

Causal Abstention for Cost-Aware Language-Agent Collaboration

Lei Wei^{1,2,*†}, Ruiyang Huang^{2,3,†}, Zhaolu Kang^{2,†}, Qi Liu^{2,†},

Xiao Peng¹, Haoran Liu⁴, Zonghao Ying⁵, Haoyu Wang⁴,

Ye Shen⁶, Xu Dong², Hongyu Li¹, Kesu Wang¹, Yukun Lian¹, Sicong Xie^{1,✉}

¹Alibaba International Digital Commerce Group ²Peking University ³Southeast University

⁴The Chinese University of Hong Kong ⁵Beihang University ⁶Shanghai Jiao Tong University

✉xiesc13@gmail.com

Abstract

Multi-agent language systems are often deployed under the assumption that collaboration improves reasoning quality, but this assumption breaks down once token cost, latency, and collaboration-induced errors enter the picture. We study protocol selection under a cost-aware utility objective, where the system decides on each input whether to run a collaborative protocol or fall back to solo reasoning. We propose Causal Advantage Routing for Language agents (CARL), a contextual-bandit framework that estimates the conditional collaboration advantage of each protocol over solo reasoning. CARL combines confounded execution logs with budgeted randomized probes, forms one-vs-solo doubly robust scores and shrunken DR-derived pseudo-outcomes, and at deployment uses a one-sided predictive lower bound for a future realized protocol–solo contrast, otherwise abstaining to solo execution. Across math, code, and multi-hop question-answering benchmarks with an open-weight model, CARL improves cost-aware utility and substantially reduces harmful collaboration over fixed protocols, compute-matched solo strategies, MAS routers, cascade methods, and causal routing baselines, indicating that effective multi-agent systems should estimate when collaboration is causally worth its cost rather than invoking it by default.

1 Introduction

Large language models are increasingly embedded in multi-agent systems where specialized instances debate, critique, verify, refine, or decompose problems before producing a final answer (Du et al.,

2023; Li et al., 2023; Wu et al., 2023; Madaan et al., 2023; Shinn et al., 2023; Wang et al., 2026). These systems open multiple reasoning paths and can improve difficult math, coding, and planning tasks (Wei et al., 2022; Wang et al., 2023; Zhou et al., 2023; Yao et al., 2023; Li et al., 2024; Wang et al., 2025; Wei et al., 2026). They fit a broader inference-time compute paradigm that invests additional calls at test time rather than retraining the base model (Snell et al., 2024).

Collaboration consumes tokens, adds latency, and introduces failure modes: a verifier can be confidently wrong, a debate can converge on an incorrect majority, and a critic can overwrite a correct answer. Prior work has also observed that multi-agent gains are heterogeneous and that debate-style procedures can be unreliable in some settings (Liang et al., 2024; Zhu et al., 2025; Huang et al., 2026). In production, the relevant objective is not raw accuracy but cost-aware utility, the reward from a better answer minus the monetary and latency cost of the collaboration used to obtain it.

Existing approaches only partially address this objective. MAS routers such as MasRouter (Yue et al., 2025) select among collaboration modes and role configurations, but are predictive routing methods rather than estimators of a causal one-vs-solo advantage. Cascade methods such as CascadeDebate (Chang et al., 2026) insert multi-agent deliberation at confidence-based escalation boundaries, yet trigger collaboration on difficulty or confidence rather than whether the specific protocol benefits the current input. Cost-aware model routers such as FrugalGPT and RouteLLM (Chen et al., 2023b; Ong et al., 2024; Ding et al., 2024; Srivatsa et al., 2024), together with causal LLM routing from observational data (Tsiourvas et al., 2025), generally

*Work done during internship at Alibaba.

† Equal contribution.

✉Corresponding author.

route among models rather than collaborative protocols. None directly targets the input-specific cost-aware utility difference between running protocol T_k and solo reasoning.

We call this quantity the conditional collaboration advantage, the central estimand for protocol selection in our framework. Probe-supported logs allow one-vs-solo doubly robust scores whose conditional expectation recovers this estimand when either the relevant outcome-model pair or propensity pair is correct (Robins et al., 1994; Dudik et al., 2011; Chernozhukov et al., 2018). The same comparison directly supports abstention because the system can fall back to solo whenever no collaborative protocol has positive estimated advantage. Building on this view, our method, **Causal Advantage Routing for Language agents (CARL)**, trains on stabilized DR-derived pseudo-outcomes. At deployment, it invokes the protocol with the highest positive one-sided predictive lower bound on future realized protocol–solo contrast and otherwise falls back to solo (Chow, 1970; Vovk et al., 2005; Angelopoulos and Bates, 2021).

Our contributions are as follows.

- We formulate protocol selection as a contextual-bandit problem around direct one-vs-solo causal contrasts. This separates CARL from predictive MAS routers and confidence-triggered cascades, while preserving the same population-optimal action as exact absolute-utility maximization when solo is included.
- We combine cost-decomposed outcome models, doubly robust scores, stabilized pseudo-outcomes, and one-sided pessimistic deployment at the collaboration boundary. The learner targets conditional mean advantage, while calibration and coverage target future realized protocol–solo contrast.
- On math, code, and multi-hop QA benchmarks with an open-weight model, CARL improves cost-aware utility and reduces harmful collaboration over fixed protocols, compute-matched solo strategies, MasRouter-style routing, CascadeDebate-style selective deliberation, causal routing baselines, and causal ablations including DM, IPW, DR-Greedy, and DR-LCB.

2 Related Work

Multi-agent LLM systems and selective collaboration. Collaborative LLM systems include

role-based societies, debate, critique-and-revision, verifier-augmented generation, and agent frameworks (Park et al., 2023; Qian et al., 2023; Hong et al., 2024; Chen et al., 2023c; Chan et al., 2024; Khan et al., 2024; Cobbe et al., 2021; Lightman et al., 2024). Gains are input-dependent, and debate can amplify errors (Cemri et al., 2025; Gao et al., 2025). MasRouter routes among MAS configurations, whereas CascadeDebate escalates to deliberation in a cascade. CARL instead targets causal advantage over solo and treats abstention as an explicit action.

Cost-aware routing, causal routing, and inference-time compute. FrugalGPT and RouteLLM study cost-aware model routing (Chen et al., 2023b; Ong et al., 2024), while Causal LLM Routing learns routing policies from observational data. Their action is a model; CARL’s action is a full protocol—roles, prompts, topology, stopping rule, and aggregation—and its estimand is net advantage over solo.

Abstention, calibration, and doubly robust estimation. Abstention originates in selective classification (El-Yaniv and Wiener, 2010; Geifman and El-Yaniv, 2017). Conformal prediction supports uncertainty-aware decisions (Romano et al., 2019; Angelopoulos et al., 2024), and recent LLM work studies abstention and refusal calibration (Yin et al., 2023; Xiong et al., 2024; Chen et al., 2023a; Kirichenko et al., 2025). CARL abstains from collaboration rather than answering and adapts doubly robust estimation (Jiang and Li, 2016; Thomas and Brunskill, 2016) to cost-aware one-vs-solo advantage.

3 Methodology

3.1 Setup and the Collaboration Advantage

Let $X \in \mathcal{X}$ denote the pre-treatment context of one episode. It contains the raw input and frozen embedding; input descriptors such as token length, task type, and benchmark identifier; Code retrieval features such as BM25 scores and document-length statistics; external difficulty signals; and the entropy of a routing-only solo probe whose generated text is discarded. Every entry is computed before routing. Solver messages, verifier critiques, debate transcripts, and judge rationales are excluded as post-treatment mediators. The full feature list and sensitivity analysis are in sections B and H.

The finite catalog $\mathcal{T} = \{T_0, \dots, T_K\}$ specifies

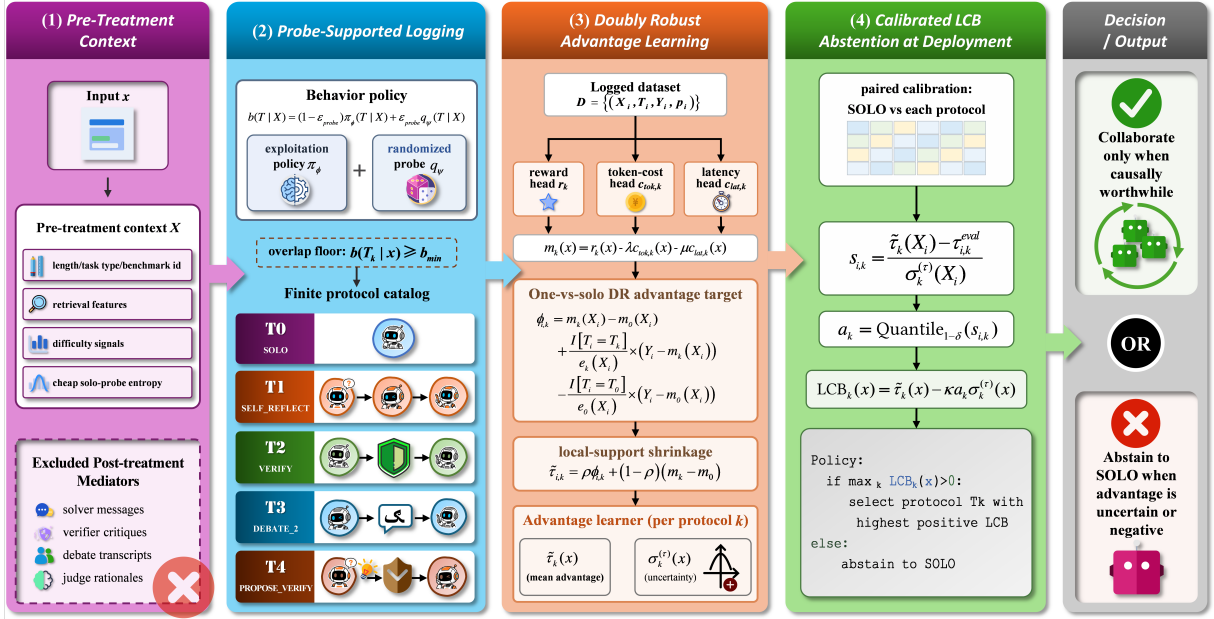


Figure 1: Overview of the CARL framework, which estimates conditional mean collaboration advantages from probe-supported logs and uses a calibrated lower predictive bound on future realized protocol–solo contrast to decide when to collaborate.

each protocol’s checkpoint, prompts, roles, decoding, topology, turn budget, stopping rule, and aggregation. We use five protocols: SOLO (T_0), the one-agent abstention baseline; SELF_REFLECT, which adds a critique-and-revision turn; VERIFY, which introduces an independent verifier over two turns; DEBATE_2, a two-proposer, two-to-three-turn debate adjudicated by a judge; and PROPOSE_VERIFY, which verifies multiple proposals over three turns.

For protocol T_k the cost-aware potential outcome is

$$Y(k) = R(k) - \lambda C_{\text{tok}}(k) - \mu C_{\text{lat}}(k), \quad (1)$$

where $R(k) \in [0, 1]$ is exact match for Math, the official resolved indicator for Code, and F1 for QA. C_{tok} is total input-plus-output tokens divided by a task-specific budget, and C_{lat} is end-to-end latency divided by a reference latency. The budgets are 8,000 tokens and 30 seconds for Math/QA, and 40,000 tokens and 180 seconds for Code. Routing and probe overhead is included, with reused prefix computation counted once. This normalization places all terms on roughly the unit scale within our setup, but does not make Y comparable with studies using different budgets. We choose $\lambda = 0.05$ and $\mu = 0.02$ on development data so that solo and the cheapest collaborative protocol have comparable mean utility.

Writing $m_k^*(x) = \mathbb{E}[Y(k) | X = x]$ for the conditional mean utility of protocol T_k , CARL targets

the conditional collaboration advantage

$$\tau_k(x) = m_k^*(x) - m_0^*(x), \quad k \geq 1. \quad (2)$$

For stochastic protocol executions, define the paired realized contrast

$$D_k = Y(k) - Y(0), \quad \mathbb{E}[D_k | X = x] = \tau_k(x). \quad (3)$$

The DR learner estimates $\tau_k(x)$, whereas conformal residuals use draws of D_k . Thus the calibrated lower bound predicts a future paired realized contrast under the specified decoding distribution; it is not a confidence bound for the CATE. A protocol is mean-harmful when $\tau_k(x) < 0$, distinct from a single harmful realization $D_k < 0$. The conditional-mean oracle policy collaborates only when the best mean advantage is positive,

$$\pi^*(x) = \begin{cases} T_{k^*}, & \text{if } \tau^* > 0, \\ T_0, & \text{otherwise,} \end{cases} \quad (4)$$

where $k^* = \arg \max_k \tau_k(x)$ and $\tau^* = \max_k \tau_k(x)$. Although this conditional-mean oracle is algebraically equivalent to maximizing $m_k^*(x)$ over a catalog containing T_0 , the one-vs-solo form directly targets the collaboration boundary. Historical logs are difficulty-biased because expensive protocols are often assigned to harder

inputs, so the logged contrast decomposes as

$$\begin{aligned} \mathbb{E}[Y \mid T=T_k] - \mathbb{E}[Y \mid T=T_0] \\ = \mathbb{E}[\tau_k(X) \mid T=T_k] \\ + \mathbb{E}[m_0^*(X) \mid T=T_k] \\ - \mathbb{E}[m_0^*(X) \mid T=T_0], \end{aligned} \quad (5)$$

where the last two terms form a difficulty-selection bias that can make useful protocols appear harmful. CARL therefore estimates causal advantage over solo rather than predicting the highest logged reward.

3.2 Probe-Supported Logs and Propensities

Training data are logged episodes $\mathcal{D} = \{(X_i, T_i, Y_i, b_i, s_i)\}_{i=1}^n$, where s_i records the sampled component for diagnostics and $b_i = b(T_i \mid X_i)$ is the marginal assignment probability of the executed treatment under the live mixture behavior policy

$$\begin{aligned} b(T \mid X) = (1 - \epsilon_{\text{probe}})\pi_\phi(T \mid X) \\ + \epsilon_{\text{probe}} q_\psi(T \mid X). \end{aligned} \quad (6)$$

We use $\epsilon_{\text{probe}} = 0.15$. The randomized component q_ψ is the identification anchor, depends only on X , and is a softmax of epistemic uncertainty, an inverse-count coverage bonus, and a small negative cost term. We enforce $b(T_k \mid x) \geq b_{\min} = 0.02$ for every (x, k) , so the training-time propensity clip $e_{\min} = 0.02$ matches the exploration floor.

Every row used by DM/IPW/SNIPW/DR stores the full vectors $\pi_\phi(\cdot \mid X_i)$ and $q_\psi(\cdot \mid X_i)$ and uses the marginal propensity $b_k(X_i) = (1 - \epsilon_{\text{probe}})\pi_{\phi,k}(X_i) + \epsilon_{\text{probe}}q_{\psi,k}(X_i)$, regardless of which component sampled the action. Using component-specific probabilities would not represent the observed behavior policy; the component label is diagnostic only. Rows without the logged marginal b_{prob} are excluded from causal estimation. In equation (8), $\hat{e}_k(X_i) = \max\{b_k(X_i), e_{\min}\}$; overlap and OPE diagnostics audit the bias introduced by clipping.

3.3 Cost-Decomposed Outcome Models and DR Advantage Scores

A shared encoder $h_\theta(X)$ feeds treatment-specific reward, token-cost, and latency heads, and the utility prediction recombines them as

$$\hat{m}_k(x) = \hat{r}_k(x) - \lambda \hat{c}_k^{\text{tok}}(x) - \mu \hat{c}_k^{\text{lat}}(x). \quad (7)$$

The three heads are trained on their natural targets (R_i , normalized token cost, and normalized

latency), not composite Y_i . This separates a cost-independent stage—the shared encoder and reward, token-cost, and latency heads, trained once—from a cost-dependent stage comprising the advantage head and conformal quantile, which depend on (λ, μ) through Y . Changing the deployment cost coefficients therefore requires only refitting the advantage head and recalibrating a_k from cached nuisance predictions, rather than retraining the full pipeline. Outcome and cost nuisance models are cross-fitted across five folds; logged marginal propensities are clipped only when inserted into inverse weights.

For each non-solo protocol T_k , the one-vs-solo doubly robust score is

$$\begin{aligned} \phi_{i,k} = \hat{m}_k(X_i) - \hat{m}_0(X_i) \\ + \frac{\mathbb{I}[T_i=T_k]}{\hat{e}_k(X_i)}(Y_i - \hat{m}_k(X_i)) \\ - \frac{\mathbb{I}[T_i=T_0]}{\hat{e}_0(X_i)}(Y_i - \hat{m}_0(X_i)). \end{aligned} \quad (8)$$

Proposition 1 (One-vs-solo doubly robust score).

With $e_k(x) = \mathbb{P}(T=T_k \mid X=x)$, and under consistency, overlap, and conditional exchangeability given X ,

$$\begin{aligned} \mathbb{E}[\phi_{i,k} \mid X_i=x] - \tau_k(x) \\ = \left(1 - \frac{e_k(x)}{\hat{e}_k(x)}\right)(\hat{m}_k(x) - m_k^*(x)) \\ - \left(1 - \frac{e_0(x)}{\hat{e}_0(x)}\right)(\hat{m}_0(x) - m_0^*(x)). \end{aligned} \quad (9)$$

So $\mathbb{E}[\phi_{i,k} \mid X_i=x] = \tau_k(x)$ whenever both relevant outcome models (m_k, m_0) are correct or both relevant propensities (e_k, e_0) are correct.

The score contrasts full protocols rather than models, always uses solo as the reference, and reuses the cost-decomposed residuals so that downstream LCB calibration applies directly to advantage.

To stabilize the DR target under sparse local support we shrink it toward the direct contrast,

$$\tilde{\tau}_{i,k} = \rho_{i,k} \phi_{i,k} + (1 - \rho_{i,k})(\hat{m}_k(X_i) - \hat{m}_0(X_i)), \quad (10)$$

$$\rho_{i,k} = \frac{n_{\text{eff}}(X_i, T_k)}{n_{\text{eff}}(X_i, T_k) + \gamma \hat{v}_{i,k}}, \quad (11)$$

where $n_{\text{eff}}(X_i, T_k)$ is the K_{NN} -nearest-neighbor effective sample size and $\hat{v}_{i,k}$ is the item-bootstrap variance of $\phi_{i,k}$ from nuisance and pseudo-outcome refits within each fold’s training portion; γ is selected on validation data. This stabilization

is not DR-preserving. When $\rho_{i,k} < 1$, $\tilde{\tau}_{i,k}$ is a convex combination of the DR score and direct-model contrast, trading bias toward the outcome model for lower variance under sparse support. Thus [proposition 1](#) applies to $\phi_{i,k}$, not $\tilde{\tau}_{i,k}$. We regress $\tilde{\tau}_{i,k}$ on X_i to obtain $\tilde{\tau}_k(x)$. The standard deviation of five item-bootstrap advantage heads gives epistemic uncertainty $\hat{\sigma}_k^{(\tau)}(x)$; validation probes separately estimate a realized-contrast noise floor $\hat{\sigma}_k^{(\varepsilon)}$. Their quadrature sum, $\hat{\sigma}_k^{(D)}(x) = \sqrt{(\hat{\sigma}_k^{(\tau)}(x))^2 + (\hat{\sigma}_k^{(\varepsilon)})^2}$, provides a comparable predictive scale for every protocol.

3.4 Calibrated Predictive-Lower-Bound Abstention

Rather than simply maximizing predicted utility, CARL quantifies uncertainty at the abstention boundary $\tau_k(x) = 0$ and collaborates only when it is likely to improve on solo after cost.

We hold out 1,500 paired-execution instances each for Math and QA and 300 for Code ([table 7](#)). QA calibration and test are disjoint stratified partitions of the same public labeled development pools; Math calibrates on MATH training items and evaluates on MATH-500, while Code calibrates on SWE-bench train and evaluates on SWE-bench Lite. Across splits, task definitions, protocols, outcome construction, and decoding distributions remain fixed. SOLO and every candidate protocol are executed once per calibration item, with stochastic seeds drawn from the evaluation distribution. These runs yield $D_{i,k}^{\text{eval}} = Y_i(k) - Y_i(0)$, noisy draws around $\tau_k(X_i)$ rather than conditional means. Additional replicates may be used for variance diagnostics, but averaging a few would not justify CATE coverage unless deployment targets the same replicate-average outcome. In lower-budget settings, randomized audit probes can replace the paired matrix at the cost of wider predictive bounds. On the calibration split we form one-sided standardized residuals

$$s_{i,k} = \frac{\tilde{\tau}_k(X_i) - D_{i,k}^{\text{eval}}}{\hat{\sigma}_k^{(D)}(X_i) + 10^{-6}}, \quad (12)$$

and set $a_k = s_{(r),k}$ for $n_{\text{cal},k}$ exchangeable scores, where $r = \min\{n_{\text{cal},k}, \lceil (n_{\text{cal},k} + 1)(1 - \delta) \rceil\}$ is the finite-sample-corrected rank. The scale $\hat{\sigma}_k^{(D)}$ combines estimation uncertainty with a validation-estimated outcome-noise floor. At deployment, $\text{LCB}_k(x) = \tilde{\tau}_k(x) - \kappa a_k \hat{\sigma}_k^{(D)}(x)$. We retain the

algorithm’s LCB notation, but statistically this is a lower predictive bound for D_k , not a confidence bound for τ_k . The policy is

$$\hat{T}(x) = \begin{cases} T_{\hat{k}}, & \text{if } \max_{k \geq 1} \text{LCB}_k(x) > 0, \\ T_0, & \text{otherwise,} \end{cases} \quad (13)$$

with $\hat{k} = \arg \max_{k \geq 1} \text{LCB}_k(x)$. The main experiments use $\delta = 0.1$ and $\kappa = 1.0$. We report conditional selected-action realized-contrast coverage: among inputs assigned a non-solo action, the fraction satisfying $D_{\hat{T}(X)} \geq \text{LCB}_{\hat{T}(X)}(X)$. SOLO is excluded from both numerator and denominator. Fixed-treatment predictive coverage requires exchangeable item–execution pairs and matching calibration/evaluation decoding distributions, and does not imply coverage of $\tau_k(x)$.

Proposition 2 (Calibration-dependent harm diagnostic). *If $\mathbb{P}(D_k \geq \text{LCB}_k(X)) \geq 1 - \delta$ for every $k \geq 1$, then*

$$\mathbb{P}(D_{\hat{T}(X)} < 0, \hat{T}(X) \neq T_0) \leq K\delta. \quad (14)$$

This is a diagnostic for realized execution harm rather than a CATE guarantee, and the bound can be loose for nontrivial K . Empirical coverage conditional on the selected action does not establish the per-treatment premise of [Proposition 2](#). We therefore report harmful-collaboration rate and conditional selected-action realized-contrast coverage alongside utility in [section 4](#).

3.5 Training and Inference Summary

Training proceeds in three stages. We first collect probe-supported execution logs with the marginal action propensities. We then cross-fit the outcome models, construct the shrunk DR-derived targets in [equations \(8\) and \(10\)](#), and train the advantage and uncertainty estimators. Finally, held-out paired executions calibrate the predictive lower bounds in [equation \(12\)](#). At inference, CARL selects the collaborative protocol with the largest positive lower bound and otherwise uses SOLO.

4 Experiments

4.1 Setup

Model and benchmarks. All main experiments use Qwen/Qwen3.5-27B under fixed, constrained single-pass protocols. Code uses BM25-retrieved context, without interactive repository editing or test-feedback repair, and all methods use consistent

Method	Math (MATH-500)			Code (SWE Lite)			QA (MuSiQue+2Wiki)			Avg.
	Acc	Util	CHR	Res. ³⁰⁰	Util. ³⁰⁰	CHR ¹⁰⁰	F1	Util	CHR	Util
<i>Group 1: Fixed protocols and compute-matched solo</i>										
Always-SOLO	.54	.53	N/A	.28	.26	N/A	.49	.48	N/A	.42
Always-VERIFY	.55	.52	.30	.30	.25	.50	.50	.47	.40	.41
Always-DEBATE	.56	.48	.45	.31	.24	.65	.51	.43	.55	.38
Best-Collab.-Fixed	.55	.52	.30	.30	.25	.50	.50	.48	.36	.42
Solo-SelfConsist.	.60	.55	N/A	.32	.28	N/A	.55	.51	N/A	.45
Solo-Rerank	.61	.56	N/A	.33	.29	N/A	.55	.52	N/A	.46
<i>Group 2: Predictive MAS routing and selective collaboration</i>										
MasRouter-style	.60	.55	.24	.34	.30	.30	.55	.51	.26	.45
MasRouter-style+Solo	.61	.57	.17	.36	.32	.22	.56	.53	.19	.47
CascadeDebate-style	.60	.56	.19	.35	.31	.24	.56	.52	.22	.46
Adaptive-Orch.+Abs.	.61	.57	.18	.35	.31	.23	.56	.52	.20	.47
<i>Group 3: Causal / offline bandit components</i>										
DM-Greedy	.58	.54	.21	.34	.30	.25	.53	.49	.23	.44
DR-Greedy	.61	.57	.16	.36	.32	.21	.56	.52	.18	.47
DR-LCB (absolute util.)	.62	.58	.14	.37	.33	.17	.57	.53	.16	.48
Outcome-Reg.+LCB	.60	.57	.13	.36	.32	.19	.56	.52	.17	.47
CARL	.64	.61	.08	.39	.35	.11	.59	.56	.11	.51
Hindsight Oracle	.71	.68	.00	N/A	N/A	N/A	.65	.60	.00	N/A

Table 1: Five-seed main results. Best-Collab.-Fixed excludes SOLO and uses VERIFY/VERIFY/SELF_REFLECT for Math/Code/QA. Hindsight Oracle chooses the highest-realized-utility catalog action per item, so its CHR is zero by construction; unlike the conditional-mean oracle in equation (4), it is not deployable. CHR is N/A for solo-only methods. Code Res./Util. use all 300 Lite instances; Code CHR and hindsight-oracle utility use the 100-instance audit subset, so Code and cross-task oracle averages are omitted (table 29).

decoding settings. We evaluate three task families: MATH-500 (Lightman et al., 2024) (Math, exact-match); SWE-bench Lite (Jimenez et al., 2024) with the official execution-based harness (Code, resolved indicator on the 300 Lite test instances); and MuSiQue-Ans (Trivedi et al., 2022) plus 2Wiki-MultiHopQA (Ho et al., 2020) stratified by hop count (QA, F1). Each task has four disjoint splits plus a specified all-action evaluation population (table 7); Math and QA replay the test set under all protocols, while Code runs all-action diagnostics on a stratified 100-instance audit subset. Cross-model runs are exploratory; no hyperparameter or calibration quantile is tuned on test or audit data.

Baselines. We group baselines into fixed protocols and compute-matched solo (Group 1), predictive MAS routers and selective-deliberation methods (Group 2), and causal or offline-bandit ablations (Group 3). The +Solo variants give a router the same abstention action as CARL. Full definitions and additional baselines (Confidence-Trigger, Causal-Routing-style, Naive-Obs.) are in section D.

Metrics. The primary metric is cost-aware utility Y ; task-native reward (EM, resolved rate, or F1)

is secondary. CHR is the fraction of all test inputs assigned a collaborative protocol with $D_k < 0$, whereas PTC is the fraction of collaborative selections with $D_k \geq 0$; their denominators differ. Conditional selected-action coverage measures $D_{\hat{T}(X)} \geq \text{LCB}_{\hat{T}(X)}(X)$ among non-solo selections. Realized-contrast ECE is the weighted prediction–outcome gap over 10 equal-frequency bins. Cross-split OPE discrepancy averages the absolute gap between training-log fixed-policy estimates and test-audit values over arms and seeds; it also reflects population shift. Results average five seeds.

4.2 Main Results

Overall result. Table 1 reports the main result. CARL achieves the highest non-oracle cost-aware utility on all three task families, with asymmetric gains across tasks. On Math, CARL improves utility by 0.03 over DR-LCB and 0.04 over MasRouter-style+Solo, and its CHR is 0.08 compared with 0.14 for DR-LCB and 0.17 for MasRouter-style+Solo. On QA, the utility gap to the strongest learned baseline is 0.03, and the dominant gain is the CHR drop from 0.16 to 0.11. On Code, the resolved-rate improvement over

Task	Est.	SR	VER	DEB	PV
Math	Log	−.012	−.071	−.029	−.046
	Audit	−.018	−.012	−.049	−.038
Code	Log	+.003	−.054	−.041	−.061
	Audit	+.011	+.022	+.008	+.014
QA	Log	+.019	+.008	−.058	−.030
	Audit	+.002	−.011	−.047	−.014

Table 2: Logged contrast vs. audit-matrix contrast of each non-solo protocol’s advantage over solo. Because the training logs and audit matrices come from different benchmark splits, their gaps combine assignment/composition effects with train–test distribution shift and do not isolate confounding. Math/QA audit means are aligned with the corresponding fixed-policy utilities in [table 1](#), up to unrounded values.

MasRouter-style+Solo is about three points, with a comparable utility gain after accounting for patch-generation cost, and the CHR on the audit subset drops from 0.22 to 0.11.

Why fixed and predictive baselines fall behind.

These gaps are driven by substantial input-level heterogeneity: on the all-action evaluation populations, no collaborative protocol has positive realized advantage on 46% of inputs on average, while at least one non-solo protocol is beneficial on the remaining 54%. Verification is harmful on roughly one third of math instances and debate on more than half of code audit instances. CARL’s selected-action mix tracks this pattern—solo dominates on Code (51%) and QA (55%), while verification carries most of the non-solo mass on Math. Per-task and per-protocol breakdowns are in [section I](#).

Targeted comparisons. Three comparisons are worth singling out. Adding the solo abstention action to a predictive router (MasRouter-style $\rightarrow$ MasRouter-style+Solo) accounts for most of the CHR improvement on QA and Code, but the residual gap to CARL on both utility and CHR remains, suggesting that calibration at the collaborate-versus-solo boundary matters when solo is already adequate. CascadeDebate-style triggers deliberation when solo confidence is low, which is helpful on QA but less so on Code, where confidence on hard issues is a poor proxy for whether a multi-agent patch will resolve the test. DR-LCB places separate lower bounds on absolute utilities, whereas CARL models the paired contrast and therefore retains the covariance between protocol and solo predictions when calibrating the

Task	DM	IPW	SNIPW	DR
Math	.047	.039	.033	.021
Code	.078	.065	.054	.034
QA	.053	.046	.038	.027

Table 3: Cross-split OPE discrepancy against the audit matrix, averaged over five seeds. For each seed, the discrepancy is the mean absolute difference between the training-log estimate of each fixed catalog policy and its paired test-audit value, averaged over the five catalog arms. Because the two sides target different benchmark splits, this quantity combines estimator error with population shift; DR has the lowest descriptive discrepancy on every task.

decision boundary. With exact means and no uncertainty penalty, the two parameterizations have the same argmax; the empirical difference is attributable to estimation and one-sided calibration, not to a different oracle target.

Hindsight-oracle recovery. Using per-task hindsight-oracle recovery $(U(\pi) - U(\text{Solo})) / (U(\text{Hindsight Oracle}) - U(\text{Solo}))$, CARL recovers 53% of the hindsight-oracle improvement on Math and 67% on QA. Code recovery is computed entirely on the same 100-instance audit subset—Solo .25, CARL .34, and Hindsight Oracle .44—giving 47%; it never mixes the full-test CARL utility with an audit-only oracle.

4.3 Confounding Diagnostics

A central motivation for CARL is that observational protocol logs are difficulty-biased, since harder inputs are more likely to be routed to expensive collaborative protocols. We therefore do not interpret logged contrasts as causal effects on their own. Identification relies on correctly logged marginal behavior propensities, consistency of the protocol definitions, conditional exchangeability given recorded routing features, and positive overlap in the behavior policy before clipping. Randomized probes improve overlap but do not by themselves repair omitted routing covariates. Sensitivity analyses for the exchangeability assumption are in [section H](#).

[Table 2](#) compares the logged observational contrast with the held-out audit-matrix contrast. For verification, the negative logged contrast is strongly attenuated on Math and reverses on the Code audit subset. These patterns are consistent with harder inputs being preferentially routed to verification,

Variant	Util $\uparrow$	CHR $\downarrow$	PTC $\uparrow$
Full CARL	.59	.10	.81
Absolute utility	.56	.15	.75
Single utility head	.55	.14	.77
No DR shaping	.54	.17	.73
Greedy, no LCB	.59	.22	.73
Uniform probing	.57	.11	.80
No shrinkage	.57	.13	.78
Shared head	.54	.17	.73

Table 4: Component ablations of CARL, averaged over Math and QA.

Method	Util $\uparrow$	CHR $\downarrow$	Abst.
CARL	.59	.10	.48
MasRouter-style+Solo	.55	.18	.30
Outcome-Reg.+LCB	.55	.15	.51
CascadeDebate-style	.54	.21	.46
Solo confidence threshold	.52	.19	.44

Table 5: Simple abstention alternatives, averaged over Math and QA.

but the comparison does not isolate that mechanism because the training-log and test-audit populations also differ. On QA, debate remains harmful under both estimates. Table 3 reports the corresponding cross-split discrepancy for four estimators, with DR lowest on every task.

4.4 Ablations and Abstention Alternatives

Table 4 reports component ablations averaged over Math and QA. Replacing the direct contrast with separately calibrated absolute-utility heads reduces performance; because both parameterizations share the same population argmax when estimated exactly, this ablation supports the value of paired contrast estimation and one-sided uncertainty calibration rather than a novel oracle estimand. Collapsing the cost-decomposed heads into a single utility head also lowers utility and raises CHR, since the resulting model cannot reuse reward and cost structure when coefficients change. Removing DR-derived pseudo-outcome shaping lowers utility further, consistent with outcome regression alone being insufficient under difficulty-biased logs, and removing LCB pessimism keeps utility high but raises CHR substantially, exposing the utility-harm tradeoff at the abstention boundary. Switching to uniform probing gives up sample efficiency without much CHR change, since overlap is preserved

Method	Task	ECE $\downarrow$	PTC $\uparrow$	Cov. $\uparrow$
Outcome-Reg.+LCB	Math	0.10	0.76	0.86
	Code	0.14	0.60	0.81
	QA	0.11	0.61	0.84
DR-LCB (abs.)	Math	0.08	0.80	0.88
	Code	0.12	0.73	0.84
	QA	0.09	0.72	0.87
CARL	Math	0.05	0.86	0.91
	Code	0.08	0.78	0.87
	QA	0.06	0.76	0.89

Table 6: Realized-contrast calibration, PTC, and conditional selected-action predictive coverage among non-solo selections. The 90% line is a reference value: split-conformal coverage concerns future realized contrasts for a fixed treatment under exchangeability, not the CATE or this selection-conditional metric.

either way.

Table 5 contrasts CARL with simple abstention strategies. Calibrated thresholding on solo confidence and confidence-triggered collaboration each reduce some harmful calls but neither matches CARL’s tradeoff, since their abstention boundary is not aligned with the one-vs-solo advantage. The κ sweep and a $b_{\min}$ sensitivity study are in section I.

4.5 Calibration and Coverage

Table 6 reports realized-contrast ECE, PTC, and conditional selected-action predictive coverage for the LCB-based methods. CARL has the lowest empirical ECE on every task and the highest PTC, and its per-task conditional coverage is close to the 90% reference level. Coverage on Code is slightly lower than on Math and QA, consistent with greater outcome noise and estimator variance on the audit subset. These empirical values do not establish the per-treatment premise of Proposition 2.

4.6 Robustness

We evaluate three stress settings: injected protocol failures, rate-limit-style missing responses, and adversarial distractor critiques. Treatment-drift tests, sequential deployment diagnostics, and exploratory additional-model runs (Llama-3.1-8B and Qwen2.5-72B) are in sections I and J.

5 Conclusion

CARL frames protocol selection around the conditional mean collaboration advantage. The population-optimal action is unchanged from exact absolute-utility maximization when solo is in the catalog; CARL’s distinction is direct contrast

estimation plus one-sided predictive calibration of future realized protocol–solo outcomes. Across math, code, and multi-hop QA benchmarks, its unshrunk doubly robust scores, shrunken DR-derived training targets, and calibrated abstention improve cost-aware utility and reduce harmful collaboration.

Limitations

CARL relies on probe-supported data collection, which adds upfront training cost and may be impractical when collaborative protocol executions are expensive on a per-call basis. The causal mean-advantage claims require correctly logged marginal mixture propensities, positive overlap before clipping, and pre-treatment features that capture every routing signal used by the behavior policy. Clipping bounds weights but cannot create overlap. Separately, the conformal claim concerns future realized contrasts under exchangeable item–execution pairs, not the CATE. The harm bound in [proposition 2](#) remains conditional on its calibration premise and loosens linearly with the number of collaborative protocols.

The protocol catalog is finite and manually specified, so CARL does not cover dynamic tool use, adaptive retrieval, memory, or protocol composition. Code evaluation uses SWE-bench Lite with the official harness, but Lite is a subset of SWE-bench and the auxiliary judge-based score in [section F](#) is a sanity check rather than a substitute for execution. The main experiments use a single open-weight model family, and we do not claim contamination-free evaluation because the model’s training cutoff is not auditable. Cross-model results are reported only as exploratory robustness checks.

Declaration of Generative AI Usage

During the preparation of this work, the authors used AI assistants solely for technical formatting of $\text{\LaTeX}$ and coding assistance within the scope of permitted guidelines. Specifically, AI tools were employed to optimize the layout and formatting of LaTeX tables and to assist in error detection and debugging of the experimental code. The authors declare that no AI tools were used for the conceptualization, methodology development, or the writing of the core research content, ensuring the authenticity and integrity of the study. Furthermore, all bibliographic references were manually curated, verified, and imported without the use of

AI, ensuring that all cited sources are accurate and authentic. The authors bear full responsibility for the final content of the manuscript.

Ethical considerations

CARL is a routing layer over existing LLM protocols and does not generate user-facing content beyond what its constituent protocols already produce. Its direct effect is to skip collaborative calls when their estimated advantage over solo is non-positive, which reduces token usage, latency, and energy consumption relative to always-collaborate baselines. We expect the dominant risk to come from miscalibration rather than from new generative capability: a lower bound that is too tight will under-collaborate on inputs where deliberation would have caught an error, and a lower bound that is too loose will spend compute on harmful collaboration. We mitigate this through conservative calibration, audit logs of selected actions and realized advantages, and recalibration whenever the protocol catalog or base model changes.

The benchmarks used here (MATH, SWE-bench Lite, MuSiQue, 2WikiMultiHopQA) are public research datasets used under their original licenses, and we do not redistribute them. The open-weight model we use was released by its original authors; we report serving and decoding settings in the run manifest so others can replicate or audit the runs. CARL should not be deployed in high-stakes settings (medical, legal, safety-critical) without task-specific validation and human review, since the abstention decision concerns whether to invoke a collaborative protocol, not whether the final answer is correct.

References

- Anastasios N. Angelopoulos and Stephen Bates. 2021. [A gentle introduction to conformal prediction and distribution-free uncertainty quantification](#). *Preprint*, arXiv:2107.07511.
- Anastasios N. Angelopoulos, Stephen Bates, Adam Fisch, Lihua Lei, and Tal Schuster. 2024. Conformal risk control. In *International Conference on Learning Representations*.
- Mert Cemri, Melissa Z. Pan, Shuyi Yang, Lakshya A. Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. 2025. Why do multi-agent llm systems fail? *arXiv preprint arXiv:2503.13657*.

- Chi-Min Chan, Weize Chen, Yusheng Su, Jianxuan Yu, Wei Xue, Shanghang Zhang, Jie Fu, and Zhiyuan Liu. 2024. Chateval: Towards better llm-based evaluators through multi-agent debate. In *International Conference on Learning Representations*.
- Raeyoung Chang, Dongwook Kwon, Jisoo Lee, and Nikhil Verma. 2026. [Cascadedebate: Multi-agent deliberation for cost-aware llm cascades](#). *Preprint*, arXiv:2604.12262.
- Jiefeng Chen, Jinsung Yoon, Sayna Ebrahimi, Sercan O Arik, Tomas Pfister, and Somesh Jha. 2023a. [Adaptation with self-evaluation to improve selective prediction in llms](#). *Preprint*, arXiv:2310.11689.
- Lingjiao Chen, Matei Zaharia, and James Zou. 2023b. [Frugalgpt: How to use large language models while reducing cost and improving performance](#). *Preprint*, arXiv:2305.05176.
- Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, Yujia Qin, Xin Cong, Ruobing Xie, Zhiyuan Liu, Maosong Sun, and Jie Zhou. 2023c. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors. *arXiv preprint arXiv:2308.10848*.
- Victor Chernozhukov, Denis Chetverikov, Mert Demirer, Esther Duflo, Christian Hansen, Whitney Newey, and James Robins. 2018. [Double/debiased machine learning for treatment and structural parameters](#). *The Econometrics Journal*, 21(1):C1–C68.
- C. K. Chow. 1970. [On optimum recognition error and reject tradeoff](#). *IEEE Transactions on Information Theory*, 16(1):41–46.
- 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. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.
- Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Ruhle, Laks V. S. Lakshmanan, and Ahmed Hassan Awadallah. 2024. Hybrid llm: Cost-efficient and quality-aware query routing. In *International Conference on Learning Representations*.
- Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. 2023. [Improving factuality and reasoning in language models through multiagent debate](#). *Preprint*, arXiv:2305.14325.
- Miroslav Dudik, John Langford, and Lihong Li. 2011. [Doubly robust policy evaluation and learning](#). In *Proceedings of the 28th International Conference on Machine Learning*, pages 1097–1104.
- Ran El-Yaniv and Yair Wiener. 2010. On the foundations of noise-free selective classification. *Journal of Machine Learning Research*, 11:1605–1641.
- Mingyan Gao, Yanzi Li, Banruo Liu, Yifan Yu, Phillip Wang, Ching-Yu Lin, and Fan Lai. 2025. Single-agent or multi-agent systems? why not both? *arXiv preprint arXiv:2505.18286*.
- Yonatan Geifman and Ran El-Yaniv. 2017. Selective classification for deep neural networks. In *Advances in Neural Information Processing Systems*.
- Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. [Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps](#). In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, Barcelona, Spain (Online). International Committee on Computational Linguistics.
- Sirui Hong, Mingchen Zhuge, Jiaqi Chen, Xiaowu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2024. Metagt: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations*.
- Ruiyang Huang, Chenxi Wang, Tinghe Zhang, Fengrui Liu, Jiayan Sun, Haocheng Wang, and Yifan Wu. 2026. Toposhield: Reshaping the flow of malice via spatio-temporal risk-aware topological evolution in multi-agent systems. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 8740–8768.
- Nan Jiang and Lihong Li. 2016. Doubly robust off-policy value evaluation for reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 652–661.
- Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R. Narasimhan. 2024. [SWE-bench: Can language models resolve real-world github issues?](#) In *International Conference on Learning Representations*.
- Akbir Khan, John Hughes, Dan Valentine, Laura Ruis, Kshitij Sachan, Ansh Radhakrishnan, Edward Grefenstette, Samuel R. Bowman, Tim Rocktäschel, and Ethan Perez. 2024. Debating with more persuasive llms leads to more truthful answers. In *Advances in Neural Information Processing Systems*.
- Polina Kirichenko, Mark Ibrahim, Kamalika Chaudhuri, and Samuel J. Bell. 2025. [Abstentionbench: Reasoning llms fail on unanswerable questions](#). *Preprint*, arXiv:2506.09038.
- Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. [CAMEL: Communicative agents for “mind” exploration of large language model society](#). In *Advances in Neural Information Processing Systems*, volume 36.
- Junyou Li, Qin Zhang, Yangbin Yu, Qiang Fu, and Deheng Ye. 2024. More agents is all you need. *Transactions on Machine Learning Research*.

- Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu. 2024. [Encouraging divergent thinking in large language models through multi-agent debate](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 17889–17904, Miami, Florida, USA. Association for Computational Linguistics.
- Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let’s verify step by step. In *International Conference on Learning Representations*.
- Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. [Self-refine: Iterative refinement with self-feedback](#). In *Advances in Neural Information Processing Systems*, volume 36.
- Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M. Waleed Kadous, and Ion Stoica. 2024. [RouteLLM: Learning to route llms with preference data](#). *Preprint*, arXiv:2406.18665.
- Joon Sung Park, Joseph C. O’Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative agents: Interactive simulators of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*.
- Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2023. Chatdev: Communicative agents for software development. *arXiv preprint arXiv:2307.07924*.
- James M. Robins, Andrea Rotnitzky, and Lue Ping Zhao. 1994. [Estimation of regression coefficients when some regressors are not always observed](#). *Journal of the American Statistical Association*, 89(427):846–866.
- Yaniv Romano, Evan Patterson, and Emmanuel J. Candès. 2019. Conformalized quantile regression. In *Advances in Neural Information Processing Systems*.
- Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. [Reflexion: Language agents with verbal reinforcement learning](#). In *Advances in Neural Information Processing Systems*, volume 36.
- Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*.
- KV Aditya Srivatsa, Kaushal Kumar Maurya, and Ekaterina Kochmar. 2024. Harnessing the power of multiple minds: Lessons learned from llm routing. In *Proceedings of the First Workshop on Insights from Negative Results in NLP*.
- Philip S. Thomas and Emma Brunskill. 2016. Data-efficient off-policy policy evaluation for reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 2139–2148.
- Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. [MuSiQue: Multi-hop questions via single-hop question composition](#). *Transactions of the Association for Computational Linguistics*, 10:539–554.
- Asterios Tsourvas, Wei Sun, and Georgia Perakis. 2025. [Causal llm routing: End-to-end regret minimization from observational data](#). *Preprint*, arXiv:2505.16037.
- Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. 2005. *Algorithmic Learning in a Random World*. Springer.
- Chenxi Wang, Ruiyang Huang, Jiayan Sun, Lei Wei, and Yifan Wu. 2026. Out of sight, not out of mind: Unveiling latent attack in latent-based multi-agent systems. *arXiv preprint arXiv:2605.28214*.
- Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. 2025. Mixture-of-agents enhances large language model capabilities. In *International Conference on Learning Representations*.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.
- Lei Wei, Qi Liu, Ruiyang Huang, Xiao Peng, Sicong Xie, Lanbo Lin, Chenhao Jiang, Yuanwu Xu, Tianyuan Yang, Jiayao Liu, Li Cai, Zhaolu Kang, and Bin Wang. 2026. [Evoc2f: Compiling tool orchestration for efficient and evolvable LLM agents](#). In *Forty-third International Conference on Machine Learning*.
- Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. 2023. [AutoGen: Enabling next-gen llm applications via multi-agent conversation](#). *Preprint*, arXiv:2308.08155.

- Miao Xiong, Zhiyuan Hu, Xinyang Lu, Yifei Li, Jie Fu, Junxian He, and Bryan Hooi. 2024. Can llms express their uncertainty? an empirical evaluation of confidence elicitation in llms. In *International Conference on Learning Representations*.
- Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*.
- Zhangyue Yin, Qiushi Sun, Qipeng Guo, Jiawen Wu, Xipeng Qiu, and Xuanjing Huang. 2023. Do large language models know what they don't know? *arXiv preprint arXiv:2305.18153*.
- Yanwei Yue, Guibin Zhang, Boyang Liu, Guancheng Wan, Kun Wang, Dawei Cheng, and Yiyan Qi. 2025. [MasRouter: Learning to route llms for multi-agent systems](#). *Preprint*, arXiv:2502.11133.
- Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and Ed Chi. 2023. Least-to-most prompting enables complex reasoning in large language models. In *International Conference on Learning Representations*.
- Kunlun Zhu, Hongyi Du, Zhaochen Hong, Xiaocheng Yang, Shuyi Guo, Zhe Wang, Zhenhailong Wang, Cheng Qian, Xiangru Tang, Heng Ji, and Jiaxuan You. 2025. [Multiagentbench: Evaluating the collaboration and competition of llm agents](#). *Preprint*, arXiv:2503.01935.

Appendix

A Benchmark and Dataset Details

Table 7 reports four disjoint data splits per task plus the subset on which all five catalog actions are executed. The all-action column is an evaluation-coverage designation, not a fifth split. Math uses MATH training-pool items for router training, validation, and calibration, and reserves MATH-500 only for final testing. For QA, train and validation are sampled from the public labeled training splits of MuSiQue-Ans and 2WikiMultiHopQA. Calibration and test are disjoint stratified random partitions of the public labeled development split of each source, balanced by source and stratified by hop count and question type (Train: 4,000 +4,000 ; Val: 500 +500 ; Calib.: 750 +750 ; Test: 500 +500). The hidden-label official 2WikiMultiHopQA test split is not used.¹ Code uses SWE-bench Lite with the official execution-based harness. The 23 official Lite dev instances are too few to train a router, so we sample 500 training and 100 validation instances from the SWE-bench train set, with no overlap with Lite test; the 300 calibration instances are drawn from the same source under the same disjointness constraint. The final test split is the full 300 Lite test instances, and all-action diagnostics (CHR, hindsight oracle, OPE, coverage) are computed on a stratified 100-instance audit subset drawn from Lite test. Public benchmark data are used under their original releases; we do not redistribute them and we do not claim contamination-free evaluation. We also compute a judge-based textual patch-quality score on Code, but it is reported only as an auxiliary diagnostic; the resolved indicator from the official Lite harness is the sole reward entering $R(k)$, $Y(k)$, and the DR-derived targets.

Benchmark summary. Math uses MATH training data plus MATH-500 with exact-match reward and no retrieval. Code uses SWE-bench Lite with the official execution-based harness; the resolved indicator is the primary reward, with BM25 file retrieval over the repository as the only retrieval signal in X , and a judge-based textual patch-quality score reported as an auxiliary diagnostic in section F. QA uses MuSiQue-Ans together with 2WikiMultiHopQA with token-level F1 as the reward and

¹The official release states that test examples do not include answers: <https://github.com/Alab-NII/2wikimultihop>.

Task	Train	Val	Calib.	Test	All-action eval.
Math	5,000	1,000	1,500	500	500
QA	8,000	1,000	1,500	1,000	1,000
Code	500	100	300	300	100

Table 7: Four disjoint split sizes and all-action evaluation coverage. Math/QA execute all actions on their test split; Code uses a 100-instance audit subset of its 300-instance test split. QA calibration and test are random disjoint partitions of the same public labeled development pools.

dataset-provided passages (no external retrieval beyond what each benchmark ships with). All reported QA F1 values use public labeled development examples reserved before experimentation; no result uses the hidden-label 2WikiMultiHopQA test split.

QA source	Solo F1	Solo Util	CARL F1	CARL Util
MuSiQue-Ans	.47	.46	.58	.55
2Wiki	.51	.50	.60	.57
Pooled	.49	.48	.59	.56

Table 8: QA results by source. Pooled values are equally weighted; evaluation uses reserved labeled development examples.

Harness execution. Each Code patch is evaluated with the official SWE-bench Lite harness using a 15-minute per-instance timeout. Patches that fail to apply, run-time errors, and timeouts are all counted as unresolved. Protocol outputs and harness outcomes are cached by (instance, protocol, seed), so evaluating multiple routing policies reuses the same potential-outcome cache rather than re-running an identical patch per method. The 100-instance audit contains $100 \times 5 \times 5 = 2,500$ executions and is a subset, not an addition, of the at-most $300 \times 5 \times 5 = 7,500$ full-test catalog executions; it therefore represents one third of that maximum. Actual execution counts and cache-hit rates are recorded in the final run manifest.

B Pre-treatment Features and Excluded Mediators

Table 9 lists the routing covariates that enter X and the post-treatment signals that are deliberately excluded. The encoder sees only the raw task input plus pre-treatment retrieved context. The entropy of the routing-only solo probe is computed from a 32-token generation; its generated text is

discarded and never reaches the protocol catalog. Its incremental tokens and latency are included in learned-policy costs, with shared prefix computation counted once.

Group	Examples	In X ?
Input descriptors	length, task type, benchmark id	yes
Raw pre-treatment input	task text and frozen embedding	yes
Retrieval (Code)	BM25 score, top- k doc length	yes
Difficulty signals	problem-class, task-source ids	yes
Solo probe proxy	entropy of 32-token routing probe	yes
Solver messages	generated answers, scratchpads	no
Verifier critiques	verifier outputs, votes	no
Debate transcripts	turn-level rationales	no
Judge rationales	adjudication reasoning	no

Table 9: Routing features and excluded post-treatment signals.

C Protocol Details

The finite protocol catalog is fixed before data collection and matches the protocol IDs used in the main text. Each treatment definition specifies the model identifier, prompts, roles, decoding parameters, communication topology, turn budget, stopping rule, and aggregation rule. Generated solver messages, verifier critiques, debate transcripts, and judge rationales are post-treatment variables and not used as routing covariates. Prompts, decoding parameters, and stopping rules are recorded in the experiment configuration. Each treatment definition is SHA-256 hashed over all prompts, decoding parameters, and checkpoint identifiers before data collection, and the treatment-definition audit log is stored with each episode.

D Baseline Details

The +Solo variants expose T_0 at inference, giving MasRouter-style and Causal-Routing-style the same abstention action space as CARL.

MasRouter-style and +Solo. A cascaded controller with a stage-1 binary head (collaborate vs. not) and a stage-2 protocol selector. Both stages train on logged cost-aware utility with the same features and capacity as CARL. The +Solo variant exposes T_0 to the selector at inference, matching CARL’s action space.

CascadeDebate-style. Solo runs first; if its confidence falls below a validation-tuned threshold, VERIFY runs; if VERIFY confidence remains below a second threshold, DEBATE_2 runs. Thresholds are tuned on validation utility.

Causal-Routing-style and +Solo. A causal-routing objective on protocol-level utility. The +Solo variant treats T_0 as a selectable action with target $\hat{m}_0(x)$. With exact conditional means it has the same population argmax as the one-vs-solo parameterization; unlike CARL, however, it does not directly model or one-sidedly calibrate the paired collaboration advantage $\hat{m}_k(x) - \hat{m}_0(x)$.

E Full Main Table with Additional Baselines

Table 10 extends table 1 with the four baselines moved here in the main paper (Confidence-Trigger, Causal-Routing-style, Causal-Routing-style+Solo, Naive-Obs.). None of them is competitive with CARL, but reporting them keeps the comparison auditable.

Method	Math		Code		QA	
	Util	CHR	Util	CHR	Util	CHR
Naive-Obs.	.52	.26	.22	.31	.48	.29
Confidence-Trigger	.55	.18	.28	.23	.50	.21
Causal-Routing-style	.56	.16	.29	.20	.51	.18
... +Solo	.57	.14	.30	.18	.52	.16

Table 10: Additional baselines on the same three task families. Code CHR uses the 100-instance SWE-Lite audit subset.

F Code Textual Patch-Quality Diagnostic

The auxiliary judge-based patch-quality score is computed by prompting the same model family to rate each generated patch on a 5-point rubric covering localization, edit minimality, and apparent correctness, normalized to $[0, 1]$. Table 11 compares textual score and resolved indicator for the main methods on Lite test. The textual score is consistently higher than the resolved indicator and ranks methods similarly, but it does not require execution. We treat it as a sanity check rather than a substitute for the official harness.

Method	Tex.	Res.
Always-SOLO	.42	.28
Best-Collab.-Fixed	.47	.30
MasRouter-style+Solo	.52	.36
CascadeDebate-style	.51	.35
DR-LCB (absolute)	.53	.37
CARL	.58	.39

Table 11: SWE-bench Lite test: judge textual patch-quality score (Tex.) vs. official resolved rate (Res.).

G Calibration and Selected-Action Realized-Contrast Coverage

Table 12 reports realized-contrast predictive coverage conditional on each collaborative protocol being selected. SOLO is omitted because it has no protocol-solo contrast. Coverage on Code is slightly below the 90% reference for debate-style treatments, consistent with greater execution noise on the audit subset. These empirical values neither establish selection-conditional guarantees nor cover the CATE.

Task	SR	VER	DEB	PV
Math	.93	.91	.90	.89
Code	.88	.87	.84	.86
QA	.89	.90	.87	.88

Table 12: Per-treatment realized-contrast predictive coverage of CARL conditional on the non-solo protocol being selected. Math and QA use the final test matrix; Code uses the 100-instance audit subset.

H Confounding, OPE, and Sensitivity Diagnostics

Propensity overlap. Table 13 reports the mean marginal behavior probability, the 5th percentile (a low-tail overlap proxy), the effective sample size, and the largest realized inverse-propensity weight under $e_{\min} = 0.02$. For arm k , $w_i = \mathbb{I}[T_i = T_k] / \max\{b_k(X_i), e_{\min}\}$ and $ESS_k = (\sum_i w_i)^2 / \sum_i w_i^2$. Because only rows assigned to arm k have nonzero weight, $ESS_k \leq n_k$; the table therefore reports ESS/n_k , where $n_k = \sum_i \mathbb{I}[T_i = T_k]$, rather than the mathematically incompatible ESS/n normalization. At the default $b_{\min} = e_{\min} = 0.02$, clipping should not bind when propensities are logged exactly; it remains a guard against numerical or logging-roundoff violations.

Treatment	Mean prop.	5th pct.	ESS/n_k	Max w_i
SOLO	0.38	0.18	0.82	5.0
SELF_REFLECT	0.17	0.07	0.74	8.0
VERIFY	0.21	0.08	0.76	8.0
DEBATE_2	0.12	0.04	0.61	10.0
PROPOSE_VERIFY	0.12	0.04	0.59	10.0

Table 13: Propensity overlap on training logs.

Placebo treatments. Table 14 reports placebo treatments that should not look helpful if the advantage learner is well calibrated. “Random critique”

inserts noise; “Shuffled verifier” swaps verifier inputs across items; “Extra summary agent” adds a redundant turn.

Placebo	Est. adv.	Audit gain	Sel. %
Random critique	−0.03	−0.04	2%
Shuffled verifier	−0.05	−0.06	1%
Extra summary agent	−0.01	−0.02	4%

Table 14: Placebo treatments: estimated advantage, true audit gain, and CARL selection rate.

Bootstrap standard errors. Table 15 reports bootstrap SE ($B = 1000$) for the logged and audit contrasts in table 2. The resampling unit is the benchmark item: all protocol outcomes and all seed-specific predictions for a sampled item remain clustered together. Code resamples the 100 audit items; Math and QA resample their all-action test items.

Task	Treatment	Log est.	Log SE	Audit est.	Audit SE
Math	SR	−0.012	.010	−0.018	.009
	VER	−0.071	.018	−0.012	.017
	DEB	−0.029	.019	−0.049	.018
	PV	−0.046	.017	−0.038	.015
Code	SR	+0.003	.012	+0.011	.011
	VER	−0.054	.022	+0.022	.020
	DEB	−0.041	.021	+0.008	.014
	PV	−0.061	.020	+0.014	.013
QA	SR	+0.019	.010	+0.002	.009
	VER	+0.008	.011	−0.011	.010
	DEB	−0.058	.018	−0.047	.017
	PV	−0.030	.013	−0.014	.012

Table 15: Bootstrap SE ($B=1000$) for the contrasts in table 2. The cross-split logged-audit gaps for VERIFY on Math and Code exceed two combined bootstrap SE; among these two gaps, only Code reverses sign. This comparison does not distinguish treatment-selection effects from population shift.

Cross-split OPE-discrepancy bootstrap SE. Table 16 reports SEs for the cross-split discrepancies in table 3.

Task	DM	IPW	SNIPW	DR
Math	.005	.006	.005	.004
Code	.009	.010	.008	.007
QA	.006	.007	.005	.005

Table 16: Bootstrap SE ($B=1000$) for the cross-split OPE discrepancies in table 3; these include both estimator error and population shift.

Rosenbaum-style hidden-confounding sensitivity. Table 17 reports a Rosenbaum-style sensitivity analysis for the VERIFY advantage within a pre-specified high-difficulty Math subgroup (the top difficulty quartile), not for the overall fixed-policy contrast in table 2. Larger Γ corresponds to stronger unobserved assignment bias. The lower bound reaches zero near $\Gamma = 1.25$ and is negative by $\Gamma = 1.5$.

Γ	Lower	Point	Upper
1.00 (none)	+0.02	+0.04	+0.06
1.25	+0.00	+0.04	+0.08
1.50	−0.01	+0.04	+0.09
2.00	−0.04	+0.04	+0.12

Table 17: Rosenbaum-style sensitivity for VERIFY advantage in the pre-specified high-difficulty Math subgroup.

Behavior policy logging. During training, the mixture component is sampled as randomized probe with probability 15% and exploitation with probability 85%, with the behavior-policy floor $b_{\min} = 0.02$ enforced on every (x, k) . Regardless of the sampled component, causal estimators use the logged marginal $b_k(x) = 0.85\pi_k(x) + 0.15q_k(x)$; component-specific probabilities are diagnostic fields only. The exploitation policy assigns harder inputs (estimated from pre-treatment solo-probe entropy and difficulty labels) disproportionately to verification and debate, which can contribute to the sign differences in table 2; train-test population shift is an additional unresolved contributor in that cross-split comparison. The DR interpretation depends on logging this marginal correctly and on the routing-relevant pre-treatment features in table 9 being sufficient.

$b_{\min}$ sensitivity. Table 18 sweeps the behavior-policy floor $b_{\min}$. Lowering $b_{\min}$ to zero would in principle break overlap on expensive protocols on hard inputs, but the CARL policy is fairly stable across $b_{\min} \in \{0.01, 0.02, 0.05\}$ because the epistemic component of $\hat{\sigma}_k^{(D)}$ widens when local support is thin.

I Additional Experimental Results

I.1 Risk Parameter Sweep

Table 19 sweeps the LCB risk parameter κ . Larger κ reduces CHR and tokens but eventually lowers utility by abstaining too often. The validation-selected $\kappa = 1.0$ sits near the elbow.

$b_{\min}$	Util	CHR	Cov.
0.01	.58	.12	.87
0.02 (default)	.59	.10	.90
0.05	.59	.10	.91

Table 18: Sensitivity to the behavior-policy floor $b_{\min}$, averaged over Math and QA. Cov. is conditional selected-action realized-contrast coverage among non-solo selections.

κ	Util	CHR	Abst.	Tok./ep
0.0	.59	.22	.22	5.7K
0.5	.60	.14	.35	4.8K
1.0	.59	.10	.48	4.1K
1.5	.57	.06	.59	3.4K
2.0	.54	.03	.68	2.8K

Table 19: Effect of κ , averaged over Math and QA. $\kappa = 1.0$ is the validation-tuned default. At $\kappa = 0$, the policy is exactly the “Greedy, no LCB” ablation in table 4; both rows use the same cached predictions and therefore report the same utility and CHR.

I.2 Compute-matched Solo Comparison

Table 20 compares CARL with solo strategies that consume comparable token budgets but never invoke T_1 – T_4 . Reward-per-1K-tokens is a normalized efficiency measure.

Method	Util	Reward	Tok./ep	R/1K
Solo-SelfConsist.	.53	.58	4.2K	.138
Solo-Rerank	.54	.58	4.0K	.145
MasRouter+Solo	.55	.59	5.2K	.113
Adapt.+Abs.	.55	.59	5.1K	.116
CARL	.59	.62	4.1K	.151

Table 20: Compute-matched comparison, averaged over Math and QA. Reward is the arithmetic mean of the two task-level raw rewards in table 1; R/1K is recomputed from that mean and Tok./ep.

I.3 Selected Treatment Distribution

Table 21 reports the distribution of CARL-selected actions on the final test set.

I.4 Distribution Shift

Table 22 reports controlled distribution-shift diagnostics. These are finite-catalog stress tests and not guarantees under arbitrary deployment shift.

I.5 Protocol Drift

Table 23 reports treatment-drift stress tests in which protocol definitions or model behavior change after calibration. “No recal.” applies the original selector

Task	SOLO	SR	VER	DEB	PV
Math (MATH-500)	42%	12%	29%	7%	10%
Code (SWE Lite)	51%	14%	22%	4%	9%
QA (MuSiQue+2Wiki)	55%	13%	18%	6%	8%

Table 21: CARL selected treatment distribution. Code uses the 300-instance Lite test set.

Setting	MR+Solo		CARL	
	Util	CHR	Util	CHR
In-domain (Math)	.57	.17	.61	.08
Harder math	.51	.24	.55	.12
Longer SWE issues	.29	.29	.33	.14
Distractor-passage QA	.48	.26	.51	.13

Table 22: Distribution-shift evaluation. Math uses the hardest MATH quartile; Code uses longer SWE Lite issues with extended BM25 context; QA reorders the benchmark-provided passages and inserts additional in-dataset distractor passages, without external retrieval.

to the shifted protocol; “Recal.” refits a_k on 200 probe episodes from the shifted setting.

Shift	NR Util	NR CHR	RC Util	RC CHR
Verifier prompt wording	.55	.14	.57	.11
Judge prompt wording	.55	.13	.57	.11
Decoding temp. +0.2	.54	.15	.56	.12
Sibling checkpoint	.53	.17	.55	.13

Table 23: Treatment-drift stress tests, averaged over Math and QA.

I.6 Sequential Deployment

Table 24 reports a sequential deployment diagnostic on Math, showing how performance changes as probe-supported calibration data accumulate over 500 deployment episodes. The probe rate is gradually reduced after the initial collection; Cov. is realized-contrast predictive coverage with the non-solo conditional denominator defined in section 4.1.

I.7 Stress Tests

The main text mentions three post-routing stress settings: injected protocol failures (a fraction of collaborative protocol executions return errors and fall back to the solo answer), rate-limit-style missing responses (a fraction of agent calls return empty, simulating a busy serving stack), and adversarial distractor critiques (a synthetic verifier inserts plausible but incorrect critiques into the verifier protocol). Table 25 reports CARL utility and CHR

Episodes	Util	CHR	Cov.	PTC	Probe
1–100	.54	.19	.82	.73	25%
101–200	.56	.14	.85	.79	20%
201–300	.58	.12	.88	.81	17%
301–400	.59	.10	.90	.83	15%
401–500	.61	.08	.91	.86	15%

Table 24: CARL over 500 sequential deployment episodes on Math.

under each setting at a 10% per-call rate, averaged over Math and QA. Because these failures occur after routing and no service-health feature or recalibration signal is provided, the pre-routing $\hat{\sigma}_k^{(D)}$ does not widen in response. The table therefore measures degradation of the fixed selector and its explicit error-to-solo fallback, not adaptive robustness.

Setting	CARL Util	CARL CHR
No stress (default)	.59	.10
Injected protocol failures	.56	.11
Rate-limit missing responses	.57	.10
Adversarial distractor critiques	.55	.13

Table 25: CARL under three stress settings, averaged over Math and QA. Each stress is applied at a 10% per-call rate.

J Cross-Model Robustness

The main experiments use Qwen/Qwen3.5-27B throughout, which raises a reasonable concern: are CARL’s gains specific to this checkpoint, or do they extend to base models with different reasoning profiles? We report a controlled cross-model robustness study here. These results are exploratory: cross-model runs are not used for any main claim, the protocol catalog is held fixed across models, all hyperparameters (probe ratio, δ , κ , $b_{\min}$, γ , K_{NN}) are inherited from the Qwen-27B-tuned defaults rather than re-tuned per model, and each additional-model (model, task) cell is run with a single seed. The five-seed Qwen3.5-27B main results are not inserted as rows in the single-seed tables below; they remain in tables 1 and 6. Exploratory per-cell numbers should therefore be read as directional evidence rather than precise effect sizes.

J.1 Setup

We evaluate CARL with two additional base models drawn from different families and capability tiers: a smaller open-weight model

(Llama-3.1-8B-Instruct) and a larger open-weight model (Qwen2.5-72B-Instruct). We use Qwen2.5-72B rather than a larger Qwen3.5 variant because the Qwen3.5 family above 27B uses sparse/mixture-of-experts architectures (e.g., Qwen3.5-122B-A10B), which complicate the latency/token cost accounting underlying our utility definition. For each additional model we rerun the full CARL pipeline—behavior-policy logging with the same $\epsilon_{\text{probe}} = 0.15$, cross-fitted nuisance training, DR advantage estimation, and conformal calibration—using 30% of the original training logs per task together with the full calibration matrix (1,500/1,500/300 for Math/QA/Code). Logging is reduced because behavior-policy data collection dominates router training cost, while the calibration matrix is retained at full size to avoid conflating model-transfer effects with conformal-quantile noise. Each protocol’s prompts, roles, decoding parameters, and stopping rules are unchanged; only the checkpoint behind every protocol is swapped. The latency reference budget in Y is renormalized per model so that token and latency costs remain on the unit scale, which means absolute utility values are not directly comparable across models—only the within-model ranking of methods is meaningful. We compare against the two strongest non-CARL baselines from [table 1](#): MasRouter-style+Solo and DR-LCB (absolute utility).

J.2 Main Cross-Model Results

[Table 26](#) reports cost-aware utility and CHR for the two additional models on the three task families. Every row in this table uses the same reduced-log, single-seed protocol. The qualitative ranking $\text{CARL} \succeq \text{DR-LCB} \succeq \text{MR+Solo}$ holds within each (model, task) cell, with a utility margin over DR-LCB of 0.02–0.04 and a CHR reduction of 0.04–0.09. The margins contract on Qwen2.5-72B, where within-model SOLO performance is already high under its own normalization and the headroom for any collaborative protocol to add positive advantage is smaller; on Llama-3.1-8B, within-model utilities are uniformly lower and CHR uniformly higher than the corresponding Qwen-27B cells (subject to the per-model normalization caveat above), consistent with the noisier reward signal a weaker base model produces. Because each exploratory cell uses a single seed, its margins should be read as directional rather than as calibrated effect sizes; the qualitative ranking is the load-bearing claim, not the specific decimal values.

J.3 Per-Model Recalibration

A more pointed concern is whether CARL’s calibrated lower bound can be recalibrated after the base model changes. This is not zero-shot calibration transfer: a_k is refit separately for every model. We report conditional selected-action coverage and realized-contrast ECE for each model in [table 27](#). Conditional coverage falls within 2–6 percentage points of the 90% reference level, with the largest undercoverage on Llama-3.1-8B for Code (0.84), where the reduced training-log budget and the higher reward noise of the weaker base model together produce the widest residual distribution. On Math and QA, Llama-3.1-8B undercoverage is 2–4 points, and on Qwen2.5-72B coverage is essentially at the target. ECE remains below 0.10 except for Llama-3.1-8B on Code (0.11), consistent with the same noise pattern. These results suggest that the LCB construction adapts to noisier residuals by widening a_k rather than silently violating coverage, but the Code/Llama-8B cell indicates the adaptation is imperfect under reduced logs and serves as a useful failure mode to flag.

J.4 Selected-Action Distribution Shifts

The CARL-selected action mix shifts with model capability in a way that is qualitatively consistent with the collaboration-advantage estimand. On Llama-3.1-8B, SOLO is selected on 31% of Math instances (compared with 42% for the Qwen3.5-27B main study and 55% for Qwen2.5-72B), and verification-style protocols take more mass, reflecting the larger upside of an independent checker when the base solver is weaker. On Qwen2.5-72B, SOLO dominates further on Math and QA (55% and 63%), and CHR drops correspondingly, since the headroom for any collaborative protocol to add positive advantage is smaller. [Table 28](#) reports the full breakdown. The direction is what the estimand predicts: as $m_0^*(x)$ increases, $\tau_k(x)$ shrinks for most k and the LCB policy abstains more often. We do not claim this monotone shift is a calibrated quantitative prediction, only that the qualitative direction is consistent.

J.5 Discussion and Limitations

Three observations are worth recording. First, the ranking $\text{CARL} \succeq \text{DR-LCB} \succeq \text{MR+Solo}$ is preserved across the exploratory model–task cells, but the full-log Qwen3.5 reference is not a matched cross-model replicate. Second, the per-model re-

Model	Method	Math Util/CHR	Code Util/CHR [†]	QA Util/CHR
<i>Exploratory: 30% logs, one seed per cell</i>				
Llama-3.1-8B	MR+Solo	.47 / .22	.21 / .31	.43 / .26
	DR-LCB	.49 / .18	.23 / .25	.45 / .22
	CARL	.53 / .11	.26 / .16	.48 / .15
Qwen2.5-72B	MR+Solo	.63 / .14	.38 / .19	.58 / .16
	DR-LCB	.64 / .12	.39 / .15	.59 / .13
	CARL	.66 / .07	.41 / .10	.61 / .09

[†]Code Util is on the full 300-instance Lite test split; Code CHR is on the 100-instance audit subset.

Table 26: Cross-model robustness results. Each cell shows cost-aware utility and harmful-collaboration rate. Absolute utility values are not directly comparable across models because the latency reference budget is renormalized per model; the within-model method ranking is the intended comparison. All rows use 30% logs and one seed; the five-seed Qwen3.5 main study is reported separately in [table 1](#).

Model	Math		Code		QA	
	Cov	ECE	Cov	ECE	Cov	ECE
<i>Exploratory: 30% logs, one seed per cell</i>						
Llama-3.1-8B	.88	.07	.84	.11	.86	.08
Qwen2.5-72B	.91	.04	.88	.07	.90	.05

Table 27: Per-model recalibration results. All rows use 30% logs and one seed. Coverage is conditional selected-action realized-contrast predictive coverage among non-solo selections; ECE compares predicted mean advantage with paired realized contrasts and is not CATE coverage.

Model	Task	SOLO	SR	VER	DEB	PV
Llama-3.1-8B	Math	31%	15%	36%	8%	10%
	Code	39%	16%	27%	6%	12%
	QA	42%	16%	24%	8%	10%
Qwen2.5-72B	Math	55%	10%	22%	5%	8%
	Code	60%	12%	18%	3%	7%
	QA	63%	11%	16%	4%	6%

Table 28: CARL-selected action distribution by model. The Qwen3.5-27B distribution is in [table 21](#). SOLO selection rises with base-model capability, consistent with shrinking collaboration advantage as the solo solver improves.

calibration diagnostics stay within a plausible band even when the base model is markedly weaker, except on the Llama-3.1-8B Code cell where under-coverage reaches 6 points and ECE reaches 0.11; we flag this rather than claim universal calibration robustness. Third, the action mix shifts in the direction the collaboration-advantage estimand predicts: stronger models abstain more, weaker models collaborate more.

Several limitations apply specifically to this section. (i) We hold the protocol catalog fixed across models. In principle a 72B verifier paired with

an 8B solver might shift the advantage structure substantially, and we do not study heterogeneous-model catalogs here. (ii) Hyperparameters are inherited from the Qwen3.5-27B defaults; some per-model tuning would likely improve absolute numbers, especially for Llama-3.1-8B. (iii) Cross-model runs use 30% of the training logs and a single seed per (model, task) cell, so the per-cell numbers are not accompanied by bootstrap standard errors and should be read as directional evidence rather than as precise effect sizes. We do not report significance tests for this appendix-only single-seed study; [table 26](#) is intended only for within-model method comparisons. Any qualitative comparison with the five-seed Qwen3.5-27B study must consult [table 1](#) separately and is not a matched estimate. (iv) The latency reference budget is renormalized per model, so absolute utility values are not comparable across the two model blocks in [table 26](#), or with the main Qwen3.5-27B study; only the within-model method ordering is. (v) The numbers in [tables 26 to 28](#) come from cross-model runs with the reduced-log budget described above; they are not a substitute for re-running the main experiments under each base model with full training data and multiple seeds. For this reason, no numerical robustness claim from cross-model results enters the main text.

K Hindsight-Oracle Gain Convention

The main text reports hindsight-oracle recovery per task. To keep the population fixed, Code uses audit-only Solo, CARL, and hindsight-oracle utilities here. The resulting audit-aligned pooled cross-task average (which therefore differs from the full-test

Avg. column in table 1) is

$$\frac{U_{\text{CARL,avg}} - U_{\text{Solo,avg}}}{U_{\text{Hindsight,avg}} - U_{\text{Solo,avg}}} = \frac{0.503 - 0.420}{0.573 - 0.420} \approx 54\% . \quad (15)$$

This descriptive ratio summarizes how much of the hindsight-oracle improvement over solo is recovered under the validation-tuned cost coefficients; it is not online observable. The per-task values are in table 29.

Task	Solo	CARL	Hindsight	Recov.
Math (MATH-500)	.53	.61	.68	53%
Code audit (100)	.25	.34	.44	47%
QA (MuSiQue+2Wiki)	.48	.56	.60	67%

Table 29: Per-task hindsight-oracle recovery on a common population; Code is audit-only.

L Additional Theoretical Details

Proposition 3 (Auxiliary greedy-policy advantage-error bound). *Under finite $\mathcal{T}$, bounded outcomes, i.i.d. inputs, and $\sup_{x,k} |\tilde{\tau}_k(x) - \tau_k(x)| \leq \epsilon$ for all $k \geq 1$, the greedy one-vs-solo policy $\hat{\pi}$ satisfies $J(\pi^*) - J(\hat{\pi}) \leq 2\epsilon$.*

This auxiliary result applies to the greedy plug-in policy that maximizes $\tilde{\tau}_k(x)$ with solo fixed at zero. It does not directly bound the deployed LCB policy in equation (13), whose additional predictive pessimism depends on $\kappa a_k \hat{\sigma}_k^{(D)}(x)$.

Proof. For any input x , let $a^*(x)$ be the mean-optimal action and $\hat{a}(x)$ the greedy action under the estimated advantages. Since T_0 has advantage zero by definition and every collaborative advantage is estimated within ϵ , the estimated advantage of $\hat{a}(x)$ is at least that of $a^*(x)$. The true advantage of $\hat{a}(x)$ is therefore at most 2ϵ below that of $a^*(x)$. Taking expectations over X gives the result. $\square$

Corollary 4 (Compute spent on harmful collaboration). *Under the realized-contrast calibration condition in proposition 2, the probability of selecting a collaborative protocol whose realized utility is below its paired solo utility is at most $\sum_{k=1}^K \delta_k$. If the maximum excess compute cost of any collaborative protocol over solo is $C_{\max}$, the expected excess compute spent on such harmful realized executions is at most $C_{\max} \sum_{k=1}^K \delta_k$.*

This corollary inherits the limitations of proposition 2: it is a calibration-dependent diagnostic, not a distribution-free safety guarantee.

M Reproducibility Details

For reproducibility, the experiment records include protocol manifests, prompt templates, treatment hashes, split identifiers, exact model identifiers, decoding parameters, usage logs, and SWE-bench Lite harness invocations. Public benchmark data are obtained from their original sources and are not redistributed. Cross-model runs are marked exploratory and recorded with their request dates and model-specific configurations.

N Compute Resources

The experiments do not train or fine-tune the base language model. Compute is dominated by inference-time protocol executions and, for Code, by SWE-bench Lite harness runs. Main experiments use local vLLM serving with a fixed GPU configuration, and token counts are logged for every protocol execution. Cost-aware utility is computed from logged token counts and latency using the paper’s coefficients λ and μ .

O Broader Impacts

CARL may reduce unnecessary LLM calls by abstaining from collaboration when the estimated advantage is not positive, which can lower cost, latency, and energy use. It may also improve reliability by making harmful collaboration measurable and by exposing calibration diagnostics. Potential negative impacts include overreliance on calibrated lower bounds, deployment in high-stakes settings without task-specific review, and degraded behavior under distribution shift or treatment drift. Mitigations include conservative abstention thresholds, audit logs, calibration monitoring, task-specific validation, human review in high-stakes settings, and recalibration when protocol definitions or model versions change.

P Asset Licenses and Terms

The paper uses public benchmarks, open-weight model checkpoints, and standard software libraries. MATH, MATH-500, SWE-bench Lite, MuSiQue, and 2WikiMultiHopQA are cited in the main paper and used according to their public release terms. The model checkpoint, software dependencies, prompts, manifests, split identifiers, and evaluation settings are identified in the experiment records. Benchmark assets are obtained from their original sources in accordance with the corresponding licenses and terms of use.