N_r|}{|S_u| + \varepsilon_c}. \tag{11}$$

Here $\varepsilon_c = 1$ stabilizes the denominator. The overlap measures what the message supplies that the receiver lacks; h_r weights its ability to use that information.

Rerouting changes the transcript distribution used for calibration. For each candidate graph G', we rerun Section 5.1 on the new graph and its edge ablations, then apply the diagnostic repair rules in Section 5.2. We evaluate the resulting pair $(G', a_{G'})$ on the same development tasks as the current pair and accept the best candidate if it scores higher (Figure 3c, bottom). Each graph is thus judged with a model adapted to its communication. Appendix B.3 gives candidate construction and evaluation; Appendix B.4 summarizes the procedure.

6 Experiments

We ask whether Q-RoR performs best across settings (Q1), whether its gains come from recovered communication (Q2), what each component contributes (Q3), whether graphs and quantized models must be adapted jointly (Q4), and how fast it runs (Q5).

6.1 Experimental Settings

We evaluate MBPP, HumanEval, GSM8K, MATH, and HotpotQA using a six-agent workflow with shared model weights. Qwen3-8B is the main model, with Llama-3.1-8B-Instruct and Mistral-Small-3.2-24B-Instruct for replication. W4A8 is the main precision; W8A8 and W4A4 test milder and stronger quantization. At each precision, baselines include GPTQ, OmniQuant, QuaRot, SERQ, and SERQ combined with GPTSwarm topology search before or after quantization. All methods share calibration and development tasks within each in-domain or cross-task setting and deploy a graph–model pair selected on development tasks. Appendix C details the data splits, workflow graphs, baselines, and calibration protocol.

6.2 Q1: Performance across Diverse Settings

We first vary the model, the task, and the calibration domain at W4A8. Table 1 compares methods on three models and five datasets, with column groups marking in-domain and cross-task calibration. The four PTQ baselines keep the original graph. The two sequential baselines add topology search to SERQ, which has the highest PTQ average on each model: Search → SERQ chooses the graph

Table 1: **Main W4A8 comparison across three models and five task datasets.** Subcolumns Q, L, and M are Qwen3-8B, Llama-3.1-8B-Instruct, and Mistral-Small-3.2-24B-Instruct. Rules separate PTQ on the original graph, topology search before quantization, and search after it.

	In-domain						Cross-task											
	MBPP			HumanEval			GSM8K			MATH			HotpotQA			Avg.		
Method	Q	L	M	Q	L	M	Q	L	M	Q	L	M	Q	L	M	Q	L	M
FP	68.4	61.8	71.9	84.1	70.7	87.2	91.7	85.2	92.6	61.4	38.6	58.0	51.2	48.9	56.4	71.4	61.1	73.2
GPTQ	64.6	56.2	69.8	79.9	64.6	83.5	89.0	82.4	91.5	56.2	33.4	54.8	47.3	44.7	54.0	67.4	56.3	70.7
OmniQuant	65.3	58.3	69.3	79.3	65.9	84.8	89.9	82.1	91.2	57.6	35.2	55.4	48.7	46.3	54.6	68.2	57.6	71.1
QuaRot	66.0	57.1	70.3	80.5	67.7	84.1	90.6	83.2	92.0	57.0	34.0	55.8	49.3	45.6	55.0	68.7	57.5	71.4
SERQ	66.5	59.3	70.7	81.7	67.1	85.4	90.4	83.6	91.9	58.8	35.8	56.4	49.5	46.8	55.3	69.4	58.5	71.9
Search → SERQ	65.8	59.7	71.2	82.3	67.7	86.0	90.2	83.9	91.7	58.4	35.4	56.8	49.1	47.3	55.5	69.2	58.8	72.2
SERQ → search	67.0	60.2	70.5	81.1	68.3	85.4	90.9	83.5	92.1	59.2	36.6	56.6	49.8	46.9	55.8	69.6	59.1	72.1
Q-RoR (ours)	**67.7**	**60.9**	**71.4**	**82.9**	**69.5**	**86.6**	**91.1**	**84.4**	**92.3**	**60.4**	**37.4**	**57.2**	**50.6**	**48.1**	**56.1**	**70.5**	**60.1**	**72.7**

in FP and then quantizes, and SERQ → search chooses it with the frozen SERQ model. Both orders use GPTSwarm edge optimization over the nine single-edit graphs.

Q-RoR scores highest on every model–dataset combination at W4A8, in both in-domain and cross-task settings. Over SERQ, it raises the five-task average by 1.1, 1.6, and 0.8 points on Qwen3-8B, Llama-3.1-8B, and Mistral-Small-3.2-24B, respectively, and it also exceeds both Search → SERQ and SERQ → search on every model and dataset. It stays within 1.0 point of the FP average on all three models, and its gains hold on the three cross-task datasets, whose calibration and development tasks come only from DROP.

At the milder W8A8 and the stronger W4A4, Q-RoR again performs best on every dataset (Appendix C.4). On Qwen3-8B, its margin over SERQ grows with quantization strength, from 0.4 points at W8A8 to 1.1 at W4A8 and 1.3 at W4A4.

6.3 Q2: Recovery of Useful Communication

We next ask whether this accuracy comes from recovered communication, starting with edge values (equation 2) on the original graph. **PTQ leaves each edge at most 40% of its FP value, whereas Q-RoR brings every edge back to at least three quarters of it** (Figure 4a).

On edge $0 \to 3$, we change only the planner's message while the FP solver and the requirements checker's message stay fixed. Quantization roughly halves what the planner adds to this team, and calibrated role offsets recover most of the loss (Figure 4b).

Finally, on edge $2 \to 4$, the reviewer sees identical helpful, task-mismatched, and absent verifier messages. Quantization nearly halves the gain from the helpful message; correcting attention temperature and segment bias restores almost all of it, while the response to the mismatched message stays within the FP level (Figure 4c).

6.4 Q3: Contribution of the Design Components

We remove each component of Q-RoR on Qwen3-8B at W4A8 and separate the two choices in its calibration; Appendix C.5 gives both tables and their protocol. **Removing any component lowers the five-task average, and removing communication-aware initialization lowers it most, by 1.31 points**; removing rerouting or candidate recalibration costs 0.69 and 0.42.

The two repairs act on different tasks. Removing sender repair costs 1.0 point on MBPP, matching the planner's conditional gain it restores in Figure 4b, but does not lower HotpotQA; removing receiver repair costs 0.9 on HotpotQA.

In calibration, matching edge-ablated executions mainly reduces edge-value drift and helps cross-task HotpotQA, whereas sensitivity weighting mainly helps MBPP. Combining the two cuts the drift from 2.1 to 1.2 points and gives the best score on both tasks.

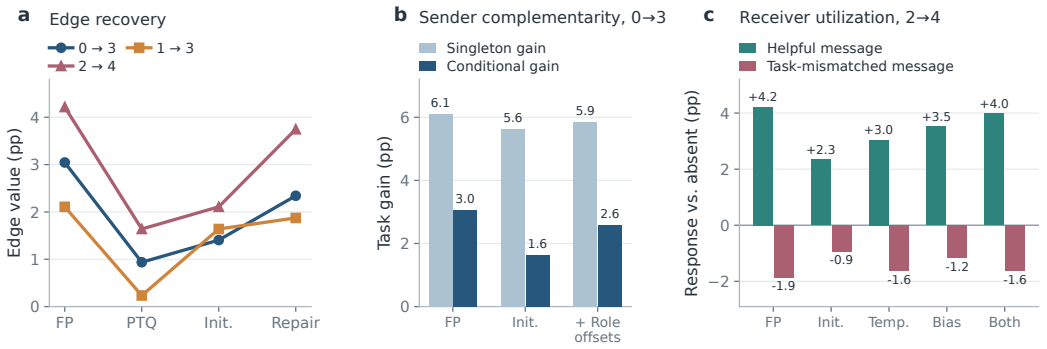

Figure 4: **Communication diagnostics.** Qwen3-8B at W4A8 on MBPP, original six-agent graph; Init. is the initialized model before repair. (a) Three original-graph edges across FP, PTQ, initialization, and repair. (b) Singleton and conditional task gains of the planner's message to an FP solver. (c) Reviewer response to the verifier's message, relative to the same reviewer's no-message baseline.

6.5 Q4: Joint Adaptation of Graphs and Quantized Models

The best graph changes with the graph used for calibration. Candidate G_1 swaps the receivers of the planner and the verifier, so the verifier's checks reach the solver. Judged with the model calibrated on the original graph G_0, G_1 looks worse than G_0 (66.0 vs. 66.3); recalibrated on G_1, it becomes the best at 66.7 (Figure 5a). The two sequential baselines of Table 1 evaluate all nine graphs, yet neither deploys G_1: Search $\rightarrow$ SERQ picks the reassignment $1 \rightarrow 4$ in FP, and SERQ $\rightarrow$ search picks G_2, which moves only the verifier's message to the solver. Recalibrating each candidate adds 0.5 MBPP point over rerouting with a fixed quantized model, and the complete pair exceeds SERQ with per-graph recalibration by 0.4–1.0 points on every dataset (Appendix C.6). Receiver matching decides which candidates are recalibrated first: its first candidate already yields half of the final gain, twice that of random or structural ranking, and its first three, our budget, reach the full gain (Figure 5b).

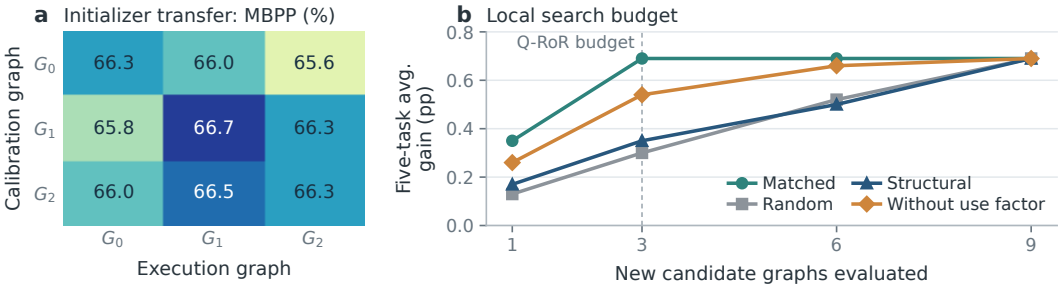

Figure 5: **Joint-adaptation analysis.** (a) Initialization-only MBPP accuracy per calibration and execution graph. (b) Five-task average gain over repair on G_0 vs. candidate budget.

6.6 Q5: Inference Runtime

On an NVIDIA RTX A6000, Q-RoR runs the six-call Qwen3-8B W4A8 workflow within 3% of GPTQ and 1.05× and 1.04× faster than SERQ on MBPP and MATH, because its repairs add only per-group and attention-logit operations (Appendix C.7).

7 Conclusion

Quantization can reduce the value of multi-agent communication even when messages are still delivered. By letting each message's edge value guide calibration, repair, and rerouting, Q-RoR consistently outperforms PTQ and sequential topology-search baselines. The experiments also trace these gains to the intended mechanisms: edges regain most of their FP value, role offsets recover sender complementarity, and attention calibration restores receivers' use of helpful messages. Every component contributes, and the best graph changes once each candidate is judged with its own recalibrated model. Multi-agent systems should thus be quantized as a team.

Ethics Statement

This work improves the efficiency of multi-agent LLM systems and uses only publicly available models and benchmarks. It involves no human subjects or personal data.

Reproducibility Statement

Code is available at `https://anonymous.4open.science/r/Q-RoR-0221`. Appendix B gives the derivations, implementation details, and complete procedure of Q-RoR, and Appendix C gives the tasks, data splits, models, workflow graphs, baselines, and calibration protocol.

AI Use Statement

We used generative AI tools to polish the writing and to help write code. We have reviewed all AI-assisted content and take full responsibility for the final content of this work.

References

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L. Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. Quarot: Outlier-free 4-bit inference in rotated llms. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 100213–100240. Curran Associates, Inc., 2024. doi: 10.52202/079017-3180. URL `https://proceedings.neurips.cc/paper_files/paper/2024/file/b5b939436789f76f08b9d0da5e81af7c-Paper-Conference.pdf`.

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`.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021. URL `https://arxiv.org/abs/2107.03374`.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL `https://arxiv.org/abs/2110.14168`.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2368–2378, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1246. URL `https://aclanthology.org/N19-1246/`.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

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. *NeurIPS*, 2021.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, zili wang, Steven Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. Metagpt: Meta programming for a multi-agent collaborative framework. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 23247–23275, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/6507b115562bb0a305f1958ccc87355a-Paper-Conference.pdf.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 21344–21377, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/36b7acf6f6010652b3f2a433774a66fe-Paper-Conference.pdf.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. In P. Gibbons, G. Pekhimenko, and C. De Sa (eds.), *Proceedings of Machine Learning and Systems*, volume 6, pp. 87–100, 2024. URL https://proceedings.mlsys.org/paper_files/paper/2024/file/42a452cbafa9dd64e9ba4aa95cc1ef21-Paper-Conference.pdf.

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. Spinquant: Llm quantization with learned rotations. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 92009–92032, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/e5b1c0d4866f72393c522c8a00eed4eb-Paper-Conference.pdf.

Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. Dynamic llm-agent network: An llm-agent collaboration framework with agent team optimization, 2023.

Yeonsik Park, Hyeonseong Kim, and Seungkyu Choi. Serq: Saliency-aware low-rank error reconstruction for llm quantization. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 35889–35909, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/3ca380c5fe9f174a71a230478741169f-Paper-Conference.pdf.

Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Gao Peng, Yu Qiao, and Ping Luo. Omniquant: Omnidirectionally calibrated quantization for large language models. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 45472–45496, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/c6483c8a68083af3383f91ee0dc6db95-Paper-Conference.pdf.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. SmoothQuant: Accurate and efficient post-training quantization for large language models. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 38087–38099. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/xiao23c.html.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1259. URL https://aclanthology.org/D18-1259/.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, XiongHui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. Aflow: Automating agentic workflow generation. In Y. Yue, A. Garg, N. Peng, F. Sha, and

R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 34040–34077, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/5492ecbce4439401798dcd2c90be94cd-Paper-Conference.pdf.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. GPTSwarm: Language agents as optimizable graphs. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 62743–62767. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/zhuge24a.html.

A Diagnostic Protocol Details

This appendix specifies the setting of Figures 1 and 2 and the message conditions behind the receiver diagnostic of Section 4.

Setting of Figures 1 and 2. Both figures use Qwen3-8B on the 301 MBPP training and validation problems outside the sanitized release (Appendix C), so no test problem enters them and every score is a multiple of $1/301$. Models A and B are W4A8 models from conventional single-call PTQ, calibrated on 128 generic text sequences with the activation quantizer of Appendix B.1: model A rounds weights with GPTQ (Frantar et al., 2022) and model B rounds them to nearest. In Figure 1a, a chain of d calls solves each problem: the first call writes a program, each later call revises its predecessor's program, and the last program is scored. A single call is the solver of Figure 2 without incoming messages. Figure 1b runs two graphs over a specification checker u_1, a test writer u_2, a solver v, and a final solver s: fan-in sends both checks to the solver ($u_1 \to v$, $u_2 \to v$, $v \to s$), and bypass sends the specification check to the final solver instead ($u_1 \to s$, $u_2 \to v$, $v \to s$). In Figure 2, Q is model A, and panel (b) adds the test writer to the specification-checking and algorithm-design parents.

Two message conditions against one baseline. Figure 2c places a receiver under a useful message, under a task-mismatched message, and under no incoming message. The useful message is the full-precision specification-checking message for the task the receiver is solving, the single parent of the $k = 1$ star, held fixed across the full-precision and quantized receivers so that both read identical text. The no-message condition removes the incoming message and leaves the rest of the prompt unchanged, and each receiver is scored against its own no-message baseline.

Constructing the task-mismatched message. The task-mismatched message is a message the same sender role produced on a different task instance, truncated to the same length as the useful message it replaces. The two conditions thus share the message's origin, role, and length, and differ only in whether its content is relevant to the task at hand. The same construction supplies the task-mismatched-message control that constrains receiver repair in Appendix B.2.2.

B Method Derivations and Implementation Details

Appendices B.1–B.3 follow Sections 5.1–5.3 in order, giving the proofs, calibration rules, and implementation of each step. Appendix B.4 lists the complete procedure.

B.1 Calibrate to Preserve Edge Values

Transcript objective and reference data. Let $\zeta = (M, Y)$, $\mathcal{I}(G) = \{0\} \cup E$, and $G^{(0)} = G$, $G^{(e)} = G - e$ as in Section 5.1. For a fixed task distribution μ, define

$$\begin{aligned} P_W^G(\iota, \zeta \mid x) &= \pi_G(\iota) P_W(\zeta \mid x, G^{(\iota)}), \\ P_a^G(\iota, \zeta \mid x) &= \pi_G(\iota) P_a(\zeta \mid x, G^{(\iota)}). \end{aligned} \tag{12}$$

and

$$\mathcal{L}_{\text{init}}^{\mu}(G, a) = \mathbb{E}_{x \sim \mu} D_{\text{KL}}\big(P_W^G(\iota, \zeta \mid x) \,\big\|\, P_a^G(\iota, \zeta \mid x)\big), \qquad a \in \mathcal{A}_\beta(G). \tag{13}$$

The shared schedule cancels in the likelihood ratio, recovering equation 5 at $\mu = \mathcal{D}_{\text{cal}}$. Each calibration task supplies an original FP workflow and every edge ablation, $N_{\text{cal}}(|E| + 1)$ workflows in total. An ablation shares every call outside the removed edge's downstream region with the original workflow, so collection needs at most $N_{\text{cal}}(|E| + 1)|V|$ calls.

Conditional and marginal distribution control. For node v and generated-token position t, the history $\mathcal{H}_{v,t}^{\iota}$ contains the task, role prompt, parent messages, and generated prefix. Let Y_v be call v's output, a message for non-sink nodes and Y for the sink. Define the teacher-forced conditional

mismatch and the marginal mismatch under each model's own upstream messages:

$$C_v^\iota = \mathbb{E}_{\substack{x\sim\mu,\\ \zeta\sim P_W(\cdot|x,G^{(\iota)})}} \sum_t D_{\mathrm{KL}}\big(p_W(\cdot \mid \mathcal{H}_{v,t}^\iota)\|p_a(\cdot \mid \mathcal{H}_{v,t}^\iota)\big)\,,$$
$$M_v^\iota = \mathbb{E}_{x\sim\mu} D_{\mathrm{KL}}\Big(P_W(Y_v \mid x, G^{(\iota)})\|P_a(Y_v \mid x, G^{(\iota)})\Big)\,. \tag{14}$$

Proposition 1 (Simultaneous distributional control) *Suppose FP and quantized executions share a finite causal factorization, prompt construction, and stopping convention, with $\pi_G(\iota) > 0$ for every intervention. Then*

$$\mathcal{L}_{\mathrm{init}}^\mu(G,a) = \sum_\iota \pi_G(\iota) \sum_{v\in V} C_v^\iota, \qquad \max\{C_v^\iota, M_v^\iota\} \le \frac{\mathcal{L}_{\mathrm{init}}^\mu(G,a)}{\pi_G(\iota)}. \tag{15}$$

Proof. Order calls topologically and pad stopped outputs with a termination symbol. The factorization in equation 6 makes the transcript log-likelihood ratio a sum of token log-likelihood ratios. Taking FP conditional expectations gives

$$\mathcal{L}_{\mathrm{init}}^\mu(G,a) = \mathbb{E}_{\substack{x\sim\mu,\\ (\iota,\zeta)\sim P_W^G(\cdot|x)}} \sum_{v\in V}\sum_t D_{\mathrm{KL}}\big(p_W(\cdot \mid \mathcal{H}_{v,t}^\iota) \,\big\|\, p_a(\cdot \mid \mathcal{H}_{v,t}^\iota)\big)\,. \tag{16}$$

Write $D_\iota^\mu = \sum_v C_v^\iota$. Nonnegativity gives $C_v^\iota \le D_\iota^\mu \le \mathcal{L}_{\mathrm{init}}^\mu/\pi_G(\iota)$. Marginalization to Y_v gives $M_v^\iota \le D_\iota^\mu$. □

Replaying cached FP parent messages and prefixes estimates this chain-rule sum.

Edge-value bound. For the same task distribution μ, write

$$D_\iota^\mu(G,a) = \mathbb{E}_{x\sim\mu} D_{\mathrm{KL}}\Big(P_W(\zeta \mid x, G^{(\iota)}) \,\Big\|\, P_a(\zeta \mid x, G^{(\iota)})\Big)\,. \tag{17}$$

Let $J_\mu(G',a)$ be the expected task score of graph G' under μ, for a quantized model a or $a = \mathrm{FP}$, and $U_{e,\mu}^a = J_\mu(G,a) - J_\mu(G-e,a)$.

Proposition 2 (Edge-value preservation) *For measurable terminal utility $u(x,\zeta) \in [0,1]$, with the same task distribution μ and temperature-one sampling in FP and quantized execution,*

$$\big|U_{e,\mu}^a(G) - U_{e,\mu}^{\mathrm{FP}}(G)\big| \le \sqrt{\frac{D_0^\mu(G,a)}{2}} + \sqrt{\frac{D_e^\mu(G,a)}{2}}. \tag{18}$$

Proof. For fixed x, ι, let $d_\iota(x)$ be the transcript KL and P_W^ι, P_a^ι the corresponding distributions. Bounded utility and Pinsker give

$$|\mathbb{E}_{P_a^\iota} u - \mathbb{E}_{P_W^\iota} u| \le \mathrm{TV}(P_W^\iota, P_a^\iota) \le \sqrt{d_\iota(x)/2}.$$

Averaging over tasks and applying Jensen yields $|J_\mu(G^{(\iota)},a) - J_\mu(G^{(\iota)},\mathrm{FP})| \le \sqrt{D_\iota^\mu/2}$. The triangle inequality for the original and ablated scores proves the result. □

An FP edge value exceeding this bound remains positive after quantization. Both propositions hold for any fixed μ: $\mathcal{D}_{\mathrm{cal}}$ gives the calibration bounds and $\mathcal{D}$ the deployment counterparts. The bound concerns sampled execution, which the FP calibration references follow. It does not cover the greedy decoding used for reported scores, and it is loose for long workflows because the transcript KL sums over every token. It therefore guides what calibration matches, and Section 6 measures greedy edge values directly.

Intervention schedule. The original discrepancy D_0 enters every edge bound. By Proposition 1, $\pi_G(0)D_0^\mu + \pi_G(e)D_e^\mu \le \mathcal{L}_{\mathrm{init}}^\mu$, so Cauchy–Schwarz bounds the right side of equation 18 by $\sqrt{\mathcal{L}_{\mathrm{init}}^\mu\,(1/\pi_G(0) + 1/\pi_G(e))/2}$ for every edge. With the ablations sharing the remaining mass equally, this worst-case bound is smallest at

$$\pi_G(0) = \frac{1}{1+\sqrt{|E|}}, \qquad \pi_G(e) = \frac{\pi_G(0)}{\sqrt{|E|}},$$

where it equals $(1+\sqrt{|E|})\sqrt{\mathcal{L}_{\mathrm{init}}^\mu/2}$. The original execution thus weighs $\sqrt{|E|}$ times as much as each ablation. Because every term is a conditional KL of the same workflow distribution, message generation and use need no separately tuned loss coefficients.

Local quadratic and activation probes. With FP activations, let θ vectorize a layer's weights. Local regularity and common support give

$$D_{\mathrm{KL}}\big(P_\theta^G \| P_{\theta+\delta}^G\big) = \frac{1}{2}\delta^\top F_{\mathrm{comm}}(G)\delta + o(\|\delta\|_2^2). \tag{19}$$

We estimate this metric through activation sensitivities. From FP reference workflows sampled at temperature one, we draw 128 calls with replacement: a task uniformly, an intervention from π_G, and a call uniformly within that workflow. These samples estimate the expectation over P_W in the objective; task scores are evaluated with greedy decoding (Appendix C).

For a sampled call, let $\mathcal{T}$ contain its generated-token prediction positions, $\mathcal{H}_t$ the cached history at position t, and $z_{\ell i} \in \mathbb{R}^{m_\ell}$ the output of linear layer ℓ at input position i. At the fixed prefixes, we sample independent auxiliary labels

$$\widetilde{y}_t^{(r)} \sim p_W(\cdot \mid \mathcal{H}_t), \qquad \psi^{(r)} = \sum_{t\in\mathcal{T}} \log p_W(\widetilde{y}_t^{(r)} \mid \mathcal{H}_t), \qquad \nabla_{\ell i}^{(r)} = \frac{\partial \psi^{(r)}}{\partial z_{\ell i}}. \tag{20}$$

The labels do not alter the cached histories $\mathcal{H}$. Conditioned on them, the contribution $\nabla_{\ell i,t}$ of each token t has mean zero, so independent labels give

$$\mathbb{E}[\nabla_{\ell i,t}\nabla_{\ell i,t'}^\top \mid \mathcal{H}] = 0 \quad (t \neq t'), \qquad \mathbb{E}[\nabla_{\ell i}\nabla_{\ell i}^\top \mid \mathcal{H}] = \sum_t \mathbb{E}[\nabla_{\ell i,t}\nabla_{\ell i,t}^\top \mid \mathcal{H}].$$

One backward pass through the sum therefore collects all layer-output gradients, including those at prompt and parent-message positions. We cache only

$$\omega_{\ell i} = \frac{1}{R m_\ell} \sum_{r=1}^{R} \|\nabla_{\ell i}^{(r)}\|_2^2, \tag{21}$$

with $R = 1$. Sampled labels are needed because their score has nonzero variance at FP agreement, whereas the gradient of the FP-versus-itself KL is zero.

Weighted reconstruction. Discarding cross-layer and cross-position Fisher terms and approximating each output-channel block $\mathbb{E}[\nabla_{\ell i}\nabla_{\ell i}^\top]$ by $\omega_{\ell i} I_{m_\ell}$ gives equation 8, whose index t runs over the input positions of all sampled calls. The cross-token score terms vanish in expectation, whereas the discarded cross-position terms are an approximation. The matrix H_ℓ keeps input-channel correlations and is shared across output rows, and constant sensitivities recover uniform reconstruction. The edge-value bound applies to the transcript KL, which is evaluated separately from this local approximation.

Weight grids and compensated rounding. Transformer blocks are processed in execution order, and their projections in dependency order: Q/K/V, attention output, MLP gate/up, then MLP down. Each projection's inputs are recollected after upstream projections are quantized, with FP sensitivities kept at the same cached positions. Each output row and contiguous group of 128 input columns has four candidate affine grids at the prescribed bit width. Their ranges scale the group's FP range $[\min(0, \min w), \max(0, \max w)]$ by $\alpha \in \{1, 0.95, 0.90, 0.85\}$. With n_ℓ input channels and damping $\lambda = 0.01\,\mathrm{tr}(H_\ell)/n_\ell$, the regularized objective is $\mathcal{E}_\ell(\hat{W}) + \frac{\lambda}{2}\|\hat{W} - W_\ell\|_F^2$. Within each group, every candidate is rounded column by column from the same compensated weights with GPTQ updates (Frantar et al., 2022). Its score is the increase in the regularized objective with earlier groups fixed and later columns optimally compensated as continuous variables. Each row commits its lowest-score grid and codes, with ties favoring the larger α, and the committed rounding errors are compensated in later columns. Activation quantization is disabled during probing and weight rounding.

Activation policy. Attention and MLP linear layers quantize their weights and inputs; embeddings, the vocabulary head, normalization, softmax, and biases keep floating-point precision. With weights fixed, activations use dynamic affine quantization on each token's contiguous 128-channel groups, with range $[\alpha \min(0, \min x), \alpha \max(0, \max x)]$ for group x. A single $\alpha \in \{1, 0.95, 0.90, 0.85\}$, shared across roles and layers, is chosen by teacher-forced transcript KL on the same sampled calls with all activation quantizers enabled; ties favor the larger α. The weight codes, group scales, zero points, clipping ratios, and activation policy form a_0.

B.2 Recover Complementary Information and Its Use

Weights and graph stay fixed. Edge values are estimated on the calibration tasks from paired runs, in points of task score, with thresholds $\gamma_{\text{use}} = 2.5$ in equation 3 and $\gamma_{\text{dead}} = 1$ in equation 4. For each edge useful in FP but not under a_0, we compare two quantities with their FP references: the conditional task gain of the sender's message, measured with a fixed FP reader and the other parents' messages, and the receiver's response to the fixed helpful message. A loss in the first selects sender repair, and a loss in the second selects receiver repair; either repair may leave the model unchanged. When both are selected, the sender is calibrated first and its resulting messages are frozen for receiver calibration. As in Section 5.2, the repaired model becomes a_R only if its development score is no lower than that of a_0.

B.2.1 Control Quantization Errors across Roles

Step size and enablement. Each activation group is indexed by role v, layer ℓ, token t, and channel group g of width d_g; Section 5.2 drops these indices. The initialized quantizer $\mathcal{Q}_{0,v\ell g}$, including its clipping, scale, zero point, and integer range, gives the baseline error $E^0_{v\ell tg}$ and the step $\widetilde{\Delta}_{v\ell tg}$ as in Section 5.2. The operator is enabled for a group only if every allowed phase fits within the integer range without clipping; otherwise the group keeps $\mathcal{Q}_{0,v\ell g}$.

Shared phases. Activation groups are aligned across roles by workflow event, layer, structural sequence segment, local token position, and channel group; groups without a counterpart in another role keep the baseline quantizer. A stateless hash of the task, layer, segment, local token position, and group gives the shared phase $\xi_{\ell tg} \sim \mathcal{U}[-1/2, 1/2)$, so every role derives the same value from the same key. Each role stores one offset $\phi_{v\ell g} \in [0, 1)$ per layer and group, giving $c_{v\ell tg} = \text{wrap}(\xi_{\ell tg} + \phi_{v\ell g})$ in equation 9; the zero point follows the baseline rule at the new step.

Error budget and phase selection. Without clipping, removing the offset after decoding makes each coordinate's error uniform on $[-\widetilde{\Delta}/2, \widetilde{\Delta}/2)$, so the expected squared error is $d_g\widetilde{\Delta}^2/12 = E^0$. A fixed role offset keeps c uniform and hence keeps this expected error. Relative offsets change only how errors align across roles, which sets the error in a role difference, $(y_v - y_{v'}) - (x_v - x_{v'}) = e_v - e_{v'}$ with $e_v = y_v - x_v$. For each aligned tuple of n_{role} roles, let $A_{\ell g}$ be the Frobenius-normalized column block of the next quantized linear map, fixed during selection. Offsets are screened by

$$\mathcal{L}_{\text{diff}}(\phi) = \mathbb{E}_{(x_1,\ldots,x_{n_{\text{role}}})\sim\mathcal{D}_{\text{cal}},\xi} \frac{2}{n_{\text{role}}(n_{\text{role}}-1)} \sum_{v<v'} \left\| A_{\ell g}(e_{v\ell tg} - e_{v'\ell tg}) \right\|_2^2. \tag{22}$$

One reference role keeps offset zero. Starting from zero, a single coordinate sweep in a fixed role order searches $\{0, 1/4, 1/2, 3/4\}$ for each other role, and groups are proposed in order of their improvement in $\mathcal{L}_{\text{diff}}$. Among these proposals, we maximize the conditional task gain of the message to a fixed FP reader's other inputs, subject to every role's singleton utility staying within $\epsilon_{\text{single}} = 1$ point of its value under a_0. Leaving a group unchanged is always a candidate.

Integer implementation. For shared quantized weights $\widehat{W}$ partitioned by activation group,

$$\widehat{W}y = \sum_g \widetilde{\Delta}_g \widehat{W}_g(q_g - z_g\mathbf{1}) - \sum_g \widetilde{\Delta}_g c_g \underbrace{\widehat{W}_g\mathbf{1}}_{\bar{w}_g}, \tag{23}$$

where q_g holds the integer codes. The vectors $\bar{w}_g$ are computed once per quantized model and shared across roles, and each role stores only its offsets and enablement masks. Runtime overhead consists of baseline group-error measurement, the enablement check, phase generation, and output correction; integer matrix multiplication remains the main path.

B.2.2 Restore Receivers' Use of Helpful Messages

Attention measurements. The receiver sequence is partitioned into system and beginning-of-sequence tokens, role, task, one segment T_e per incoming edge, and generated tokens. FP and quantized receivers read identical prompts, frozen parent messages, and teacher-forced suffixes. Only edges whose helpful message raises the FP receiver's score over its no-message baseline are

used. For head group g of receiver v, let $m^{\nu}_{v,g,\chi}$ be its average attention mass on segment χ for the FP receiver ($\nu = \mathrm{FP}$) or the quantized receiver ($\nu = a$), and let $\overline{H}^{\nu}_{v,g}$ be its normalized within-message attention entropy, averaged over calibration segments; a head group is the set of query heads that share one key–value head. Head groups whose total FP attention mass on parent messages is below 0.05 are excluded.

Correction direction and step selection. The temperature and segment biases of equation 10 are set per receiver and head group as $\tau_{v,g}$ and $b_{v,g,\chi}$. The initial temperature $\tau^{(0)}_{v,g} > 0$ solves $\overline{H}^{a}_{v,g}(\tau^{(0)}_{v,g}) = \overline{H}^{\mathrm{FP}}_{v,g}$. For nonconstant logits, within-segment entropy increases strictly and continuously with temperature, so an attainable nondegenerate target has a unique solution. At this temperature, with $\epsilon_m = 10^{-4}$, the biases are initialized relative to the role segment:

$$b^{(0)}_{v,g,\chi} = \log \frac{m^{\mathrm{FP}}_{v,g,\chi} + \epsilon_m}{m^{a}_{v,g,\chi}(\tau^{(0)}_{v,g}) + \epsilon_m} - \log \frac{m^{\mathrm{FP}}_{v,g,\mathrm{role}} + \epsilon_m}{m^{a}_{v,g,\mathrm{role}}(\tau^{(0)}_{v,g}) + \epsilon_m}, \qquad b^{(0)}_{v,g,\mathrm{role}} = 0. \tag{24}$$

A segment bias multiplies the segment's unnormalized attention mass by a constant, so this choice matches relative masses for a single attention distribution; averaging over queries gives a shared moment-matching direction.

The applied correction is $b = \eta b^{(0)}$ and $\tau = (\tau^{(0)})^{\eta}$ for $\eta \in \{0.1, 0.2, \ldots, 1.5\}$. On the calibration tasks, we choose the smallest step that restores at least 90% of the FP helpful-message response while keeping the mismatched-message response within its FP tolerance: the score drop caused by the task-mismatched message may exceed the FP receiver's drop by at most 0.5 points. If the unadjusted receiver already meets the 90% target, no step is feasible, or the entropy target is degenerate, the receiver is left unchanged. The stored parameters are the temperatures, segment biases, and head-group masks of the selected receivers and layers.

B.3 Match Receivers and Jointly Adapt

Edges considered for rerouting. After repair, we estimate $U^{a_R}_e$ on the calibration tasks from paired runs. Rerouting considers every edge that is useful in FP and still loses value, $U^{\mathrm{FP}}_e \geq \gamma_{\mathrm{use}}$ and $U^{a_R}_e < U^{\mathrm{FP}}_e$: repair usually restores most but not all of an edge's value, and a receiver that lacks the message's information may use it better. Edges not useful in FP keep their receivers. The same rule applies when repair retains a_0.

Legal candidates. For a considered edge $u \to v$ in the current graph G_k, candidates are the allowed edits of Section 3 that move u's message: reassigning it to another receiver r, or swapping receivers with another sender x, so that $u \to v, x \to w$ becomes $u \to w, x \to v$. Every candidate keeps the sources, one successor per non-sink node, acyclicity, and sink reachability, so v must keep another parent. The original receiver remains a candidate.

Matching profiles. Profiles are defined over common calibration units $\mathcal{Z}$: task–assert pairs for programmatically verified tasks and whole tasks otherwise. For continuous scores such as F1, units carry clipped nonnegative weights. Under the retained model a_R:

- S_u: units where sender u's message adds positive value to a fixed FP reader with frozen sibling messages.
- N_r: units not covered by receiver r's other inputs, measured on its current inputs for a reassignment and with the displaced parent removed for a swap.
- h_r: the median retained fraction of FP helpful-message response over receiver r's probes whose mismatched-message response stays within the FP tolerance of Appendix B.2.2, clipped to $[0, 1]$ and set to zero if no probe is usable.

Sender profiles are refreshed with the retained model's messages; receiver records are reused only when their inputs and execution settings are unchanged. Given the profiles, the compatibility in equation 11 requires no model calls. The candidates of all considered edges are ranked together by κ, ties going to the edge with lower retention $U^{a_R}_e / U^{\mathrm{FP}}_e$, whose denominator is at least γ_{use}; when a swap moves two senders, maximum-weight matching ranks one-to-one assignments under the graph

constraints. The original assignment is kept when S_u is empty or every alternative scores zero, and only the top-ranked graphs receive recalibration budget.

Candidate recalibration and acceptance. For each candidate G', we collect its affected FP traces and rerun initialization with $\pi_{G'}$ over the new graph and its ablations, reusing traces only when their contexts and distributions are unchanged. Edge values are then remeasured and the repair rules of Appendix B.2 reapplied, sender before receiver, with the same retention test against the candidate's initialized model; this yields $a_{G'}$. Candidates are recalibrated in ranking order, one recalibration each, up to $N_{\text{recal}} = 3$ in total, a third of the nine single-edit graphs of our workflow. A candidate improves on the current pair (G_k, a_k) if $J(G', a_{G'}) > J(G_k, a_k)$ on the development tasks. The best improving candidate becomes the new pair; if budget remains, edge values and profiles are refreshed and ranking repeats. The sequential search baselines do not recalibrate, so they evaluate all nine graphs with their fixed models.

B.4 Complete Procedure

Given (G_0, W, β), disjoint calibration and development tasks, and N_{recal}:

1. Collect FP references for the original workflow and its edge ablations, initialize a_0, and measure $U_e^{a_0}$ (Appendix B.1).
2. Diagnose the edges useful in FP but not under a_0, calibrate sender and then receiver repairs, and apply the retention test to obtain a_R (Appendix B.2).
3. Find the FP-useful edges that still lose value (Appendix B.3); stop if none remains.
4. Form profiles, rank legal edits, recalibrate candidate models, and accept the best improving graph–model pair (Appendix B.3). Retain the current pair if none improves.
5. After acceptance, refresh the measurements and repeat; stop when no candidate is accepted or the recalibration budget is exhausted.

Return the accepted graph and its adapted model at the prescribed bit width.

C Experimental Details and Additional Results

C.1 Tasks and Data Splits

We evaluate sanitized MBPP (Austin et al., 2021) and HumanEval (Chen et al., 2021) by pass@1, GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021) by answer exact match, and HotpotQA (Yang et al., 2018) by answer F1. We test on all 427 sanitized MBPP problems, the complete HumanEval (164) and GSM8K (1,319) test sets, and 500 uniformly drawn questions each from the MATH test set and the HotpotQA distractor development set.

Calibration and development tasks never come from a test set. The in-domain setting (MBPP, HumanEval) covers code answers, which must implement a function interface and pass unit tests; it uses the 301 MBPP training and validation problems outside the sanitized release, split into 60 calibration and 241 development tasks. The cross-task setting (GSM8K, MATH, HotpotQA) uses DROP (Dua et al., 2019), whose F1-scored questions require both passage reading and arithmetic; we draw 1,000 questions from its development set, two per passage, and split them by passage into 200 calibration and 800 development tasks.

C.2 Models and Workflow Graphs

We use Llama-3.1-8B-Instruct, Qwen3-8B, and Mistral-Small-3.2-24B-Instruct-2506, all in non-thinking mode with greedy decoding; FP runs use the released BF16 weights. Qwen3-8B is the main model; the other two repeat the main comparison. Six agents share one model: a planner and a requirements checker brief a solver, whose draft and an independent verifier's checks go to a reviewer that corrects the draft for the final solver. Number the agents 0–5 as planner, requirements checker, verifier, solver, reviewer, and final solver. The original graph has edges $0 \to 3$, $1 \to 3$, $2 \to 4$, $3 \to 4$, and $4 \to 5$. One allowed edit (Section 3) yields nine alternative graphs with the same roles

and six calls. These consist of seven reassignments, giving the sender the new edge $0 \to 4$, $0 \to 5$, $1 \to 4$, $1 \to 5$, $2 \to 3$, $2 \to 5$, or $3 \to 5$, and two swaps that exchange the verifier's receiver with that of the planner or the requirements checker. The reviewer cannot move, because the final solver would lose its only input.

C.3 Baselines and Calibration Protocol

W4A8 is the main precision, with W8A8 and W4A4 as milder and stronger quantization. At every precision we compare four PTQ approaches: GPTQ (Frantar et al., 2022) for second-order weight rounding, OmniQuant (Shao et al., 2024) for learned clipping and equivalent transformation, QuaRot (Ashkboos et al., 2024) for rotation, and SERQ (Park et al., 2026) for saliency-aware error reconstruction. The two sequential baselines of Section 6 run GPTSwarm (Zhuge et al., 2024) edge optimization over the same nine single-edit graphs.

Every method quantizes activations with the same dynamic asymmetric quantizer on each token's 128-channel groups (Appendix B.1); OmniQuant, QuaRot, and SERQ add their own transformations or reconstruction on top of it. Every PTQ baseline calibrates on the same FP workflow traces as Q-RoR. Within each calibration setting, every method compiles one graph–model pair chosen on the development tasks and deploys it unchanged on all test tasks of that setting.

C.4 Results at Additional Precisions

Table 2: **Comparisons at two additional precisions.** Both panels use Qwen3-8B and the baselines of Table 1; rules separate PTQ on the original graph, topology search before quantization, and search after it.

	In-domain		Cross-task			
Method	MBPP	HumanEval	GSM8K	MATH	HotpotQA	Avg.
W8A8: milder weight–activation quantization						
GPTQ	67.9	83.5	91.4	60.8	50.6	70.9
OmniQuant	67.7	82.9	91.2	61.0	50.9	70.7
QuaRot	68.1	83.5	91.5	60.6	50.8	70.9
SERQ	67.9	83.5	91.3	61.2	50.7	70.9
Search → SERQ	68.1	83.5	91.4	61.0	50.9	71.0
SERQ → search	67.9	82.9	91.5	61.2	50.8	70.9
Q-RoR	**68.4**	**84.1**	**91.6**	**61.4**	**51.1**	**71.3**
W4A4: stronger weight–activation quantization						
GPTQ	61.6	74.4	86.7	52.0	44.8	63.9
OmniQuant	60.9	75.0	87.0	51.6	46.6	64.2
QuaRot	61.8	75.0	85.8	53.2	48.2	64.8
SERQ	64.6	79.9	88.6	55.8	47.9	67.4
Search → SERQ	64.2	80.5	88.4	55.0	47.6	67.1
SERQ → search	65.3	79.3	89.0	56.8	48.4	67.8
Q-RoR	**65.8**	**81.1**	**89.5**	**58.0**	**48.9**	**68.7**

Average is the unweighted mean of the five displayed task metrics.

Table 2 repeats the comparison of Table 1 at W8A8 and W4A4 with the same baselines. At W8A8, every method stays within 0.7 points of the FP average (71.4) and Q-RoR within 0.1. At W4A4, the PTQ baselines spread over 3.5 points, from GPTQ at 63.9 to SERQ at 67.4, and Q-RoR reaches 68.7, 0.9 points above the strongest baseline, SERQ → search.

C.5 Component Ablation and Calibration Comparison

Table 3 lists the complete method and six removals under the Qwen3-8B W4A8 setting.

Table 3: **Component ablation.** All variants use the same Qwen3-8B W4A8 setting as Table 1.

Variant	MBPP	HotpotQA	Five-task avg.	Δ avg.
Full Q-RoR	67.7	50.6	70.5	+0.00
w/o communication initialization	66.0	49.6	69.2	-1.31
w/o sender repair	66.7	50.7	70.2	-0.33
w/o receiver repair	67.2	49.7	69.8	-0.74
w/o both repairs	66.5	49.8	69.6	-0.95
w/o rerouting	67.0	49.9	69.9	-0.69
w/o candidate recalibration	67.2	50.3	70.1	-0.42

Each removal is recalibrated independently under the same development budget. Without communication initialization, repair and rerouting start from row A of Table 4.

Table 4 crosses original-only versus original-plus-ablated contexts with uniform versus sensitivity-weighted reconstruction. Comparing D with B isolates the ablation contexts; comparing D with C isolates sensitivity weighting.

Table 4: **Calibration comparison.** Contexts and sensitivity weighting form a 2×2 design.

ID	Contexts	Weights	MBPP	HotpotQA	Edge drift
A	Original	Uniform	65.1	48.4	2.1
B	Original	Sensitive	65.8	48.3	1.7
C	Original + ablated	Uniform	65.6	49.1	1.4
D	Original + ablated	Sensitive	66.3	49.4	1.2

All four rows stop after calibration, without repair or rerouting; row D is the initialization of Q-RoR. Edge drift is the mean absolute FP–Q edge-value difference in percentage points on the original MBPP graph. All four variants share the task pool, calibration-token budget, candidate grids, and solver.

C.6 Joint Graph–Model Adaptation

The transfer matrix in Figure 5a isolates initialization: a model calibrated on the original graph G_0 or one of two candidates G_1, G_2 is executed on all three, with repair disabled. G_1 swaps the receivers of the planner and the verifier ($0 \to 4$, $2 \to 3$), so the verifier's checks reach the solver; G_2 moves only the verifier's message to the solver ($2 \to 3$). Calibration context changes the graph ranking.

Table 5 extends the comparison to complete pairs, which also include repair and development-based pair selection, on all five datasets. Fixed-graph repair keeps the original graph, and rerouting without recalibration changes the graph while holding the repaired quantized model fixed. The full pair gains 0.7 MBPP point over fixed-graph repair and 0.5 over rerouting without recalibration. The strongest alternative recalibrates SERQ on each of the nine candidate graphs and keeps the best graph–model pair on development tasks; the complete pair exceeds it on every dataset, by 0.4–1.0 points. On Qwen3-8B, in both calibration settings, Q-RoR deploys G_1, Search $\to$ SERQ deploys the reassignment $1 \to 4$ chosen in FP, and SERQ $\to$ search deploys G_2. Both sequential baselines evaluate all nine graphs, so their shortfall comes from judging graphs with a fixed model.

For Figure 5b, four rankings order all nine single-edit graphs and share the candidate recalibration procedure. Matched ranking is the compatibility ranking of Appendix B.3; without the use factor, it sets $h_r = 1$ in equation 11. Structural ranking uses only the graph as a proxy for missing information: receivers with fewer incoming messages come first, and ties go to the receiver closer to the sink. Random ranking averages five random orders. The plotted quantity is the gain of the pair selected on development tasks over repair on the original graph, evaluated on held-out tasks and averaged over the five datasets; Q-RoR stops after three candidates (Appendix B.3), and every ranking reaches the same gain once all nine are evaluated.

Table 5: **Joint graph–model adaptation.** All entries use Qwen3-8B at W4A8.

	In-domain		Cross-task			
Method	MBPP	HumanEval	GSM8K	MATH	HotpotQA	Avg.
SERQ on original graph	66.5	81.7	90.4	58.8	49.5	69.4
SERQ + per-graph recalibration	67.2	82.3	90.7	59.4	50.2	70.0
Q-RoR repair on original graph	67.0	81.7	90.9	59.8	49.9	69.9
Q-RoR, reroute without recalibration	67.2	82.3	90.8	60.0	50.3	70.1
Q-RoR, complete graph–model pair	**67.7**	**82.9**	**91.1**	**60.4**	**50.6**	**70.5**

Per-graph recalibration recalibrates SERQ on each of the nine single-edit graphs and keeps the best graph–model pair on development tasks.

C.7 Inference Runtime

Table 6 reports the wall-clock time of one six-call workflow for Qwen3-8B at W4A8 on an NVIDIA RTX A6000, with calls run one at a time in topological order. GPTQ adds nothing to the W4A8 integer path and is the fastest. Q-RoR stays within 3% of GPTQ: sender repair adds a per-group phase and output correction, receiver repair adds a segment bias and temperature to attention logits, and integer matrix multiplication remains the main path. Rerouting keeps six calls per task. SERQ adds a low-rank reconstruction branch to each quantized layer and is the slowest; Q-RoR is 1.05× faster than SERQ on MBPP and 1.04× on MATH.

Table 6: **Workflow runtime of Qwen3-8B at W4A8.** Seconds per task for one six-call workflow on an NVIDIA RTX A6000; speedup is relative to SERQ.

	MBPP		MATH	
Method	Time (s)	Speedup	Time (s)	Speedup
GPTQ	10.8	1.07×	23.9	1.06×
SERQ	11.6	1.00×	25.3	1.00×
Q-RoR (ours)	11.1	1.05×	24.4	1.04×