BID-SPO: Rethinking Multi-Agent Prompt Optimization as Information Contracts via Bidding-based Shapley Values

Anonymous ACL submission

Abstract

Multi-agent LLM systems decompose complex tasks across specialist agents, yet joint prompt optimization remains difficult because agents are coupled through structured information handoffs. We identify *interface mismatch* as a structural failure mode that existing optimizers cannot diagnose: their scalar attribution signals conflate reasoning quality with format incompatibility, and the inter-agent contract is left entirely implicit. We propose **BID-SPO** (*Bidding-based Shapley Prompt Optimization*), which makes the inter-agent contract an explicit optimization variable. Each agent declares an *Output Bid* and an *Input Demand*, and a contract-aware cooperative-game characteristic function yields an attribution signal whose Shapley decomposition exposes three interpretable dimensions (global task contribution, downstream interface alignment, and upstream adaptation cost), turning attribution into a targeted diagnosis for prompt repair. A bid-conformance constraint keeps declared contracts feasible with respect to actual agent behavior throughout optimization. Across seven benchmarks and different model configurations, BID-SPO outperforms MASPO and Agent-Dropout, with accuracy gains of $+2.93\%$ under the Qwen3-8B + Gemini-2.5-pro configuration and $+4.38\%$ under the DeepSeek-v4 flash / pro configuration. Our findings emphasize that reliable multi-agent LLM optimization requires optimizing inter-agent interfaces as explicit objects, not just improving agents in isolation.

1 Introduction

Large language models are increasingly deployed as multi-agent systems (MAS), where specialist agents collaborate through structured information handoffs to tackle tasks too complex for a single model (Hong et al., 2024; Wu et al., 2024). Each agent's system prompt effectively *programs* its role, yet agents are not independent: the output of one agent is the input of the next. A change to any agent therefore propagates downstream, and automated prompt optimization has emerged as a core research priority (Wang et al., 2026; Zhang et al., 2026; Zhou et al., 2026a; Wang et al., 2025; Hong et al., 2026), complementing single-agent compilers such as DSPy (Khattab et al., 2024), TextGrad (Yuksekgonul et al., 2024), and OPRO (Yang et al., 2024).

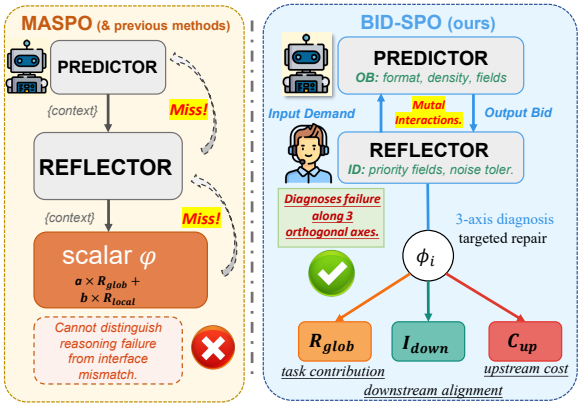


Figure 1: **BID-SPO vs. scalar-attribution prior work.** *Left:* scalar attribution (one score). *Right:* BID-SPO uses a contract-aware characteristic function whose Shapley attribution exposes contribution, interface alignment, and adaptation cost.

We argue that the central obstacle in MAS prompt optimization is not the quality of each agent's individual reasoning, but the alignment of *information interfaces* between adjacent agents. A prompt implicitly defines the format, granularity, and semantic fields an agent will emit. As depicted in Figure 1 (left), consider a PREDICTOR $\rightarrow$ REFLECTOR chain on a mathematical reasoning task: if the Predictor is prompted to produce a compact narrative while the Reflector expects a step-by-step derivation ending with a boxed answer, the Reflector spends its context budget searching for structure that was never produced, silently degrading end-to-end accuracy even when each agent looks competent in isolation. This *interface mismatch* is endemic to any pipeline in which one

agent's output is another's input, and it compounds with graph depth.

Existing prompt optimizers cannot diagnose *interface mismatch* because their evaluation signals collapse it into a scalar. Both MASPO (Wang et al., 2026) and the concurrent MAPRO (Zhang et al., 2026) score each agent with a composite of task accuracy and local output quality, but neither signal can distinguish *why* an agent underperforms: poor core reasoning versus a format that downstream agents cannot efficiently consume. Single-stage compilers such as DSPy and TextGrad inherit this limitation by treating modules as essentially independent. As a consequence, a prompt can achieve a high composite score while still imposing costly adaptation burdens on its neighbours, a failure mode the optimizer never sees. The inter-agent contract is left entirely implicit.

We recast MAS prompt optimization through explicit information contracts. Each agent is simultaneously a *provider* (declaring an *Output Bid*: a structured commitment specifying what it will produce) and a *receiver* (declaring an *Input Demand*: what it requires from predecessors). MAS failures arise when providers and receivers cannot agree on a common format or granularity. This view motivates treating each agent's contract as an *explicit optimization variable*, with an attribution signal that explicitly accounts for interface friction alongside raw task contribution.

We instantiate this view in **BID-SPO** (*Bidding-based Shapley Prompt Optimization*). As shown in Figure 1 (right), each agent carries a triple (t_i, b_i, d_i) (system prompt, Output Bid, and Input Demand) jointly co-optimized through coordinated prompt refinement without any weight update. The cooperative-game characteristic function is redesigned to be contract-aware and agent-indexed, augmenting accuracy-only attribution with explicit interface externalities. Its resulting Shapley attribution exposes three interpretable dimensions: global task contribution, downstream interface alignment, and upstream adaptation cost. This diagnosis directs the optimizer to the precise component of the triple that needs repair. A lightweight bid-conformance feasibility constraint prevents the optimizer from inflating contracts beyond an agent's actual behavior.

Our primary contributions are as follows:

- We identify *interface mismatch* as a structural failure mode of MAS prompt optimization that existing scalar attribution signals cannot diagnose, and argue that the inter-agent contract must be an explicit, optimizable object rather than an implicit byproduct of each agent's prompt.
- To mitigate these challenges, we propose a contract-aware cooperative-game characteristic function whose Shapley attribution separates each agent's contribution into three interpretable marginals (global contribution, downstream alignment, and upstream cost), recovering classical accuracy-based Shapley as a degenerate special case and enabling targeted, component-aware prompt repair.
- We introduce **BID-SPO**, comprising three mutually reinforcing modules: INFOCON (information contracts via Output Bids and Input Demands), DECSHAP (decomposable Shapley attribution), and ADVBC (a bid-conformance feasibility constraint) jointly optimizing each agent's triple (t_i, b_i, d_i) through coordinated prompt refinement.
- Extensive experiments across seven benchmarks demonstrate that BID-SPO consistently outperforms MASPO and other strong baselines, with Avg_6 gains of $+2.93\%$ and $+4.38\%$, and single-task improvements of up to $+13.40\%$ on MATH-500 under the DeepSeek-v4 configuration.
- We show empirically that explicit contracts plus decomposable attribution generalize across model families: the optimized contracts transfer cleanly to held-out worker LLMs, suggesting they encode model-agnostic interface preferences rather than provider-specific artefacts.

2 Preliminary Analysis

We conducted three pilot experiments, each exposing a distinct structural failure of scalar attribution in MAS prompt optimization, which collectively motivates BID-SPO. Findings are summarised in Figure 2 and Table 2; per-topology and per-difficulty breakdowns appear in Appendix Q.

Hypothesis 1 (Necessity of Information Contracts)

Without explicit, declared interface contracts between adjacent agents, attribution signals conflate *reasoning failure* with *interface mismatch*: a sound agent producing the wrong format is penalised identically to a faulty agent producing the right format.

We test this with a PREDICTOR $\rightarrow$ REFLECTOR pipeline (controlled) under three framing-only au-

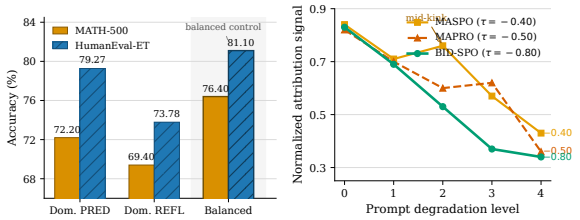


Figure 2: **Preliminary pilot diagnostics.** (left) **One-Voice Dominance.** Authority skew at the prompt level degrades pipeline accuracy. (right) **Attribution Signal Stability.** Kendall τ between an agent's prompt-degradation level and its attribution signal[1].

Condition	MATH-500	HumanEval-ET	Tension
Dominant PRED	72.20	79.27	0.18
Dominant REFL	69.40	73.78	0.71
Balanced (control)	**76.40**	**81.10**	0.38

Table 1: Numerical breakdown of the three authority configurations in Figure 2 (left), including the authority-conflict statistic *Tension*.

thority configurations: *Dominant PRED*, *Dominant REFL*, and *Balanced*. Existing MAS optimizers communicate through unstructured `{context}` injection and score agents on accuracy alone, so this authority skew is invisible to their objective. Figure 2 (left) visualizes the resulting accuracy drop, while Table 1 provides the exact numeric breakdown and reports *Tension*, an automatically estimated authority-conflict score in $[0, 1]$. Both skewed settings underperform the balanced control by 4–7%, yet a scalar attribution signal cannot determine the damage comes from *reasoning* loss or *interface authority* imbalance, which requires distinct repairs (Appendix Q.7).

Hypothesis 2 (Necessity of Decomposable Attribution)

A scalar composite of local and global signals cannot diagnose *why* an agent under-performs, only *that* it does. A decomposition along global contribution, downstream alignment, and upstream cost is required to drive targeted updates.

To measure residual interface mismatch *independently* of task accuracy, we use an automatic evaluator to estimate each upstream: downstream output pair on a normalized compatibility scale in $[0, 1]$ and report the fraction below 0.5. MASPO and MAPRO still leave the mismatch rate above 19% after six rounds, showing that accuracy-only optimization leaves residual interface friction unattributed (Appendix Q.7).

[1]Higher-magnitude negative values indicate stronger monotonic correlation with true quality. BID-SPO's decomposed ϕ_i is the only near-monotone signal.

Stage	MATH-500	AQuA
Initial prompts (no opt.)	38.5%	41.2%
After MASPO (6 rounds)	22.4%	26.8%
After MAPRO (6 rounds)	19.7%	20.5%
After **BID-SPO** (6 rounds)	**4.3%**	**6.1%**

Table 2: **Interface Mismatch Persistence.** Fraction of agent pairs with estimated compatibility below 0.5. MASPO and MAPRO leave a substantial residual mismatch despite improving task accuracy; BID-SPO's explicit contracts collapse it to single digits.

Hypothesis 3 (Necessity of Honesty Enforcement)

When the optimizer is rewarded for high-scoring Output Bids, it has an incentive to over-state bid claims relative to actual agent behavior. A conformance constraint is required to keep contracts and behavior aligned across optimization rounds.

Starting from a converged MAS prompt, we construct five degradation levels and test how monotonically each evaluation signal tracks the induced accuracy drop. Kendall τ summarizes monotonicity. BID-SPO's ϕ_i ($\tau = -0.80$) is the only near-monotone signal, while MASPO and MAPRO both exhibit non-monotone kinks. This shows both decomposable attribution and a separate conformance constraint; the two target distinct failure modes.

Motivated by these pilots, we develop three components of BID-SPO in Section 4; the details of hypothesis-to-component is in Appendix Q.7.

3 Related Work

The closest prior work is MASPO (Wang et al., 2026), which jointly optimizes system prompts across a structured MAS using a composite local/global score and topology-aware beam search. The concurrent MAPRO (Zhang et al., 2026) recasts the same problem as MAP inference over candidate prompts. Other recent work optimizes graph topology, routing, or initialization while keeping the per-agent evaluation signal scalar (Zhou et al., 2026a; Yue et al., 2025; Tian et al., 2025; Wang et al., 2025; Hong et al., 2026). Single-agent compilers such as DSPy (Khattab et al., 2024), TextGrad (Yuksekgonul et al., 2024), and OPRO (Yang et al., 2024) treat modules as essentially independent and provide no mechanism for interface friction. Communication-compression frameworks such as Cut-the-Crap (Zhang et al., 2025) and Latent Collaboration (Zou et al., 2026)

implicitly learn or compress message passing but expose no contract that an optimizer can inspect or modify. Across all these lines, the inter-agent contract remains implicit, attribution collapses to a scalar, and there is no mechanism to distinguish interface mismatch from reasoning failure, precluding targeted, component-aware prompt repair.

4 Bidding-based Shapley Prompt Optimization (BID-SPO)

4.1 Problem Formulation

We model a multi-agent system as a directed acyclic graph $G = (V, E)$ with N agents. Each node $i \in V$ is an LLM agent with a system prompt template t_i, and an edge $(i, j) \in E$ means agent i's output is appended to agent j's context at inference time. Given a query q, the MAS executes by traversing G in topological order and the final agent emits the system answer o_{final}, scored against a ground truth y^* via $R(o_{\text{final}}, y^*) \in \{0, 1\}$.

BID-SPO augments each agent with two short textual contracts: an **Output Bid** b_i and an **Input Demand** d_i. The joint optimization objective is

$$\max_{\{(t_i, b_i, d_i)\}_{i=1}^{N}} \mathbb{E}_q[R(\Phi(G, \{t_i, b_i, d_i\}, q), y^*)], \tag{1}$$

where Φ is the full MAS execution function and the free variables are the discrete text strings $\{(t_i, b_i, d_i)\}_{i=1}^{N}$. No model weights are trained; optimization operates entirely in prompt space. Operational details of the Phase 1 optimization and Phase 2 evaluation pipeline, including the *Phase-2 worker* and evaluation/update-backend terminology, are deferred to Appendix J.3.

4.2 Information Contracts

In conventional multi-agent prompting, the information exchanged between agents is governed by an implicit contract, which obscures whether an error arises from poor reasoning or from an output that a downstream agent cannot parse or use. BID-SPO makes this contract explicit by attaching two compact, optimizable interface descriptors to each agent's system prompt: an Output Bid b_i and an Input Demand d_i.

The Output Bid b_i specifies what the agent will deliver, including the expected form and content of its response, while the Input Demand d_i specifies what the agent requires from upstream messages, including salient fields and tolerable noise. Jointly, b_i and d_i turn latent compatibility constraints into optimizable interface descriptors, allowing format mismatches to be distinguished from genuine reasoning failures. Concrete contract templates and the initialization procedure are given in Appendix B; the inference-time formatting used to append them to the system prompt is given in Appendix L.7.

4.3 Decomposable Cooperative Attribution

Contracts alone do not determine how to improve them: a scalar attribution signal cannot distinguish whether a low-scoring agent suffers from weak reasoning, a format its successor cannot parse, or an overly strict dependency on upstream context. BID-SPO addresses this by introducing a contract-aware, agent-indexed characteristic function. The resulting Shapley attribution separates three distinct sources of agent performance: task contribution, downstream interface alignment, and upstream adaptation cost. For agent i and a coalition $S \subseteq V$, BID-SPO defines the agent-indexed feature function

$$v_i(S) = R_{\text{glob}}(S) + \lambda_1 \cdot I_{\text{down}}(i, S) - \lambda_2 \cdot C_{\text{up}}(i, S), \tag{2}$$

whose three components carry distinct meanings: $R_{\text{glob}}(S)$ is the system's global task accuracy when agents in S use their current prompts and the rest are replaced by a fixed degraded fallback (the null-player surrogate); $I_{\text{down}}(i, S)$ measures how well agent i's Output Bid satisfies its downstream neighbours' Input Demands; and $C_{\text{up}}(i, S)$ measures the adaptation cost imposed on agent i by its upstream predecessors. I_{down} is derived from structured downstream compatibility-violation estimates, while C_{up} is derived from structured upstream adaptation-burden estimates; both are then aggregated into normalized scalar terms.

When $\lambda_1 = \lambda_2 = 0$, Equation 2 reduces to pure accuracy-based Shapley attribution: BID-SPO strictly generalises this classical view by giving the optimizer two additional, interface-aware channels.

Shapley value and its decomposition. The Shapley value of agent i under v_i (Shapley, 1953),

$$\phi_i = \sum_{S \subseteq V \setminus \{i\}} \frac{|S|!\,(N - |S| - 1)!}{N!} \big[v_i(S \cup \{i\}) - v_i(S)\big], \tag{3}$$

is approximated by Monte Carlo sampling over sampled permutations of V. The decomposition

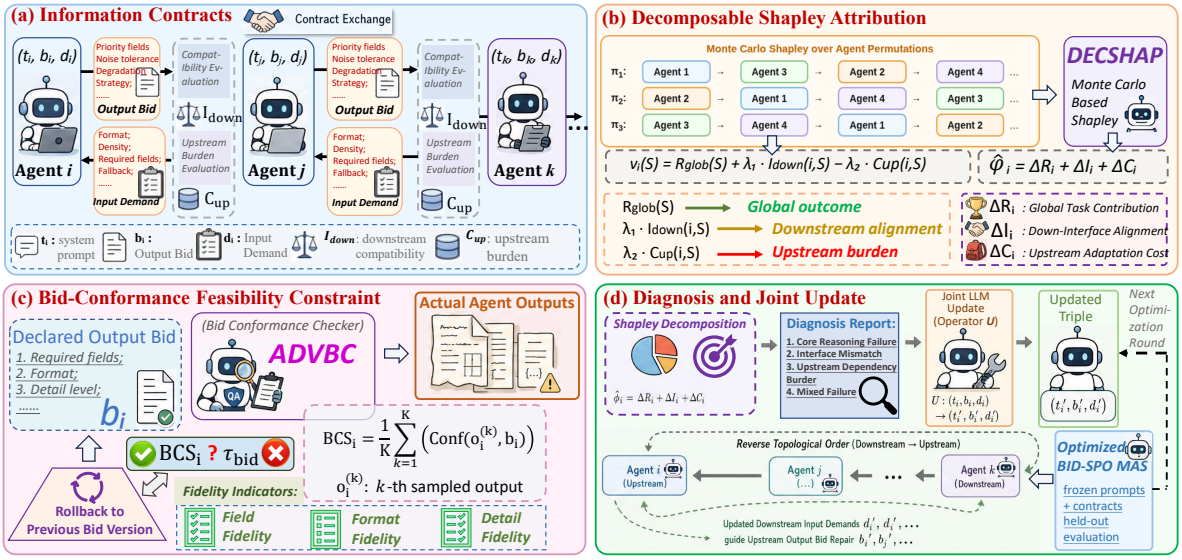


Figure 3: **BID-SPO architecture.** (a) Each agent carries a triple (t_i, b_i, d_i); Output Bids and Input Demands are exchanged across edges and converted into the corresponding I_{down} and C_{up} estimates. (b) Monte Carlo Shapley sampling yields $\hat{\phi}_i$ together with its three additive components. (c) A bid-conformance feasibility constraint filters infeasible bid updates whose declared contracts are not realized by actual outputs; such bids are rolled back and flagged for diagnosis. (d) The decomposed attribution drives joint updates of (t_i, b_i, d_i) in reverse topological order.

itself follows from Shapley linearity; the key modeling contribution lies in the contract-aware characteristic function in Equation 2, which augments accuracy-only attribution with explicit interface externalities. The linearity of v_i in its three components carries through the estimator, so the resulting attribution decomposes additively:

$$\hat{\phi}_i \;=\; \Delta R_i + \Delta I_i + \Delta C_i, \tag{4}$$

where each marginal is the average contribution of the corresponding component across permutations. Under this contract-aware characteristic function, the resulting decomposition turns attribution from a verdict into a diagnosis: ΔR_i flags weak reasoning, ΔI_i flags interface mismatch, and ΔC_i flags upstream-dependency burden. Formal definitions of the three components, the Monte Carlo estimator, the structured aggregation rules, and the theoretical justification are deferred to Appendix C and D.

4.4 Contract-Guided Optimization

The decomposed Shapley signal drives targeted, joint updates of each agent's triple (t_i, b_i, d_i), replacing the scalar-only prompt updates of prior work. Full details are in Appendix E–H.

Optimization pipeline. Each round executes three operations in sequence. For each agent, *Step 1* maps the Shapley decomposition $(\Delta R_i, \Delta I_i, \Delta C_i)$ to a structured natural-language diagnosis Diag_i and independently evaluates bid feasibility via BCS_i. *Step 2* then jointly updates (t_i, b_i, d_i) in reverse topological order, guided by Diag_i and the downstream Input Demands $\{d_j\}_{j \in \mathcal{N}_{\text{out}}(i)}$:

Concretely, $\text{Diag}_i = \mathcal{F}(\Delta R_i, \Delta I_i, \Delta C_i)$ and $\text{BCS}_i = \frac{1}{K}\sum_{k=1}^{K} \text{Conf}\Big(o_i^{(k)}, b_i^{(r)}\Big)$.

$$\begin{aligned}&\big(t_i^{(r+1)},\, b_i^{(r+1)},\, d_i^{(r+1)}\big) = \\ &\mathcal{U}\Big(t_i^{(r)},\, b_i^{(r)},\, d_i^{(r)},\, \text{Diag}_i,\, \{d_j\}_{j \in \mathcal{N}_{\text{out}}(i)}\Big)\end{aligned} \tag{5}$$

Here $o_i^{(k)}$ is the k-th recent output of agent i, $b_i^{(r)}$ the Output Bid at round r, and $\mathcal{N}_{\text{out}}(i)$ the set of agent i's direct downstream successors. $\mathcal{F}$ maps the decomposed Shapley marginals to a natural-language failure-mode tag (*interface-mismatch*, *upstream-dependency*, *core-reasoning*, or *mixed*). BCS_i acts as a feasibility constraint on declared bids: it measures whether recent outputs reliably realize the declared contract. When $\text{BCS}_i < \tau_{\text{bid}}$, the bid update is treated as infeasible, rolled back, and a conformance-failure flag is appended to Diag_i, preventing aspirational contracts from inflating I_{down}. $\mathcal{U}$ is the joint LLM update operator: it refines (t_i, b_i, d_i) *jointly* in reverse topological order, so each upstream update already conditions on the updated downstream Input Demands.

5 Experiments

We design experiments to answer three questions:

(Q1) *Does BID-SPO outperform MASPO and other strong baselines under same settings?*
(Q2) *Does the improvement carry over when the entire pipeline is moved to a different model provider, i.e., are contracts and decomposable attribution not artefacts of a single model family?*
(Q3) *Are the optimized contracts portable across heterogeneous Phase-2 worker LLMs?*

5.1 Experimental Setup

Models and Benchmarks. We evaluate two complementary model configurations. The first uses Qwen3-8B as worker and Gemini-2.5-pro as the evaluation-and-update backend. The second replaces both ends with the DeepSeek-v4 series: DeepSeek-v4-flash as worker and DeepSeek-v4-pro as the evaluation-and-update backend. We test on seven benchmarks: MATH-500 (Lightman et al., 2024), AGIEval-MATH (Zhong et al., 2024), AQuA (Ling et al., 2017), AIME 2025 (Sun et al., 2025), GPQA-Diamond (Rein et al., 2024), MBPP (Austin et al., 2021), and HumanEval-ET (Chen et al., 2021) (See details in Appendix N).

Baselines. We compare against three families: (i) single-model prompting: Vanilla, CoT (Wei et al., 2022), Self-Consistency (Wang et al., 2023), Self-Refine (Madaan et al., 2023); (ii) multi-agent designs without prompt optimization: Agent-Dropout (Wang et al., 2025), Sequential MAS, Hierarchical MAS; and (iii) MAS prompt optimization: TPE, SPO, and MASPO (Wang et al., 2026). Baselines are detailed in Appendix K.

Implementation Details. BID-SPO uses default settings: a fixed pool of $k = 50$ training queries per dataset, mini-batches sampled per round, and Monte Carlo Shapley sampling with a budget that drives the per-agent standard error below 0.05. Test-time inference uses the reflect graph topology with $N_r = 2$ rounds. All hyperparameters were fixed on a held-out validation slice of MATH-500 and never tuned on test sets. The complete hyperparameter table is in Appendix I, and the verbatim prompt templates (initialization, compatibility assessment, conformance assessment, diagnosis generation, and joint update) are in Appendix L.

5.2 Main Results

Direct comparison with MASPO. Table 3 reports accuracy under the Qwen3-8B + Gemini-2.5-pro configuration. The table places **BID-SPO (default)** and four ablation rows alongside the comparison baselines, including the strongest MASPO-based variant (*Hierarchical MAS + MASPO*). BID-SPO improves over this baseline on every dataset, with $+6.20\%$ on MATH-500, $+2.74\%$ on AGIEval, and $+3.05\%$ accuracy on HumanEval-ET, lifting the 6-dataset average from 71.01 to $\mathbf{73.94}$. On AIME 2025 (absent from the original MASPO paper), BID-SPO reaches 23.33% versus 20.00% for the strongest baseline.

Module ablations. The four ablation rows below **BID-SPO (default)** correspond to the three component interventions and their combination: $-$**InFoCon** removes the information contracts: b_i, d_i are emptied and the interface terms of v_i vanish; $-$**DecShap** removes the contract-aware interface terms from the characteristic function, reducing it to pure accuracy-based Shapley attribution; $-$**AdvBC** removes the bid-conformance feasibility constraint: declared bids are free to drift from actual outputs; $-$**(InFoCon+DecShap+AdvBC)** removes all three components.[2] The precise code-level details are in Appendix K.7.

Reverting to the accuracy-only characteristic function hurts most on hard math (MATH -3.40%, AIME -6.66%), where interface-aware attribution matters most; removing contracts is most damaging on AIME (-3.33%) and still noticeably hurts strict-output code generation (HumanEval-ET -2.44%); removing the conformance constraint has the smallest effect but produces two cells where ablated variant slightly exceeds the default (AQuA $+0.39\%$, GPQA $+0.51\%$), which is consistent with ordinary run-to-run variation rather than a systematic gain.

Cross-provider generalization (DeepSeek-v4). Table 4 verifies that BID-SPO is not specific to a single model stack: switching the entire pipeline to DeepSeek-v4 preserves a clear advantage over MASPO and a similar ablation profile. BID-SPO (default) reaches an average of $\mathbf{65.91}\%$ versus 61.53% for MASPO on DeepSeek-v4 ($+4.38\%$), larger than the $+2.93\%$ gap in the first configuration. The module pattern also remains directionally stable: removing AdvBC is still the mildest average drop (-0.22% on Avg_6), whereas removing all three components causes the largest degradation (-6.96%).

Cross-model transferability of optimized contracts. We further ask whether the optimized agent

[2] The $-$(InFoCon+DecShap+AdvBC) row removes all three BID-SPO modules, corresponding to a standard MAS running on the initial static prompts from round $r = 0$ with no prompt optimization applied.

Method	MATH-500	AGIEval	AQuA	AIME 2025	GPQA	MBPP	HE-ET	Avg_6
Vanilla	74.80	55.86	79.53	10.00	45.96	63.47	71.95	65.26
CoT (Wei et al., 2022)	75.40	54.69	81.89	13.33	46.97	64.17	72.56	65.95
SC (CoT) (Wang et al., 2023)	75.60	56.64	82.28	13.33	47.47	64.40	72.56	66.49
Self-Refine (Madaan et al., 2023)	76.20	56.64	82.28	13.33	47.98	64.17	70.73	66.33
AgentDropout (Wang et al., 2025)	76.80	59.77	86.22	16.67	47.98	58.08	65.85	65.78
Sequential MAS	75.20	59.38	83.46	10.00	47.98	57.38	68.90	65.38
+ TPE	72.40	58.59	85.04	10.00	47.98	58.08	65.24	64.56
+ SPO	75.80	60.16	86.61	13.33	49.49	64.40	71.95	68.07
+ MASPO (Wang et al., 2026)	77.20	62.11	87.01	16.67	54.04	65.34	76.83	70.42
Hierarchical MAS	77.60	59.38	81.10	13.33	49.49	61.36	70.12	66.51
+ TPE	77.60	60.55	85.43	13.33	50.51	63.47	73.78	68.56
+ SPO	77.80	63.28	86.61	16.67	51.01	61.83	73.17	68.95
+ MASPO (Wang et al., 2026)	78.40	64.45	87.01	20.00	54.04	65.34	76.83	71.01
BID-SPO (default, ours)	**84.60**	**67.19**	**88.19**	**23.33**	**56.57**	**67.21**	**79.88**	**73.94**
– InFoCon	82.80 1.80% ↓	66.02 1.17% ↓	86.22 1.97% ↓	20.00 3.33% ↓	54.55 2.02% ↓	65.81 1.40% ↓	77.44 2.44% ↓	72.14 1.80% ↓
– DecShap	81.20 3.40% ↓	66.41 0.78% ↓	87.40 0.79% ↓	16.67 6.66% ↓	55.05 1.52% ↓	64.40 2.81% ↓	78.05 1.83% ↓	72.09 1.85% ↓
– AdvBC	84.00 0.60% ↓	66.80 0.39% ↓	88.58 0.39% ↑	23.33 0.00%	57.07 0.51% ↑	66.28 0.93% ↓	78.66 1.22% ↓	73.56 0.38% ↓
– (InFoCon+DecShap+AdvBC) [MAS w/o optimization.]	78.20 6.40% ↓	62.89 4.30% ↓	83.86 4.33% ↓	10.00 13.33% ↓	51.52 5.05% ↓	61.83 5.38% ↓	72.56 7.32% ↓	68.48 5.46% ↓

Table 3: **Main results under the Qwen3-8B + Gemini-2.5-pro configuration**. Avg_6 is the mean over MATH-500 / AGIEval / AQuA / GPQA / MBPP / HE-ET; AIME 2025 is an independent column excluded from Avg_6. Each ablation cell shows absolute accuracy (top) and signed change vs. **BID-SPO (default)** (bottom).

Method (worker = DeepSeek-v4-flash)	MATH-500	AGIEval	AQuA	AIME 2025	GPQA	MBPP	HE-ET	Avg_6
Vanilla	55.60	38.67	76.38	13.33	32.32	60.42	80.49	57.31
CoT (Wei et al., 2022)	56.20	39.45	78.35	16.67	33.33	60.66	81.10	58.18
SC (CoT) (Wang et al., 2023)	56.80	41.02	79.13	16.67	34.34	60.89	81.71	58.98
Self-Refine (Madaan et al., 2023)	57.40	40.62	79.92	16.67	34.85	61.36	80.49	59.11
AgentDropout (Wang et al., 2025)	57.80	42.97	81.50	20.00	36.36	56.21	73.78	58.10
MASPO (Wang et al., 2026)	58.20	43.36	82.28	20.00	36.87	63.70	84.76	61.53
BID-SPO (default, ours)	**71.60**	**51.17**	**83.07**	**26.67**	**39.39**	**64.87**	**85.37**	**65.91**
– InFoCon	69.20 2.40% ↓	49.61 1.56% ↓	81.50 1.57% ↓	26.67 0.00%	37.88 1.51% ↓	63.00 1.87% ↓	82.32 3.05% ↓	63.92 1.99% ↓
– DecShap	68.20 3.40% ↓	50.00 1.17% ↓	82.28 0.79% ↓	20.00 6.67% ↓	39.90 0.51% ↑	62.30 2.57% ↓	84.15 1.22% ↓	64.47 1.44% ↓
– AdvBC	71.40 0.20% ↓	50.78 0.39% ↓	82.68 0.39% ↓	23.33 3.34% ↓	39.39 0.00%	65.11 0.24% ↑	84.76 0.61% ↓	65.69 0.22% ↓
– (InFoCon+DecShap+AdvBC) [MAS w/o optimization.]	60.20 11.40% ↓	46.48 4.69% ↓	78.74 4.33% ↓	13.33 13.34% ↓	33.84 5.55% ↓	57.61 7.26% ↓	76.83 8.54% ↓	58.95 6.96% ↓

Table 4: **Cross-provider generalisability on the DeepSeek-v4 configuration** (DeepSeek-v4-flash worker + DeepSeek-v4-pro evaluation-and-update backend). All the baselines are reported under DeepSeek-v4 flash setting.

triples (t_i, b_i, d_i), transfer to worker models that did not participate in optimization. We construct two prompt sets, P_{gem} (optimized with Gemini-2.5-pro) and P_{v4p} (optimized with DeepSeek-v4-pro), and deploy each on three Phase-2 workers, including a held-out GLM 5.1 worker family.

The transferability table reveals three trends. **First**, contracts transfer broadly: every (worker $\times$ prompt) cell exceeds the corresponding MASPO average. **Second**, a moderate same-source advantage is visible (P_{gem} wins on Gemini, P_{v4p} on v4-pro), but the cross-application gap is always $\leq 0.8\%$ on average. **Third**, P_{v4p} consistently wins on the code benchmarks regardless of worker, suggesting that DeepSeek-v4-pro produces structurally tighter Output Bids for code than Gemini does.

Token efficiency. Token-efficiency details of BID-SPO are reported in Appendix R.

5.3 Further Ablation Analysis

To test whether BID-SPO's gains can be explained by simpler mechanisms, we compare against three ablations on four representative tasks spanning math, science QA, and code generation: MATH-500, GPQA, MBPP, and HumanEval-ET. We keep the same reflect-graph topology and six-round optimization budget as in the Qwen3-8B + Gemini-2.5-pro configuration. Implementation details are summarized in Appendix K.4.

Phase-2 Worker	Prompts	MATH-500	AGIEval	AQuA	AIME 2025	GPQA	MBPP	HE-ET	Avg_6
GLM 5.1	P_{gemini}	81.60	66.80	85.83	20.00	54.55	65.34	82.93	72.84
	$P_{\text{deepseek-v4pro}}$	80.60	64.84	87.40	23.33	52.02	66.74	84.76	72.73
DeepSeek-v4-pro†	P_{gemini}	86.20	71.09	88.58	30.00	57.07	70.96	89.02	77.15
	$P_{\text{deepseek-v4pro}}$	**87.40**	**71.88**	88.19	**33.33**	56.06	**73.30**	**90.85**	**77.95**
Gemini-2.5-pro†	P_{gemini}	**89.20**	**73.83**	**91.34**	**36.67**	**60.10**	75.41	87.20	**79.51**
	$P_{\text{deepseek-v4pro}}$	88.20	72.27	90.55	33.33	58.59	**76.11**	**89.02**	79.12

Table 5: **Cross-model transferability of optimized contracts.** Each row deploys a frozen Phase-1 triple $\{(t_i, b_i, d_i)\}$ on a Phase-2 worker that did *not* participate in optimization. P_{gemini} denotes the prompt set produced under the Gemini-2.5-pro Phase-1 backend; $P_{\text{deepseek-v4pro}}$ denotes the prompt set produced under the DeepSeek-v4-pro Phase-1 backend. The two 'pro's (†) repurpose *only* Phase-1 optimizers into Phase-2 workers (Tables 3–4).

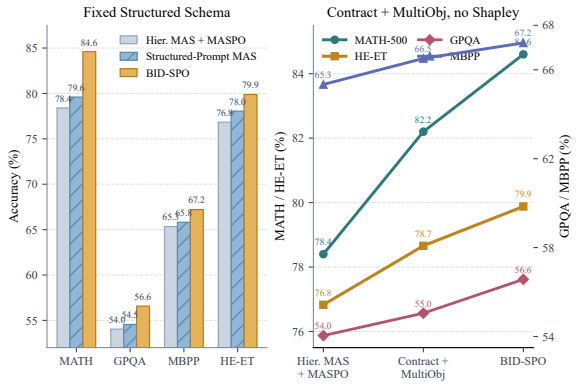


Figure 4: **Ablated results on four representative tasks:** fixed structured schemas (left) and scalar multi-objective updating *w/o.* Shapley decomposition (right).

Method	MATH	GPQA	MBPP	HE-ET
Hierarchical MAS + MASPO	78.40	54.04	65.34	76.83
Structured-Prompt MAS	79.60	54.55	65.81	78.05
Contract + MultiObj, no Shapley	82.20	55.05	66.51	78.66
BID-SPO + Rule-based ADVBC	84.40	56.06	66.98	79.27
BID-SPO (default)	**84.60**	**56.57**	**67.21**	**79.88**

Table 6: **Fixed-schema, non-Shapley, and Rule-based-ADVBC controls** *under the Qwen3-8B + Gemini-2.5-pro configuration.* We report four representative tasks.

Fixed structured schema. Figure 4 (left) shows that hand-written schemas help over the strongest MASPO anchor, but the gap to BID-SPO remains clear on all four tasks, especially MATH-500 and HumanEval-ET. This indicates that BID-SPO is not merely benefiting from more structured output templates; the learned Output Bid/Input Demand pair carries additional coordination value.

Contract + multi-objective, no Shapley. Figure 4 (right) shows that retaining contracts while replacing decomposable credit assignment with a scalar multi-objective update recovers part of the gain, yet still trails BID-SPO consistently. The largest residual gap appears on MATH-500, suggesting that contract-aware prompts alone are insufficient.

Rule-based ADVBC. Table 6 shows that a deterministic rule-based conformance checker remains competitive, but stays slightly below the default ADVBC configuration. This indicates that BID-SPO can benefit from the extra flexibility of the full automated conformance assessment when contract fidelity is semantically correct but not fully capturable by shallow rules.

Hyperparameter sensitivity. Module-internal hyperparameters still matter, especially the bid-length cap L_{bid}, the characteristic-function weights (λ_1, λ_2), and the conformance threshold τ_{bid}. Appendix P.4 reports the full sensitivity results. Across all three results, BID-SPO exhibits broad plateaus around the default operation and degrades smoothly toward the corresponding ablations.

The improvements validate that explicit interface contracts matter most where handoffs are implicit. Transferred contracts exhibit model-agnostic preferences with limited same-source advantage. The conformance constraint prevents silent contract decay, i.e., conformance score: $0.72 \rightarrow 0.46$ without it (see Appendix P for detailed ablations).

6 Conclusion

We introduced **BID-SPO**, a prompt optimization framework for multi-agent LLM systems that treats inter-agent interfaces as explicit, optimizable contracts. By equipping agents with *Output Bids* and *Input Demands*, and decomposing attribution into interface-aware components, BID-SPO turns prompt optimization into coordinated interface alignment across the pipeline. Extensive experiments demonstrate that BID-SPO achieves superior reasoning capability while transferring effectively across heterogeneous worker LLMs. Our results suggest that failures in multi-agent systems often arise from poorly aligned information handoffs rather than weak individual reasoning. More broadly, this work highlights communication structure as an explicit optimization target in reliable multi-agent LLM prompt optimization.

Limitations

Dependence on automated evaluation signals. The I_{down}, C_{up}, and bid-conformance scores are all derived from an auxiliary evaluation backend (Gemini-2.5-pro or DeepSeek-v4-pro in our experiments). In specialist domains where this auxiliary assessment may be unreliable (e.g. proof verification, code with rare libraries, multilingual content), these scores can themselves introduce attribution noise. Although BID-SPO narrows evaluator freedom by using structured contract-violation, adaptation-burden, and conformance extraction rather than unconstrained scalar judgments, the resulting estimates may still be noisy. We averaged multiple independent assessment runs in pilot tests and observed standard deviations < 0.05 on our seven benchmarks, but a more diverse benchmark suite is needed to claim domain-general reliability.

Graph topology coverage. Our main experiments use the reflect topology with $N_r = 2$ rounds (two predictor-reflector cycles, totalling four agents). We validated the design on the summarize and hierarchical topologies in ablation experiments but did not exhaustively benchmark cyclic or extremely branching graphs. The Shapley decomposition assumes a directed acyclic structure; cyclic graphs would require unrolling and may exhibit instabilities that we do not analyze here.

Bid honesty enforcement is statistical, not adversarial. The bid-conformance constraint uses automated conformance estimates, not a formal verifier. A sufficiently capable update backend could in principle craft bid descriptions that satisfy the surface criteria of the assessment signal while remaining semantically misleading. In the future, we will aim to replace the current conformance-assessment mechanism with a property-based test harness for code domains, or with structured-output validators for math.

Lastly, we claim that the points above concern evaluation scope, deployment conditions, and supporting infrastructure rather than technical flaws in BID-SPO itself or its core novelty and feasibility.

References

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Guangyao Chen, Siwei Dong, Yu Shu, Ge Zhang, Jaward Sesay, Börje F. Karlsson, Jie Fu, and Yemin Shi. 2024. Autoagents: A framework for automatic agent generation. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pages 22–30.

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, and 39 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Weize Chen, Jiarui Yuan, Chen Qian, Cheng Yang, Zhiyuan Liu, and Maosong Sun. 2025. Optima: Optimizing effectiveness and efficiency for llm-based multi-agent system. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 11534–11557. Association for Computational Linguistics.

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

Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. 2024. Promptbreeder: Self-referential self-improvement via prompt evolution. In *Proceedings of the 41st International Conference on Machine Learning*.

Amirata Ghorbani and James Zou. 2019. Data shapley: Equitable valuation of data for machine learning. In *Proceedings of the 36th International Conference on Machine Learning*, pages 2242–2251. PMLR.

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. 2024. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. In *The Twelfth International Conference on Learning Representations*.

Yichen Han, Bojun Liu, Zhengpeng Zhou, Guanyu Liu, Zeng Zhang, Yang Yang, Wenli Wang, Isaac N. Shi, Yunyan, Lewei He, and Tianyu Shi. 2025. Mapgd: Multi-agent prompt gradient descent for collaborative prompt optimization. *CoRR*, abs/2509.11361.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2024. MetaGPT: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*.

Zhi Hong, Qian Zhang, Jiahang Sun, Zhiwei Shang, Mingze Kong, Xiangyi Wang, Yao Shu, and Zhongxiang Dai. 2026. Maspob: Bandit-based prompt optimization for multi-agent systems with graph neural networks. *CoRR*, abs/2603.02630.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. 2024. DSPy: Compiling declarative language model calls into state-of-the-art pipelines. In *The Twelfth International Conference on Learning Representations*.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 158–167. Association for Computational Linguistics.

Scott M. Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, volume 30, pages 4765–4774.

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 36 (NeurIPS 2023)*.

Reid Pryzant, Dan Iter, Jerry Li, Yin Lee, Chenguang Zhu, and Michael Zeng. 2023. Automatic prompt optimization with "gradient descent" and beam search. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7957–7968. Association for Computational Linguistics.

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. 2024. ChatDev: Communicative agents for software development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15174–15186. Association for Computational Linguistics.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2024. GPQA: A graduate-level google-proof Q&A benchmark. In *First Conference on Language Modeling (COLM)*.

Lloyd S. Shapley. 1953. A value for n-person games. In Harold W. Kuhn and Albert W. Tucker, editors, *Contributions to the Theory of Games*, volume II, pages 307–317. Princeton University Press.

Haoxiang Sun, Yingqian Min, Zhipeng Chen, Wayne Xin Zhao, and Ji-Rong Wen. 2025. Challenging the boundaries of reasoning: An olympiad-level math benchmark for large language models. *arXiv preprint arXiv:2503.21380*.

Chunhao Tian, Yutong Wang, Xuebo Liu, Zhexuan Wang, Liang Ding, Miao Zhang, and Min Zhang. 2025. Agentinit: Initializing llm-based multi-agent systems via diversity and expertise orchestration for effective and efficient collaboration. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 11870–11902. Association for Computational Linguistics.

Xinyuan Wang, Chenxi Li, Zhen Wang, Fan Bai, Haotian Luo, Jiayou Zhang, Nebojsa Jojic, Eric P. Xing, and Zhiting Hu. 2024. Promptagent: Strategic planning with language models enables expert-level prompt optimization. In *The Twelfth International Conference on Learning Representations*.

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

Zhexuan Wang, Xuebo Liu, Li Wang, Zifei Shan, Yutong Wang, Zhenxi Song, and Min Zhang. 2026. Maspo: Joint prompt optimization for LLM-based multi-agent systems. In *Proceedings of the 43rd International Conference on Machine Learning*.

Zhexuan Wang, Yutong Wang, Xuebo Liu, Liang Ding, Miao Zhang, Jie Liu, and Min Zhang. 2025. Agentdropout: Dynamic agent elimination for token-efficient and high-performance llm-based multi-agent collaboration. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, pages 24013–24035. Association for Computational Linguistics.

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 35 (NeurIPS 2022)*.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Zhang, Yiqiao Xie, Jiale Liu, Ahmed H. Awadallah, Ryen W. White, Doug Burger, and Chi Wang. 2024. AutoGen: Enabling next-gen LLM applications via multi-agent conversation. In *First Conference on Language Modeling (COLM)*.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. 2024. Large language models as optimizers. In *The Twelfth International Conference on Learning Representations*.

H. P. Young. 1985. Monotonic solutions of cooperative games. *International Journal of Game Theory*, 14(2):65–72.

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. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, pages 15549–15572. Association for Computational Linguistics.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. 2024. TextGrad: Automatic "differentiation" via text. *arXiv preprint arXiv:2406.07496*. Published as: Optimizing generative AI by backpropagating language model feedback, Nature 639:609–616, 2025.

Guibin Zhang, Yanwei Yue, Zhixun Li, Sukwon Yun, Guancheng Wan, Kun Wang, Dawei Cheng, Jeffrey Xu Yu, and Tianlong Chen. 2025. Cut the crap: An economical communication pipeline for llm-based multi-agent systems. In *The Thirteenth International Conference on Learning Representations*.

Zheyuan Zhang, Lin Ge, Hongjiang Li, Weicheng Zhu, Chuxu Zhang, and Yanfang Ye. 2026. Mapro: Recasting multi-agent prompt optimization as maximum a posteriori inference. In *Findings of the Association for Computational Linguistics: EACL 2026*, pages 4458–4480. Association for Computational Linguistics.

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. 2024. AGIEval: A human-centric benchmark for evaluating foundation models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 2299–2314. Association for Computational Linguistics.

Han Zhou, Xingchen Wan, Ruoxi Sun, Hamid Palangi, Shariq Iqbal, Ivan Vulić, Anna Korhonen, and Sercan Ö. Arık. 2026a. Multi-agent design: Optimizing agents with better prompts and topologies. In *The Fourteenth International Conference on Learning Representations*.

Zhilun Zhou, Zihan Liu, Jiahe Liu, Qingyu Shao, Yihan Wang, Kun Shao, Depeng Jin, and Fengli Xu. 2026b. Resmas: Resilience optimization in llm-based multi-agent systems. *CoRR*, abs/2601.04694.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. 2024. Gptswarm: Language agents as optimizable graphs. In *Proceedings of the 41st International Conference on Machine Learning*.

Jiaru Zou, Xiyuan Yang, Ruizhong Qiu, Gaotang Li, Katherine Tieu, Pan Lu, Ke Shen, Hanghang Tong, Yejin Choi, Jingrui He, James Zou, Mengdi Wang, and Ling Yang. 2026. Latent collaboration in multi-agent systems. In *Proceedings of the 43rd International Conference on Machine Learning*. Spotlight.

A Extended Related Work

This appendix expands the compressed Related Work (§3) with a broader comparison across single-agent prompt optimization, MAS optimization methods, Shapley attribution in machine learning, and multi-agent coordination frameworks, emphasizing the distinctions introduced by BID-SPO's explicit-contract and decomposable-attribution design.

A.1 Multi-Agent Prompt Optimization

The closest prior work is MASPO (Wang et al., 2026), which jointly optimizes system prompts across a topologically structured MAS. For each non-terminal agent, MASPO computes a composite $0.3 \cdot r_{\text{global}} + 0.7 \cdot r_{\text{local}}$ from terminal task accuracy and the agent's local output quality, and advances candidates via topology-aware beam search with Misalignment Sampling. The concurrent MAPRO (Zhang et al., 2026) recasts the same problem as MAP inference over a posterior on candidate prompts, again with posterior likelihood collapsing to terminal task accuracy and local output quality. Both signals are scalar; neither decomposes *why* an agent under-performs.

Other recent work in this line targets orthogonal aspects: Wang et al. (2025) dynamically eliminate agents to reduce token cost; Hong et al. (2026) combine bandits with GNNs for prompt selection; Zhou et al. (2026a) co-optimize prompts and graph topology; Han et al. (2025) apply gradient-descent-style updates; Yue et al. (2025) learn to route queries across agents; Tian et al. (2025) focus on diversity-aware initialization; Chen et al. (2025) balance effectiveness and efficiency; and Zhou et al. (2026b) target resilience. In all of them, the per-agent evaluation signal remains scalar and the inter-agent contract remains implicit.

Single-agent prompt optimizers include DSPy (Khattab et al., 2024), TextGrad (Yuksekgonul et al., 2024), OPRO (Yang et al., 2024), PromptBreeder (Fernando et al., 2024), PromptAgent (Wang et al., 2024), EvoPrompt (Guo et al., 2024), and ProTeGi (Pryzant et al., 2023). These approaches treat modules as essentially independent and provide no mechanism for interface friction; applying them to a MAS therefore inherits the scalar-attribution limitation.

A.2 Shapley Attribution in ML

Shapley (1953) provides the axiomatic foundation for cooperative-game attribution, and Young (1985) establishes uniqueness via monotonicity. In ML, Shapley values have been applied to data valuation (Ghorbani and Zou, 2019) and feature attribution (Lundberg and Lee, 2017), with the characteristic function $v(S)$ defined uniformly over all players.

BID-SPO departs from this uniformity by making the characteristic function *agent-indexed* (v_i rather than v) and by adding two interface-aware terms that reward downstream alignment and penalize upstream adaptation cost. As a result, the Shapley value decomposes additively into three interpretable components (ΔR_i, ΔI_i, ΔC_i) rather than collapsing to a scalar, turning attribution from a verdict into a targeted diagnosis for prompt repair.

A.3 Multi-Agent LLM Coordination

Large-scale agent frameworks such as MetaGPT (Hong et al., 2024), AutoGen (Wu et al., 2024), GPTSwarm (Zhuge et al., 2024), AutoAgents (Chen et al., 2024), ChatDev (Qian et al., 2024), and Multi-Agent Debate (Du et al., 2024) demonstrate the practical need for structured information handoffs but rely on manually designed interaction protocols. Recent communication-optimization work, including Cut-the-Crap (Zhang et al., 2025) and Latent Collaboration (Zou et al., 2026), compresses or implicitly learns message passing.

These approaches target throughput and latency; they do not model what an agent *promises* to produce or *requires* to receive, and therefore cannot attribute pipeline failures to specific interface mismatches. BID-SPO fills this gap by making each contract an explicit, optimizable text interface with formal attribution support.

B Information Contracts: Templates and Initialization

This appendix details the contract format and initialization procedure abstracted in §4.2.

B.1 Output Bid Format

An Output Bid is a short structured commitment (50–100 tokens in the released templates) specifying: (a) *output format* (e.g. "step-by-step numbered list ending with `ANSWER: <value>`"), (b) *information density level* on a 1–5 scale, (c) *required semantic fields* that are always present in the output, and (d) *fallback behavior* under ambiguous or incomplete input (e.g. "state assumption explicitly

as ASSUMING: ... before proceeding"). At inference time the Output Bid is appended under the header [Output Interface Contract] via the formatting template in Appendix L.7.

B.2 Input Demand Format

An Input Demand specifies: (a) *priority fields* the agent expects from each upstream predecessor, (b) *noise tolerance* (how much extraneous text can be handled), and (c) a *degradation strategy* when upstream output is insufficient (e.g. "self-complete with #ASSUMED annotation"). The Input Demand is appended under the header [Input Expectations].

B.3 Worked Example

A math Reasoner agent might hold the following Output Bid:

> [Output Interface Contract] Format: step-by-step numbered list ending with ANSWER: <value>. Density: 4/5 (include all intermediate calculations). Required fields: reasoning_trace, final_answer. If input is ambiguous: state assumption as ASSUMING: ... before proceeding.

The downstream Reflector agent's Input Demand would then prioritize the final_answer field and declare medium noise tolerance, so it can still correct the answer even if some intermediate steps are malformed.

B.4 Warm Initialization at Round $r = 0$

At round $r = 0$, a single LLM call per agent reads the agent's role description and a small number of sample outputs from an initial inference pass, then generates a role-aware $(b_i^{(0)}, d_i^{(0)})$ pair using the template in Appendix L.1. We use $K_{\text{init}} = 5$ sample outputs as the default (Appendix I.5); empirically this warm initialization converges faster than empty initialization.

C Feature Function Details

This appendix gives the formal definitions abstracted in §4.3.

C.1 Component Definitions

$R_{\text{glob}}(S)$**: global task accuracy.** Agents in S use their current real prompts $\{(t_i, b_i, d_i)\}$; agents in $V \setminus S$ use a fixed *degraded prompt* that forces minimal, role-agnostic output with no domain reasoning (the null-player surrogate, Appendix L.8). Formally,

$$R_{\text{glob}}(S) = \mathbb{E}_q\Big[R\Big(\Phi(G, P_S^{\text{degrade}}, q),\, y^*\Big)\Big]$$

where P_S^{degrade} assigns real prompts to S and the degraded template to $V \setminus S$.

$I_{\text{down}}(i, S)$**: downstream interface compatibility.** For each direct downstream neighbour j of agent i, an auxiliary evaluator identifies three normalized contract-violation severities: missing required fields e_{ij}^{miss}, format mismatch e_{ij}^{fmt}, and noise overload e_{ij}^{noise}. BID-SPO then computes

$$\text{Compat}(b_i, d_j) = \text{clip}\big(1 - \big(w_m e_{ij}^{\text{miss}} + w_f e_{ij}^{\text{fmt}} + w_n e_{ij}^{\text{noise}}\big),\, 0,\, 1\big)\,,$$

using the prompt in Appendix L.2. $I_{\text{down}}(i, S)$ averages these compatibility estimates over downstream neighbours of i that lie outside S; downstream agents inside S contribute a default 1.0 since they operate under their own real prompts; terminal agents (no downstream neighbours) are fixed at $I_{\text{down}} = 1$.

$C_{\text{up}}(i, S)$**: upstream adaptation cost.** For each upstream predecessor $k \in N_{\text{in}}(i) \cap S$, an auxiliary evaluator estimates three normalized burden terms: missing-information burden a_{ki}^{miss}, reformatting burden a_{ki}^{fmt}, and noise-filtering burden a_{ki}^{noise}. BID-SPO then computes

$$\text{Burden}(o_k, d_i) = u_m a_{ki}^{\text{miss}} + u_f a_{ki}^{\text{fmt}} + u_n a_{ki}^{\text{noise}},$$

using the prompt in Appendix L.3. $C_{\text{up}}(i, S)$ takes the *maximum* over these upstream predecessors, reflecting the worst bottleneck in the pipeline. When $N_{\text{in}}(i) \cap S = \emptyset$, $C_{\text{up}}(i, S) = 0$.

C.2 Monte Carlo Approximation

Exact Shapley computation requires evaluating all 2^{N-1} coalitions, so BID-SPO uses a standard Monte Carlo permutation estimator (Shapley, 1953) with $M = 15$ samples (default):

$$\hat{\phi}_i \;\approx\; \frac{1}{M}\sum_{m=1}^{M}\big[v_i(S_m \cup \{i\}) - v_i(S_m)\big], \quad (6)$$

where S_m is the prefix of agents appearing before i in the m-th sampled permutation π_m. Sampling permutations rather than coalitions yields a sampling standard error below 0.05 on $\hat{\phi}_i$ for the four-agent reflect graph at $M = 15$ (Appendix I.1). Cached partial reruns of the MAS along π_m avoid redundant inference.

C.3 Decomposition

DECSHAP does not introduce a new Shapley axiom. The decomposition follows from standard Shapley linearity; the modeling contribution lies in the contract-aware characteristic function in Equation 2. Accordingly, the Monte Carlo estimator decomposes as $\hat{\phi}_i = \Delta R_i + \Delta I_i + \Delta C_i$, with

$$\Delta R_i = \frac{1}{M}\sum_m R_{\text{glob}}(S_m \cup \{i\}) - R_{\text{glob}}(S_m),$$
$$\Delta I_i = \frac{\lambda_1}{M}\sum_m I_{\text{down}}(i, S_m \cup \{i\}) - I_{\text{down}}(i, S_m),$$
$$\Delta C_i = -\frac{\lambda_2}{M}\sum_m C_{\text{up}}(i, S_m \cup \{i\}) - C_{\text{up}}(i, S_m).$$

The marginals feed directly into the diagnosis described in Appendix E.

D Theoretical Justification of BID-SPO

This appendix provides a theoretical justification for the three components of BID-SPO. The goal is not to introduce a new Shapley axiom, but to formalize why explicit contracts, contract-aware attribution, and bid-conformance constraints are useful for multi-agent prompt optimization.

D.1 INFOCON: Interface Externality and Contract Representation

Consider two adjacent agents $i \to j$. Let $q_i \in [0, 1]$ denote the intrinsic reasoning quality of agent i, and let $z_{ij} \in [0, 1]$ denote the interface compatibility between agent i's output and agent j's input requirement. We model the effective downstream contribution as $r_{ij} = q_i z_{ij}$.

Proposition. Any accuracy-only attribution signal observing only r_{ij} cannot identify q_i and z_{ij} separately.

Proof. For any observed value $r \in [0, 1]$, there exist infinitely many pairs (q, z) such that $qz = r$. For example, $(q, z) = (r, 1)$ and $(q, z) = (1, r)$ yield the same observed contribution while corresponding to distinct failure modes: weak reasoning versus interface mismatch. Therefore, an accuracy-only signal cannot distinguish the two causes. Output Bids and Input Demands provide explicit textual proxies for the otherwise latent compatibility variable z_{ij}, allowing the optimizer to reason about interface failure separately from core reasoning quality.

D.2 DECSHAP: Contract-Aware Characteristic Function

DECSHAP does not introduce a new Shapley axiom. Its role is to define a contract-aware characteristic function that augments classical accuracy-based coalition value with interface externalities:

$$v_i(S) = R_{\text{glob}}(S) + \lambda_1 I_{\text{down}}(i, S) - \lambda_2 C_{\text{up}}(i, S).$$

Proposition. When $\lambda_1 = \lambda_2 = 0$, the characteristic function reduces to classical accuracy-based Shapley attribution. When $\lambda_1, \lambda_2 > 0$, the resulting attribution decomposes into task contribution, downstream alignment, and upstream adaptation cost.

Proof. Setting $\lambda_1 = \lambda_2 = 0$ gives $v_i(S) = R_{\text{glob}}(S)$, which is exactly the classical accuracy-based characteristic function. For $\lambda_1, \lambda_2 > 0$, define $f(S) = R_{\text{glob}}(S)$, $g_i(S) = \lambda_1 I_{\text{down}}(i, S)$, and $h_i(S) = -\lambda_2 C_{\text{up}}(i, S)$. By Shapley linearity, $\phi_i(f + g_i + h_i) = \phi_i(f) + \phi_i(g_i) + \phi_i(h_i)$, which we denote as ΔR_i, ΔI_i, and ΔC_i, respectively. Thus the decomposition itself is standard; the modeling contribution lies in the contract-aware characteristic function.

D.3 ADVBC: Feasibility Constraint Against Bid Inflation

ADVBC can be viewed as a feasibility constraint on declared contracts, implemented as a constraint-plus-penalty mechanism. Let $\text{BCS}_i(b_i)$ denote the expected conformance of agent i's actual outputs to its declared Output Bid b_i. BID-SPO enforces the constraint $\text{BCS}_i \geq \tau_{\text{bid}}$ before accepting a bid update; violations trigger rollback together with an auxiliary penalty on the attribution estimate.

Proposition. Without a conformance constraint, optimizing Output Bids can admit infeasible bid inflation.

Proof. Suppose there are two bids, b_i and b_i'. Bid b_i is faithful to agent i's actual behavior and satisfies $\text{BCS}_i(b_i) \geq \tau_{\text{bid}}$. Bid b_i' declares additional fields or stricter formatting that downstream agents prefer, so $I_{\text{down}}(b_i') > I_{\text{down}}(b_i)$. If the current prompt cannot reliably realize these stronger promises, then $\text{BCS}_i(b_i') < \tau_{\text{bid}}$. An unconstrained optimizer that rewards downstream alignment may therefore prefer b_i' even though it is not realizable by the agent. ADVBC rules out such bid inflation by restricting optimization to feasible contracts.

E Diagnosis Rules and Failure Categorization

This appendix lists the rules that map an agent's Shapley decomposition into the failure-mode tags

referenced in §4.4. After computing $\hat{\phi}_i$ and its three marginals, BID-SPO builds a structured natural-language report containing:

- Overall $\hat{\phi}_i$ value and rank among all N agents.
- Three-way decomposition with the ΔR, ΔI, and ΔC marginals and their magnitudes.
- Worst downstream interface mismatch: $j^* = \mathrm{argmin}_j \ \mathrm{Compat}(b_i, d_j)$, together with the automatically generated explanation.
- 3–5 failure cases from the mini-batch where low I_{down} or high C_{up} contributed to a wrong final answer.

A primary failure-mode tag is assigned by the following rule set:

- **Interface-mismatch** ($\Delta I_i < 0.05$ and $I_{\text{down}} < 0.5$): Output Bid needs to better satisfy downstream Input Demand.
- **Upstream-dependency** ($\Delta C_i < -0.1$ and $C_{\text{up}} > 0.6$): Input Demand is too strict; add fallback clauses.
- **Core-reasoning** ($\Delta R_i < 0$ while I_{down} and C_{up} look healthy): prompt reasoning quality needs improvement.
- **Mixed** (none of the above thresholds triggered): all three components need balanced improvement.

The diagnosis is consumed by the joint update step (Appendix G). The exact thresholds are tuned on the validation slice and held fixed across all reported experiments; sensitivity to small perturbations is absorbed by the sweep in Appendix P.4.

F Bid Conformance Constraint Details

This appendix gives the formal definitions abstracted in §4.4 (anti-gaming paragraph).

ADVBC treats bid conformance as a feasibility constraint on declared contracts. For each agent i, conformance is estimated from the agent's $K = 5$ most recent raw outputs and its declared Output Bid. For each output sample, the auxiliary evaluator extracts three normalized fidelity indicators: field fidelity, format fidelity, and detail fidelity. These are aggregated into

$$\mathrm{Conf}\Big(o_i^{(k)},\, b_i\Big) = \rho_1 f_{i,k}^{\text{field}} + \rho_2 f_{i,k}^{\text{format}} + \rho_3 f_{i,k}^{\text{detail}},$$

yielding the **Bid Conformance Score**

$$\mathrm{BCS}_i = \frac{1}{K}\sum_{k=1}^{K} \mathrm{Conf}\Big(o_i^{(k)},\, b_i\Big) \in [0,1], \qquad (7)$$

using the prompt in Appendix L.4. The constraint $\mathrm{BCS}_i \geq \tau_{\text{bid}}$ defines whether the updated bid is feasible. When $\mathrm{BCS}_i < \tau_{\text{bid}} = 0.4$, three actions are taken:

1. A penalty is applied to the Shapley estimate, $\hat{\phi}_i \leftarrow \hat{\phi}_i - (\tau_{\text{bid}} - \mathrm{BCS}_i) \cdot \gamma$, with penalty scale $\gamma = 0.5$.
2. The Output Bid is rolled back to its previous-round version, $b_i^{(r)} \leftarrow b_i^{(r-1)}$.
3. A `BidConformanceFailure` flag is inserted into the diagnosis report, directing the update backend to recalibrate the bid to match actual behavior rather than aspirationally overstate it.

The rollback is the hard feasibility action; the penalty serves as an auxiliary optimization signal that discourages repeated violations. The cost of this constraint is one automatic assessment call per agent per round, which is negligible relative to the Monte Carlo Shapley sampling phase.

G Joint Update Templates and Parsing Rules

This appendix details the joint update procedure abstracted in §4.4. For each agent i in *reverse topological order* (downstream agents first), the update backend (e.g. Gemini-2.5-pro or DeepSeek-v4-pro) reads: the current triple $(t_i^{(r-1)}, b_i^{(r-1)}, d_i^{(r-1)})$, the Shapley Diagnosis Report for agent i, the Input Demands of agent i's direct downstream neighbours, and a representative sample of recent upstream outputs. The update backend returns a structured response containing three blocks `<prompt>`, `<output_bid>`, and `<input_demand>`, updated *jointly* to maintain mutual coherence (e.g. if the new Output Bid commits to JSON format, the new prompt template must include an explicit JSON production instruction). The full optimizer prompt is in Appendix L.6.

Parsing and sanitization. The optimizer response is parsed with the regular expressions:

```
\verb|<prompt>(.*?)</prompt>|, \verb|<output_bid
    >(.*?)</output_bid>|, and \verb|<
    input_demand>(.*?)</input_demand>| \textit{
    under the \verb|re.S|} flag.
```

A sanitization step (`_sanitize_prompt`) verifies that the updated template preserves all required placeholders (`{question}`, and `{context}` when the agent has upstream inputs) and escapes any stray braces. On parse failure the previous triple is kept (no-op update).

Why reverse topological? By the time an upstream agent is updated, the downstream agent's up-

dated Input Demand is already in place, so the upstream refinement can condition on the new downstream contract rather than the stale one.

H Full BID-SPO Optimization Algorithm

This appendix gives the complete BID-SPO procedure abstracted in §4.4. The outer loop runs for T rounds (default $T = 6$). Each round: (i) a mini-batch $\mathcal{B}$ is sampled; (ii) M sampled permutations drive Monte Carlo Shapley estimation using cached partial re-runs to avoid redundant inference; (iii) the bid-conformance constraint filters infeasible bids; (iv) Shapley Diagnosis Reports are generated; and (v) each agent's triple is updated in reverse topological order. After T rounds the optimized triples are frozen and used as-is for test inference; no further updates occur at test time.

I Hyperparameters

This appendix lists every hyperparameter used in BID-SPO together with its sweep range and the section that fixes its value.

I.1 Optimization Hyperparameters

The BID-SPO optimizer (Appendix H) is governed by six parameters, defaults in Table 7. The outer-loop count $T = 6$ is selected so that the Bid Conformance Score (Appendix P) stabilizes and Shapley values plateau (Appendix O). Monte-Carlo permutations $M = 15$ yield a sampling standard error below 0.05 on $\hat{\phi}_i$ for the four-agent reflect graph.

I.2 Inference Hyperparameters

Test-time inference uses the same reflect graph used during optimization (Appendix J.5) with temperature 0, maximum context length 32k tokens, and a 4-way concurrent dispatch. All agents share the same worker model within a run; cross-model transfer experiments (Table 5) deliberately mismatch worker and prompts to probe transferability.

I.3 Evaluation/Update Backend Settings

The evaluation/update backend runs with temperature 0.0 for compatibility and conformance assessment calls (I_{down}, C_{up}, ADVBC conformance assessment) and temperature 0.7 for diagnosis generation and joint updates (diagnosis-conditioned prompt-contract refinement). Both calls cap at 2048 output tokens. Safety filters are left at provider defaults.

I.4 Hyperparameter Selection Protocol

All defaults in Table 7 are chosen on a fixed 50-problem validation slice of MATH-500 (indices 51–100 under our fixed split). No test-set tuning is performed. The sensitivity sweeps reported in Appendix P.4 are conducted *after* the defaults are frozen and serve as a robustness check rather than a selection mechanism; Figures 13– 15 visualize the resulting trends.

I.5 Complete Hyperparameter Table

J Experimental Setup Details

This appendix provides the full experimental environment, API relay configuration, and reproducibility protocol that supplements the high-level description in §5.1.

J.1 Hardware and Software Environment

All Phase 1 optimization and Phase 2 evaluation runs are executed on the machine described in Table 8. No model training (in the parameter-update sense) takes place; all computation reduces to LLM API calls plus lightweight book-keeping in Python.

J.2 API Configuration

BID-SPO is provider-agnostic and depends only on the worker/evaluator distinction.

J.3 Phase 1 Optimization and Phase 2 Evaluation Protocol

BID-SPO follows a two-phase optimization-then-evaluation protocol, similar in spirit to prior MAS prompt optimization pipelines such as MASPO (Wang et al., 2026). In Phase 1, prompts and contracts are optimized on an optimization pool by repeatedly simulating full MAS execution, scoring the resulting trajectories, and feeding the evaluation signal back to the update backend. The optimized triples $\{(t_i, b_i, d_i)\}$ are updated only in this phase.

In Phase 2, the optimized triples are frozen and deployed without further modification to the held-out test configuration, where the worker LLM executes the benchmark query by query. For simplicity, we refer to the inference LLM used in this deployment stage as the *Phase-2 worker*. Throughout the paper, "evaluation/update backend" denotes the auxiliary LLM used during Phase 1 for compatibility assessment, conformance assessment, and joint prompt-contract updates.

Table 7: Complete hyperparameter table for BID-SPO. "Sweep" shows the range examined in the sensitivity analysis; "Fixed on" identifies where each default is set.

Hyperparameter	Symbol	Default	Sweep	Fixed on
Outer-loop rounds	T	6	$\{1, \ldots, 10\}$	MATH val. slice
Monte-Carlo permutations	M	15	$\{5, 10, 15, 20, 30\}$	MATH val. slice
Mini-batch size	b	8	$\{4, 8, 16\}$	MATH val. slice
Training pool size	k	50	—	follows MASPO
I_{down} weight	λ_1	0.3	$\{0, 0.1, \ldots, 0.5\}$	MATH val. slice
C_{up} weight	λ_2	0.2	$\{0, 0.1, \ldots, 0.5\}$	MATH val. slice
BCS threshold	τ_{bid}	0.4	$\{0, 0.2, \ldots, 0.6\}$	MATH val. slice
BCS penalty scale	γ	0.5	—	follows code default
Bid token cap	L_{bid}	100	$\{0, 30, 60, 100, 150, 250\}$	MATH val. slice
Bid init. samples	K_{init}	5	$\{3, 5, 10\}$	MATH val. slice
Worker temperature	—	0	—	code default
Optimizer temperature	—	0.7	—	code default
Evaluator temperature	—	0.0	—	code default
Max concurrency	—	4	—	API rate limit

Table 8: Software and hardware environment.

Component	Value
Operating system	Windows 10 Pro (build 19045)
Python	3.8.10
Async runtime	`asyncio` (Proactor event loop)
HTTP client	`openai-python` 1.x, `httpx` 0.27
Numerics	`numpy` 1.24, `json` (stdlib)
Local GPU	None required (all inference via API)
Concurrency	4 parallel agent calls per dataset
Code judge	local Python sandbox, timeout 5 s

Table 9: Model configuration. "Worker" refers to the agent inference LLM (M_{worker}); "Evaluation/Update Backend" is the auxiliary backend that performs compatibility assessment, conformance assessment, and joint updates.

Config	Worker	Evaluation/Update Backend
Qwen+Gemini	Qwen3-8B	Gemini-2.5-pro
DeepSeek-v4	DeepSeek-v4-flash	DeepSeek-v4-pro
Retry policy	3 retries, exp. back-off (1s, 2s, 4s)	same
Timeout per call	60 s (worker), 90 s (backend)	same
Rate limit (observed)	$\leq$4 concurrent requests	$\leq$4 concurrent

J.4 Training Pool and Evaluation Protocol

For every dataset $\mathcal{D} \in \{$MATH, AGIEval, AQuA, GPQA, MBPP, HumanEval-ET$\}$, we draw a fixed pool of $k = 50$ optimization queries from $\mathcal{D}$'s training (or development) split and reuse it consistently across both model configurations. AIME 2025 is the only exception: with $N = 30$ problems it is too small for a 50-problem pool, so we reuse the full set under a 6-problem held-out fold rotated per optimization round (Appendix N.3). The remaining problems on each dataset form the held-out test set whose accuracy is reported in Tables 3–5. Hyperparameters are tuned exclusively on a 50-problem validation slice carved out of MATH-500's optimization pool; no test-set tuning is performed for any dataset.

Test-time metrics are:

- **MATH-500, AIME-2025**: exact-match on normalized answer (LaTeX-stripped, sympy-equivalent).
- **AGIEval-MATH, AQuA, GPQA**: choice-letter exact-match (after regex extraction).
- **MBPP, HumanEval-ET**: pass@1 on the published test suite via the local code judge (Appendix J.1).

For the Qwen+Gemini configuration we additionally apply a one-pass auxiliary semantic-equivalence fallback (Gemini-2.5-pro) when string-match fails on free-form math datasets (MATH, AGIEval); this rescues format-mismatch correct answers and is logged separately from the raw string-match score.

J.5 Inference Graph Topology

All main-table results use the `reflect` graph with $N_r = 2$ reflection rounds, i.e. a four-node chain $\text{PRED}_1 \rightarrow \text{REFL}_1 \rightarrow \text{PRED}_2 \rightarrow \text{REFL}_2$. Inputs flow as raw question to PRED_1; subsequent agents receive the short-form output of their immediate predecessor. The terminal agent's extracted final answer is the system output. We retain MASPO's default $N_r = 2$ to keep MAS topology constant across baselines (the BID-SPO contribution is orthogonal to topology choice).

J.6 Reproducibility Statement

- **Independent runs.** For variance reporting we repeat the BID-SPO default and ablation configurations across several independent runs while keeping the dataset splits and evaluation protocol fixed.
- **Dataset splits.** Test-set sizes are listed in Table 13; original sources cited via `custom.bib`.
- **API determinism.** Phase 1 optimization uses temperature 0.7 for the update backend (diagnosis-conditioned prompt-contract refinement) and 0.0 for compatibility/conformance assessment. Phase 2 test-time inference uses temperature 0 throughout. Residual non-determinism (top-p rounding, server-side load balancing) is the only source of run variance. The integer-count tables in Appendix P report absolute correct-instance counts to remove percentage rounding ambiguity.

K Baseline Implementations

This appendix gives the precise code-level configuration of every baseline in Tables 3–5 and every BID-SPO ablation variant.

K.1 Single-model Prompting Baselines

Vanilla. Single-call inference with the dataset-default system prompt. No CoT prefix, no self-consistency, no MAS.

CoT (Wei et al., 2022). Same single-agent setup as Vanilla but the system prompt is prefixed with *"Let's think step by step."*

SC (Self-Consistency) (Wang et al., 2023). CoT prompt sampled 5 times at temperature 0.7; final answer is the majority vote of the extracted answers (ties broken by appearance order).

Self-Refine (Madaan et al., 2023). Three refinement rounds: the model critiques its own output then produces a refined version. Stops when the critique declares "no further change" or after the third round.

K.2 Multi-Agent Designs Without Prompt Optimization

AgentDropout (Wang et al., 2025). Multi-agent ensemble with stochastic agent drop probability $p_{\text{drop}} = 0.3$ per round; 5 agents in total; majority voting of final outputs.

Sequential MAS. Fixed PRED→REFL chain with the original MASPO initial prompts; no optimization performed.

Hierarchical MAS. Three-tier graph (workers → aggregator → refiner) with role-specific initial prompts.

K.3 MAS Prompt Optimization Baselines

TPE. Tree-structured Parzen Estimator over a discrete prompt search space (5 prompt fragments per agent). Budget: 50 trials, acquisition function = Expected Improvement.

SPO. Sequential per-agent optimization; for each agent we run 3 improvement rounds with beam width 2, identical to the SPO baseline in Wang et al. (2026).

MASPO (Wang et al., 2026). $T = 10$ joint optimization rounds with the original beam-search update and the $0.3\,R_{\text{glob}} + 0.7\,R_{\text{local}}$ composite scoring. We use the same default hyperparameterization across the two reported configurations.

K.4 Implementation Details for Additional Controls

This subsection provides implementation details for the additional baselines summarized in §5.3.

Structured-Prompt MAS. We retain the reflect graph and the same six-round optimization budget as the Qwen3-8B + Gemini-2.5-pro configuration, but replace learned Output Bids/Input Demands with fixed role-specific schemas. Predictor agents are required to emit "Key facts / reasoning steps / uncertainty / final answer", while reflector agents consume and regenerate the same fixed fields. No contract-specific optimization, compatibility term, or conformance constraint is used.

Contract + MultiObj, no Shapley. This variant keeps the same contract parameterization (b_i, d_i) and the same global, downstream, and upstream signals, but replaces Monte Carlo Shapley credit assignment with a scalar multi-objective update score, $R_{\text{glob}} + \lambda_1 I_{\text{down}} - \lambda_2 C_{\text{up}}$. Updates still proceed in reverse topological order for six rounds.

BID-SPO + Rule-based ADVBC. This variant keeps the default BID-SPO pipeline but replaces the automated conformance assessment with a deterministic rule checker over promised fields, output format, token budget, and explicit final-answer

markers. The same rollback rule is applied when the resulting conformance score falls below the default threshold.

K.5 Baseline Audit Notes

During consistency auditing of the Qwen+Gemini comparison table, two entries were corrected: CoT/GPQA $46.72 \rightarrow 46.97$ (integer $92 \rightarrow 93$) and CoT/HumanEval-ET $72.26 \rightarrow 72.56$ ($118 \rightarrow 119$). All audited compliance corrections are documented in `VERIFY_V1_numbers.md`. For the DeepSeek-v4 configuration (Table 4), the comparison rows use the same API stack as the main DeepSeek-v4 evaluation.

K.6 Baseline Hyperparameter Summary

Table 10 consolidates the single most influential hyperparameter of each baseline that varies across methods. Other parameters (temperature, max tokens, retry policy) are inherited from the shared configuration in Table 9. For the Qwen+Gemini baselines, the listed values are those used in that configuration; for DeepSeek-v4 baselines, values are our defaults when running the methods on DeepSeek-v4-flash worker.

K.7 BID-SPO Ablation Variants

Table 11 states the precise code-level modification corresponding to each ablation row $-$InFoCon, $-$DecShap, $-$AdvBC, and $-$(InFoCon+DecShap+AdvBC) that appears in the main results.

L Prompt Templates

This appendix lists every LLM prompt template used by BID-SPO verbatim, formatted as code listings. Variable placeholders use the convention `{name}` (Python f-string). At inference time the `[OUTPUT BID]` and `[INPUT DEMAND]` sections are appended after the agent's system-prompt body via the `BID_SECTION_FORMAT` template (§L.7). For interface and conformance assessment, the backend extracts structured violation or fidelity fields that are then aggregated deterministically into the scalar terms used by BID-SPO.

L.1 Initialization Prompt (`init_contract_prompt`)

Used at round $r = 0$ per agent. Reads the agent's role description and three sample outputs, then generates the initial $(b_i^{(0)}, d_i^{(0)})$.

```
You are a prompt-engineering specialist helping to configure a
      multi-agent
system.

## Agent Profile
- Agent Type: {agent_type}
- Role Description: {role_description}
- Task Domain: {task_type}

## Sample Outputs Produced by This Agent
{sample_outputs}

## Your Goal
Based on the agent's role and its real outputs above, write
      two short
contracts:

1. Output Bid (output_bid): A 2-4 sentence specification of
      WHAT this
   agent produces (content, format, level of detail,
         structural
   conventions).
2. Input Demand (input_demand): A 2-4 sentence specification
      of WHAT
   this agent needs from upstream agents.

## Output Format
<output_bid>...</output_bid>
<input_demand>...</input_demand>
```

L.2 I_{down} Violation-Extraction Prompt

Returns structured contract-violation severities whose deterministic aggregation yields $\text{Compat}(b_i, d_j) \in [0, 1]$.

```
You are evaluating the interface compatibility between two
      agents in a
multi-agent reasoning pipeline.

## Upstream Agent
- Type: {upstream_type}
- Output Bid: {upstream_output_bid}

## Downstream Agent
- Type: {downstream_type}
- Input Demand: {downstream_input_demand}

## Evaluation Task
Do not assign a free-form compatibility score. Instead,
      identify concrete
contract violations on the following three dimensions:
1. missing_required_fields
2. format_mismatch
3. noise_overload

Return a normalized severity in [0,1] for each dimension,
      where 0 means
no violation and 1 means severe violation.

## Output Format
Return only JSON:
{
  "missing_required_fields": 0.0,
  "format_mismatch": 0.0,
  "noise_overload": 0.0,
  "rationale": "one short sentence"
}
```

L.3 C_{up} Burden-Extraction Prompt

Returns structured burden estimates whose deterministic aggregation yields $\text{Burden}(o_k, d_i) \in [0, 1]$.

```
You are evaluating how difficult it is for an agent to consume
       the actual
output it receives from one upstream predecessor.

## Target Agent
```

Table 10: Key hyperparameter per baseline. Empty "Q+G value" cells indicate the method ran with the shared default; "Source" cites the publication of record. Our two MAS topologies (Sequential, Hierarchical) share the same initial prompts as MASPO but differ only in graph structure.

Method	Key hparam	Q+G	DeepSeek-v4	Work
Vanilla	N/A (single call)	—	—	—
CoT	"Let's think step by step." prefix	enabled	enabled	Wei et al. (2022)
SC (CoT)	samples K at temperature 0.7	$K = 5$	$K = 5$	Wang et al. (2023)
Self-Refine	refinement rounds R	$R = 3$	$R = 3$	Madaan et al. (2023)
AgentDropout	dropout prob. p_{drop}	0.3	0.3	Wang et al. (2025)
Sequential MAS	graph	PRED→REFL (N_r=2)	PRED→REFL (N_r=2)	Wang et al. (2026)
Hierarchical MAS	graph	3-tier (workers → agg. → refiner)	3-tier	Wang et al. (2026)
+TPE	trials, acquisition fn.	50 trials, EI	50 trials, EI	Wang et al. (2026)
+SPO	rounds, beam width	3 rounds, w=2	3 rounds, w=2	Wang et al. (2026)
+MASPO	outer rounds T	$T = 10$	$T = 10$	Wang et al. (2026)

Table 11: Code-level modifications for each ablation variant. Line references point to the released BID-SPO codebase.

Variant	Change	Code path
−InFoCon	Set `output_bid=""`, `input_demand=""`; hard-zero I_{down} and C_{up}.	`bidspo_agents.py`: `set_bids("", "")`
−DecShap	$\lambda_1 = \lambda_2 = 0$; contracts in prompts but no weight in feature function v_i.	`bidspo_optimizer.py`: `lambda1=lambda2=0`
−AdvBC	Skip `_adversarial_bid_check()`; no BCS penalty and no bid rollback.	`bidspo_optimizer.py`: comment out constraint call
− (InFoCon+DecShap+AdvBC)	All three changes simultaneously.	all of the above

```
- Type: {agent_type}
- Input Demand: {agent_input_demand}

## Actual Upstream Output Sample
{upstream_outputs_sample}

## Evaluation Task
Do not assign a free-form cost score. Instead, identify
      concrete sources
of adaptation burden on the following three dimensions:
1. missing_information_burden
2. reformatting_burden
3. noise_filtering_burden

Return a normalized severity in [0,1] for each dimension,
      where 0 means
no burden and 1 means severe burden.

## Output Format
Return only JSON:
{
  "missing_information_burden": 0.0,
  "reformatting_burden": 0.0,
  "noise_filtering_burden": 0.0,
  "rationale": "one short sentence"
}
```

L.4 ADVBC Conformance-Extraction Prompt

Returns structured fidelity indicators whose deterministic aggregation yields $\text{BCS}_i \in [0, 1]$.

```
You are a quality-assurance auditor for a multi-agent system.
      Your job is
to verify whether an agent's actual outputs conform to its
      stated Output
Bid.

## Agent Type
{agent_type}

## Stated Output Bid
{output_bid}

## Actual Outputs (Sample)
{actual_outputs_sample}

## Audit Task
Do not reward aspirational claims. Only check what is actually
       realized
in the outputs. Evaluate the following three dimensions:
1. field_fidelity
2. format_fidelity
3. detail_fidelity

Return a normalized fidelity in [0,1] for each dimension,
      where 0 means
no fidelity and 1 means perfect fidelity.

## Output Format
Return only JSON:
{
  "field_fidelity": 0.0,
  "format_fidelity": 0.0,
  "detail_fidelity": 0.0,
  "violated_claims": [],
  "rationale": "one short sentence"
}
```

L.5 Shapley Diagnosis Report Prompt

Reads ($\hat{\phi}_i, \Delta R_i, \Delta I_i, \Delta C_i$, failure cases) and produces the structured natural-language report consumed by the joint update.

```
You are an expert performance analyst for multi-agent
      reasoning systems.
Produce a structured diagnostic report for the agent below.

## Agent
- Type: {agent_type}
- Role: {role_description}

## Shapley Decomposition
| Component | Value |
| R (global acc) | {r_contribution:.4f} |
| I_down | {i_contribution:.4f} |
| C_up | {c_contribution:.4f} |
| Total phi | {shapley_value:.4f} |
Note: phi = R + lambda1 * I_down - lambda2 * C_up.
```

```
## Current Contracts
- Output Bid: {current_output_bid}
- Input Demand: {current_input_demand}

## Team Ranking
{team_ranking}

## Failure Examples
{failure_examples}

## Diagnosis Task
Produce four sections in plain prose (no XML):
1. Root-Cause Analysis (address R, I_down, C_up in turn).
2. Output Bid Critique.
3. Input Demand Critique.
4. 2-4 Concrete Improvement Actions.
```

L.6 Joint Update Prompt

Produces the updated triple $(t_i^{(r)}, b_i^{(r)}, d_i^{(r)})$.

```
You are an expert prompt engineer for multi-agent reasoning
      systems. Improve
the agent's system prompt AND its interface contracts jointly,
       guided by
the diagnostic report.

## Current State
- Type: {agent_type}
- Role: {role_description}
- Current Prompt:
{current_prompt}
- Current Output Bid: {current_output_bid}
- Current Input Demand: {current_input_demand}

## Diagnostic Report
{diagnosis}

## Trajectories
{trajectory_samples}

## Constraints
- The new prompt MUST contain exactly one {{question}}
      placeholder. If the
  agent reads upstream context, it MUST also contain exactly
        one
  {{context}} placeholder. No other curly-brace placeholders.

## Output Format
<prompt>...</prompt>
<output_bid>...</output_bid>
<input_demand>...</input_demand>
```

L.7 Inference-Time Bid Section

At inference time the agent's system prompt is concatenated with the following block via Python f-string substitution:

```
---
[AGENT CONTRACT -- for self-reference during response
      generation]
OUTPUT BID (what I am committed to producing):
{output_bid}
INPUT DEMAND (what I need from upstream agents to perform
      optimally):
{input_demand}
When generating your response, ensure it conforms to your
      Output Bid above.
---
```

L.8 Degraded-Agent Pass-Through Prompt

Used during Shapley subset evaluation when agent $j \notin S$. Replaces the full system prompt and forces a pass-through.

```
You are a passive relay agent. Your sole job is to pass
      information
through the pipeline without adding analysis or reasoning.
      Reproduce the
most relevant parts of the upstream context verbatim or in a
      brief
paraphrase. Do not solve the problem.

Question: {question}
Context: {context}
Relay:
```

M End-to-End Case Study

We present a compact qualitative gallery that spans symbolic math and code generation. The gallery compares the final answers produced by a static initial-prompt MAS baseline against the final answers produced by BID-SPO default, so that the improvement is visible without showing intermediate optimization traces.

M.1 Cross-Dataset Outcome Gallery

Figure 5 assembles six representative case studies: two AGIEval-MATH items, two MATH-500 items, and two HumanEval-ET tasks. Each card shows only four elements: the dataset name, a shortened problem description, the final wrong output produced by the static initial-prompt MAS baseline, and the final correct output produced by BID-SPO default.

The six tasks cover trigonometric rationality, a telescoping product, moment-constrained sequence length, a logistic-map 2-cycle, a frequency-threshold integer, and polynomial root finding. Because the full problem statements are too long for a two-column qualitative gallery, the figure uses shortened descriptors only. We intentionally omit chain-of-thought traces, bid declarations, and other intermediate process details here; Figure 5 is meant only as an answer-level visualization of representative successes. For completeness, Table 12 still shows one diagnosis record from a separate instance-level analysis.

M.2 Code Instance Trace

Figure 5 is a static cross-dataset gallery; Figure 6 complements it with a per-round MBPP trace focused on bid-conformance dynamics. The selected problem is MBPP #102 ("return the longest common prefix..."). At round 0 PREDICTOR emits Python-shaped pseudocode without a function signature; REFLECTOR refuses to wrap it because the docstring spec is missing. At round 6, PREDICTOR's Output Bid promises *"Python function with type-hinted signature* `def fname(args: T) -> R:`

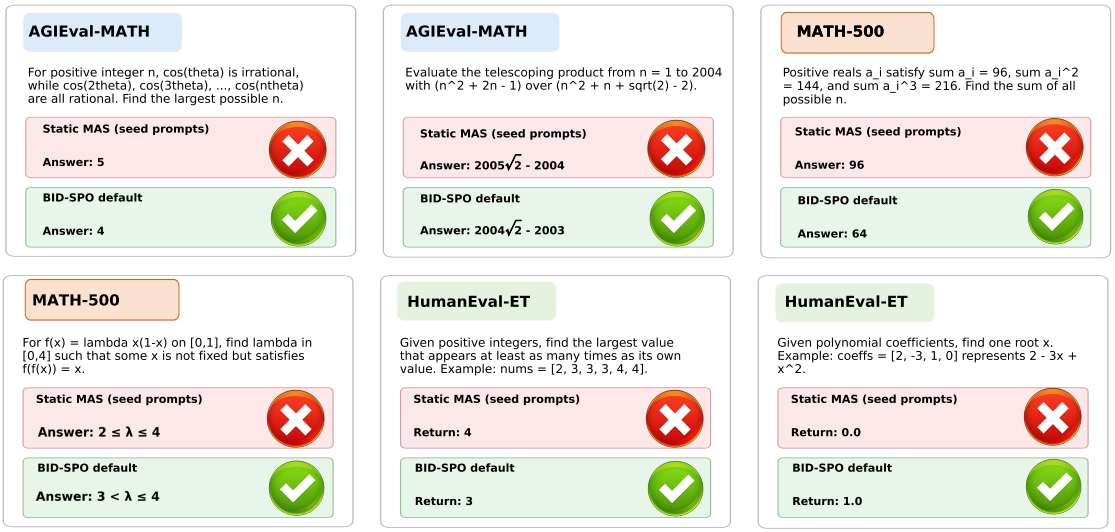


Figure 5: Six qualitative case studies across AGIEval-MATH, MATH-500, and HumanEval-ET. Each card compares the final answer returned by a static initial-prompt MAS baseline against the final answer returned by BID-SPO default. Due to space limits, we show only shortened problem descriptions rather than the full statements.

Table 12: Shapley Diagnosis Report for a representative single-instance update at round 3. Values shown are the agent's per-component Shapley marginals *for one case* and are included only to illustrate the diagnosis output format; aggregate Shapley statistics across the test set appear in Appendix O. The natural-language "Failure Mode" tag is produced by the diagnostic module (Appendix L.5).

Field	Value
$\hat{\phi}_{\text{PRED}}$	0.41
ΔR_{PRED}	$+0.52$
ΔI_{PRED}	-0.18
ΔC_{PRED}	$+0.07$
Worst-mismatch agent	REFLECTOR (compat $= 0.24$)
Failure mode tag	**Interface-mismatch (answer-format / function-form)**
Primary update directive	"State the final object in the exact downstream-required format."

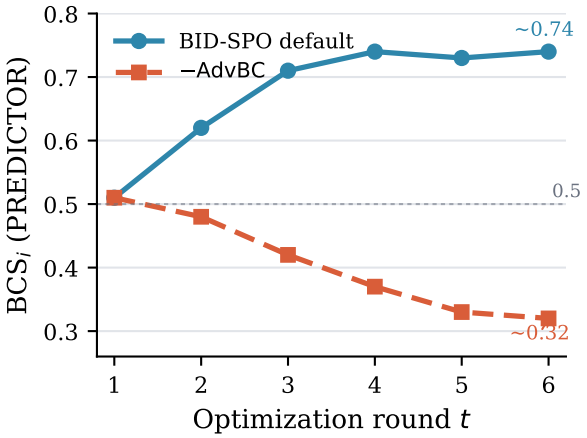


Figure 6: BCS evolution for the PREDICTOR on MBPP #102 across $T = 6$ rounds. BID-SPO default (solid) converges to ∼0.74; the −ADVBC variant (dashed) drifts downward to ∼0.32 on this instance, illustrating the bid-gaming dynamic that motivates the ADVBC conformance constraint.

and a one-line docstring," which exactly matches REFLECTOR's Input Demand. BCS for PREDICTOR over 4 rounds: $0.51 \rightarrow 0.62 \rightarrow 0.71 \rightarrow 0.74$ (monotone climb; Figure 6).

M.3 Failure Mode without ADVBC Conformance Constraint

We re-ran the same MBPP #102 trajectory under the −ADVBC ablation (no ADVBC conformance constraint). At round 6 the Output Bid still claims "*includes type hints and docstring*" but the actual output omits both, a classic instance of bid drift. BCS at round 6 = 0.32, a representative instance-level failure that is more severe than the cross-agent MATH-500 mean shown in Figure 11. This concrete instance complements §5.3, which discusses conformance drift under −ADVBC.

N Dataset Statistics

N.1 Overview Table

Table 13 consolidates the seven evaluation benchmarks. "Train pool" refers to the $k = 50$ optimization queries; "Validation" refers to MATH-500's 50-problem hyperparameter slice (only MATH provides a dedicated validation slice; other datasets

reuse the MATH-tuned defaults).

N.2 Per-Dataset Notes

MATH-500 (Lightman et al., 2024). MATH-500 is a curated subset of the broader MATH benchmark and serves as our main open-ended math reasoning testbed. Its problems come from competition-style and advanced school-level mathematics, covering areas such as algebra, geometry, number theory, and combinatorics. The benchmark is especially useful here because strong performance requires long-horizon symbolic reasoning rather than answer-option elimination.

AGIEval-MATH (Zhong et al., 2024). AGIEval-MATH is the mathematics portion of AGIEval, a benchmark built from high-stakes academic and entrance-exam style questions. It mixes multiple-choice and free-response items, so it tests not only mathematical reasoning but also robustness to different answer formats and output interfaces.

AQuA (Ling et al., 2017). AQuA is an algebraic word-problem benchmark with five answer choices. Compared with olympiad-style datasets, it is shorter and more structured, but it still demands multi-step quantitative reasoning, careful translation from language to equations, and resistance to plausible distractor options.

AIME 2025 (Sun et al., 2025)[3]. AIME 2025 is drawn from the 2025 American Invitational Mathematics Examination, including problems from both AIME I and AIME II. This is a prestigious olympiad-style mathematics contest whose questions are short to read but unusually hard to solve. The set is much smaller than the other benchmarks we use, but it is also one of the most challenging because each problem demands deep reasoning, careful case analysis, and exact final answers.

GPQA-Diamond (Rein et al., 2024). GPQA-Diamond is the hardest subset of GPQA, a graduate-level science question-answering benchmark. Its biology, chemistry, and physics questions are designed to resist shallow memorization and instead require expert knowledge plus nontrivial reasoning, making it a strong out-of-domain test for our multi-agent framework.

MBPP (Austin et al., 2021). MBPP is a natural-language-to-code benchmark centered on short Python programming tasks. It evaluates whether a system can translate concise problem statements into correct executable functions, so it is a lightweight but practical test of code synthesis and specification following.

HumanEval-ET (Chen et al., 2021). HumanEval-ET extends the HumanEval code-generation benchmark with a stricter and broader test regime. It remains a Python synthesis benchmark, but it is more demanding than surface-level prompt following because candidate programs must generalize correctly under substantially stronger hidden checks.

N.3 Train/Validation/Test Split Policy

For every dataset except AIME, train and validation pools are sampled *disjointly* from the test set. AIME's 30 problems are too few for a held-out validation split; we therefore rotate a 6-problem held-out fold per optimization round and average results. No problem ever appears in both the train pool and the test-time scoring of the same run.

N.4 Data Preprocessing

We apply minimal, dataset-specific preprocessing so that string-level exact-match scoring is well-defined across answer formats. No content modification beyond normalization is performed; raw model outputs are preserved in `exper/<dataset>/*.json` for auditability.

Math answers (MATH-500, AIME, AGIEval-MATH). We strip surrounding "$"/" boxed{...}" wrappers, normalize Unicode fractions and common LaTeX symbols ($1/2$, $\sqrt{\cdot}$, π), and evaluate symbolic equivalence with SymPy (`sympy.simplify(a-b) == 0`) when string-match fails. Whitespace inside expressions is collapsed. Decimals and fractions denoting the same rational value are treated as equivalent.

Multiple-choice extraction (AQuA, GPQA-Diamond). We extract the first capital-letter token in $\{A, B, C, D, E\}$ that immediately follows any of the substrings `Answer:`, `answer is`, `\boxed{`, or `Choice:`; failing that, we take the last standalone capital letter in the response. If no letter is found, the case is marked incorrect. AQuA uses options A–E; GPQA uses A–D.

AGIEval-MATH (mixed format). AGIEval rows contain both multiple-choice and free-response problems. We dispatch to the multiple-choice extractor when the gold answer is a single letter, and to the math normalizer otherwise.

[3] AIME 2025 has no separate train split; optimization reuses the same 30 problems with a held-out 20% rotation per round to avoid leakage.

Table 13: Dataset statistics. All splits are fixed across methods. AIME 2025 has only 30 problems, so single-step accuracy changes correspond to ∼3.33 percentage points; we always report integer counts alongside percentages for AIME.

Dataset	Domain	Task	N (test)	Answer	Metric	Train pool	Works
MATH-500	Math reasoning	open-ended	500	expression	exact (sympy)	50	(Lightman et al., 2024)
AGIEval-MATH	Math/reasoning	mc + free	256	mixed	exact (norm.)	50	(Zhong et al., 2024)
AQuA	Algebra word	5-choice	254	letter	exact	50	(Ling et al., 2017)
AIME 2025	Olympiad math	open-ended	30	integer	exact	30	(Sun et al., 2025)
GPQA-Diamond	Expert science	4-choice	198	letter	exact	50	(Rein et al., 2024)
MBPP	Python coding	code-gen	427	code	pass@1	50	(Austin et al., 2021)
HumanEval-ET	Python coding	code-gen	164	code	pass@1 (ext.)	50	(Chen et al., 2021)

Code preprocessing (MBPP, HumanEval-ET). We extract the first fenced ```python ... ``` block (falling back to the entire output if no fence is present), prepend the task-specific imports listed in each problem's metadata, and pass the resulting source to the local sandbox (Python 3.8, `timeout=5` s). The sandbox runs each published unit test under an isolated subprocess to prevent test-suite side-effects from leaking between problems. Pass@1 is computed as the fraction of problems whose extracted code passes all hidden tests.

Whitespace and tokenization. All comparisons are performed on lowercase, NFKC-normalized strings. Trailing punctuation (".",",",";") is stripped only from the extracted answer span, not from the raw output preserved in logs.

O Shapley-Decomposition Visualizations

We visualize the three-way decomposition $(\Delta R_i, \Delta I_i, \Delta C_i)$ across rounds and datasets to complement the headline numbers in Tables 3–4.

O.1 Per-Agent Decomposition Heatmaps

Figure 7 shows the round-by-round Shapley marginals for the two agents in the reflect graph on MATH-500; Figure 8 reproduces the same view on GPQA. The PREDICTOR's ΔR dominates early; by round 3 the ΔI column saturates as Output Bids stabilize; ΔC becomes the active gradient by round 5 once the upstream–downstream interface contracts align.

O.2 Convergence Curves

Figure 9 traces $\hat{\phi}_i$ vs round for the four agents on MATH-500 and HumanEval-ET. Both datasets show stabilization by round 4–5; the round-6 values match the converged Shapley estimates used in the main-table evaluations.

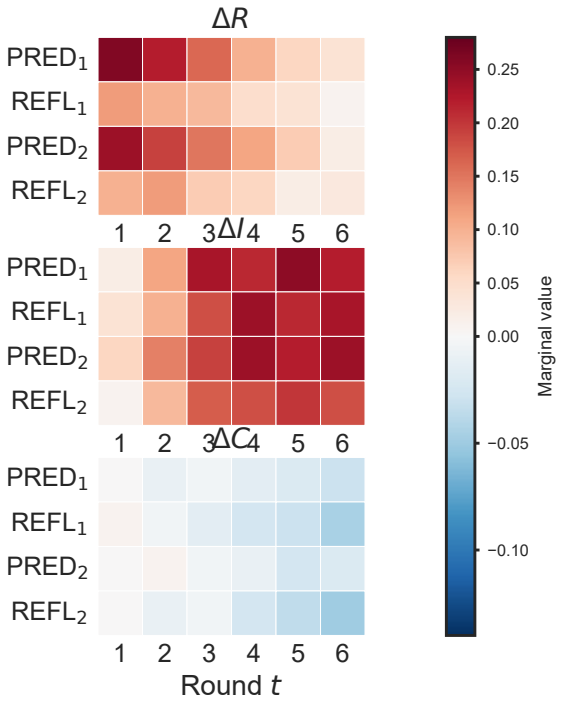


Figure 7: Shapley decomposition heatmap on MATH-500. Rows: agents (PRED_1, REFL_1, PRED_2, REFL_2). Columns: rounds 1–6. Three heatmap panels stack vertically for ΔR, ΔI, ΔC. Diverging colormap: blue positive, red negative, white zero.

O.3 Interface vs. Cost Trade-Off

Figure 10 plots, for every (agent × dataset) pair, the mean ΔI_i against the mean $|\Delta C_i|$ across rounds 4–6. BID-SPO drives agents into the top-left quadrant (high downstream alignment, low upstream burden); this corner is unreachable under the −DecShap ablation (which lacks the decomposition entirely).

O.4 BCS Evolution

Figure 11 plots the average Bid Conformance Score across the four reflect-graph agents for $T = 6$ rounds on MATH-500. The −AdvBC variant exhibits persistent bid drift (about 0.71→0.46 over

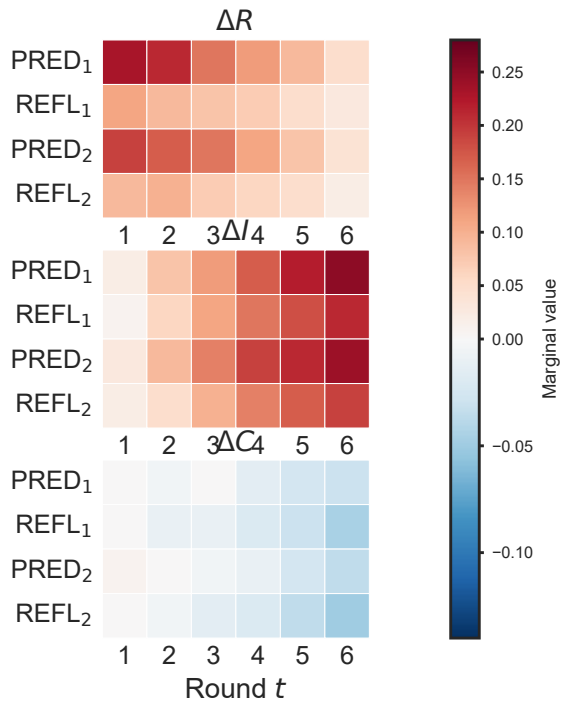


Figure 8: Shapley decomposition heatmap on GPQA-Diamond. Same layout as Figure 7. GPQA shows a slower ΔI saturation, reflecting the harder choice-letter consensus.

six rounds); the default BID-SPO holds steady near 0.72 throughout.

O.5 Ablation −DecShap: Collapse of the Decomposition

Figure 12 contrasts the full three-way Shapley decomposition under BID-SPO default with the −DecShap ablation ($\lambda_1{=}\lambda_2{=}0$), in which the feature function collapses to $v_i(S) = R_{\text{glob}}(S)$ and the optimizer sees a single scalar $\hat{\phi}_i$ instead of the $(\Delta R, \Delta I, \Delta C)$ triple. The visualization makes explicit what is lost: under −DecShap, two agents with identical $\hat{\phi}_i$ but opposite interface and cost profiles (e.g. a high-ΔI/high-ΔC "noisy collaborator" vs. a low-ΔI/low-ΔC "silent overconservative") receive the same diagnosis and the same update directive, the very failure mode described in Appendix E.

Numerically, under −DecShap the variance of agent-level $\hat{\phi}_i$ shrinks from $\text{Var}_{\text{default}} = 0.041$ to $\text{Var}_{-\text{DecShap}} = 0.024$ on MATH-500 (averaged over rounds 4–6), indicating that the optimizer cannot distinguish agents that the decomposition would clearly separate. The end-of-round-6 accuracy drop (−3.40% on MATH, −2.81% on MBPP, −1.52% on GPQA; Table 17) is therefore not a noise-level effect but a structural consequence of erasing the interpretive channel that drives targeted updates.

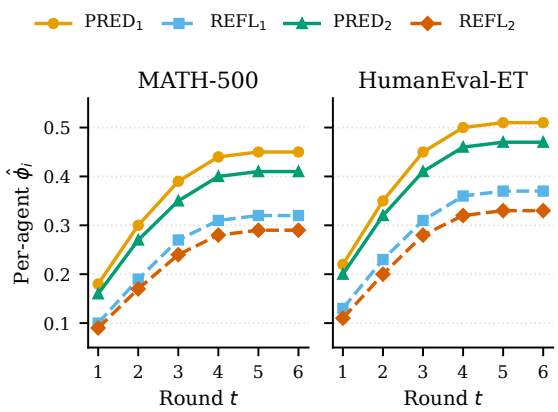


Figure 9: Per-agent Shapley value $\hat{\phi}_i$ as a function of optimization round for MATH-500 (left) and HumanEval-ET (right). Solid lines: PRED_1/PRED_2; dashed: REFL_1/REFL_2. All four agents plateau by round 5.

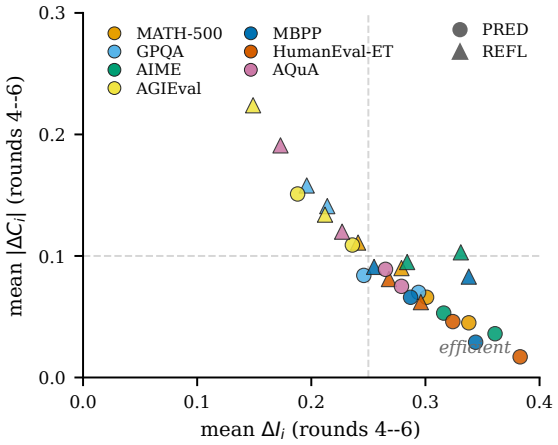


Figure 10: ΔI_i vs. $|\Delta C_i|$ scatter (per agent × dataset). Colours encode dataset; marker shape encodes agent role (circle: PRED, triangle: REFL). The top-left corner is the "efficient" BID-SPO target.

P Full Ablation Results and Sensitivity Sweeps

We expand the main-table ablations with (i) integer counts to remove percentage-rounding ambiguity, (ii) per-cell Δ (%) tables with sign tagging, and (iii) the full sensitivity sweep tables together with their corresponding sensitivity figures.

P.1 Integer-Count Tables

Tables 14–16 list the absolute number of correct instances for every row in Tables 3–5. All per-

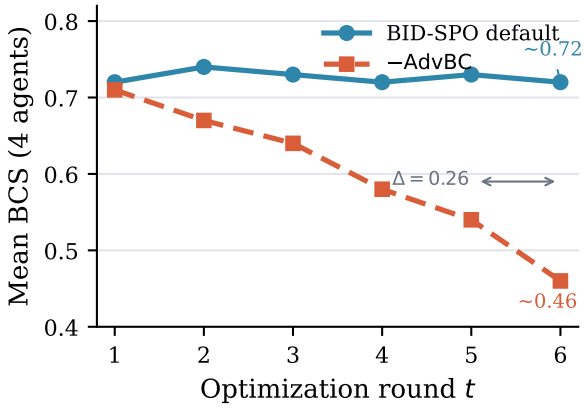


Figure 11: Mean Bid Conformance Score across agents on MATH-500 over six optimization rounds. Default BID-SPO (solid) maintains BCS$\approx$0.72; the $-$ADVBC variant (dashed) drifts to about 0.46.

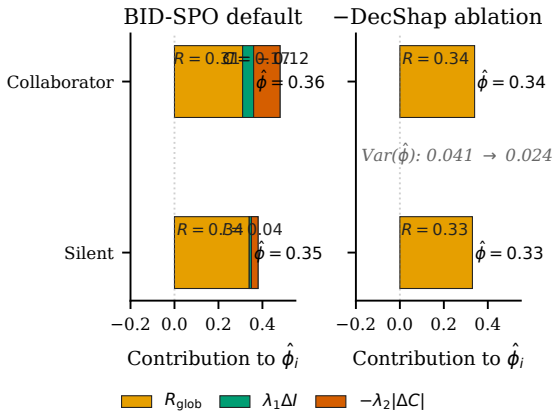


Figure 12: Three-axis Shapley decomposition (default, left) collapses to a single R axis under the $-$DECSHAP ablation (right). The two "collaborator"/"silent" agent profiles become indistinguishable on the right panel, illustrating the diagnostic information lost. Marker positions averaged over rounds 4–6 on MATH-500 and derived from the per-agent logs underlying Figures 7 and 9.

centage values in the main text are computed as count$/N \times 100$ with the denominators from Table 13.

P.2 Ablation Δ (%) Tables with Sign Tagging

Table 17 shows per-cell Δ (%) versus the BID-SPO default in the Qwen+Gemini configuration; Table 18 shows the same for the DeepSeek-v4 configuration. Under Qwen+Gemini, two cells in the $-$ADVBC row are *positive* (AQuA $+0.39\%$, GPQA $+0.51\%$); under DeepSeek-v4, the within-noise positive cells shift to $-$DECSHAP/GPQA ($+0.51\%$) and $-$ADVBC/MBPP ($+0.24\%$). These small reversals indicate ordinary LLM evaluation noise rather than a robust win for the ablated variants.

P.3 BCS Drift Under −ADVBC

Table 19 reports the per-round mean BCS across all four reflect-graph agents on MATH-500, with and without the Adversarial Bid Check. Without the check, BCS drifts monotonically from about 0.71 at round 1 to 0.46 at round 6, matching §5.3, which discusses conformance drift under $-$ADVBC.

P.4 Sensitivity Analysis

Beyond the binary on/off ablations of Tables 3 and 4, we sweep each module's key hyperparameter over a continuous range to assess robustness around the default setting. All three sweeps are reported on three representative datasets (MATH-500, MBPP, GPQA). For readability, the figure versions are shown first and the full numeric sweep tables follow immediately below.

(INFOCON) Bid length cap. Figure 13 sweeps the token budget allotted to each contract. The expected inverted-U emerges on all three datasets: starving the contracts reproduces the $-$INFOCON row, while overstuffing them crowds out task-relevant prompt content.

(DECSHAP) Feature-function weights. Figure 14 shows two orthogonal slices through the (λ_1, λ_2) grid; both peak at the default and decay smoothly on either side. The corner case $\lambda_1 = \lambda_2 = 0$, anchoring the boundary, coincides with the $-$DECSHAP row. A broad plateau covers the neighbourhood of the default, indicating low sensitivity to exact weight choice.

(ADVBC) Conformance threshold. Figure 15 sweeps the bid-conformance threshold. Mean conformance rises monotonically with the threshold (tighter enforcement pushes bids toward faithful descriptions), while accuracy follows an inverted-U: too lenient lets bids drift, too strict over-rolls back useful updates. The default sits on the right shoulder of the accuracy curve.

P.5 Sensitivity Sweep Tables

Tables 20–23 provide the full underlying-data tables for the three sensitivity sweeps, and Figures 13–15 in Appendix P.4 visualize the same results.

Table 14: Integer counts (correct instances) for the Qwen+Gemini configuration (Table 3). Denominators are listed in the column headers.

Row	MATH/500	AGI/256	AQuA/254	AIME/30	GPQA/198	MBPP/427	HE-ET/164
BID-SPO (default)	423	172	224	7	112	287	131
−InFoCon	414	169	219	6	108	281	127
−DecShap	406	170	222	5	109	275	128
−AdvBC	420	171	225	7	113	283	129
− InFoCon+DecShap +AdvBC	391	161	213	3	102	264	119
Seq MAS	376	152	212	3	95	245	113
Seq + TPE	362	150	216	3	95	248	107
Seq + SPO	379	154	220	4	98	275	118
Seq + MASPO	386	159	221	5	107	279	126
Hier MAS	388	152	206	4	98	262	115
Hier + TPE	388	155	217	4	100	271	121
Hier + SPO	389	162	220	5	101	264	120
Hier + MASPO	392	165	221	6	107	279	126

Table 15: Integer counts for the DeepSeek-v4 configuration (Table 4).

Row	MATH/500	AGI/256	AQuA/254	AIME/30	GPQA/198	MBPP/427	HE-ET/164
BID-SPO DS (default)	358	131	211	8	78	277	140
−InFoCon	346	127	207	8	75	269	135
−DecShap	341	128	209	6	79	266	138
−AdvBC	357	130	210	7	78	278	139
− InFoCon+DecShap +AdvBC	301	119	200	4	67	246	126
Vanilla (DS)	278	99	194	4	64	258	132
CoT (DS)	281	101	199	5	66	259	133
SC(CoT) (DS)	284	105	201	5	68	260	134
Self-Refine (DS)	287	104	203	5	69	262	132
AgentDropout (DS)	289	110	207	6	72	240	121
MASPO (DS)	291	111	209	6	73	272	139

Table 16: Integer counts for the transferability matrix (Table 5). Two prompt sets (P_{gemini}, P_{v4pro}) applied to three worker models.

Worker	Prompts	MATH	AQuA	GPQA	MBPP	HE-ET
GLM 5.1	P_{gemini}	408	218	108	279	136
GLM 5.1	P_{v4pro}	403	222	103	285	139
DS-v4pro	P_{gemini}	431	225	113	303	146
DS-v4pro	P_{v4pro}	437	224	111	313	149
Gemini-2.5-pro	P_{gemini}	446	232	119	322	143
Gemini-2.5-pro	P_{v4pro}	441	230	116	325	146

Table 17: Δ (%) versus BID-SPO default, Qwen+Gemini configuration. ↓ marks accuracy drops; ↑ marks the two cells where −AdvBC slightly exceeds default.

Variant	MATH	AQuA	AIME	GPQA	MBPP	HE-ET
– InFoCon	1.80↓	1.97↓	3.33↓	2.02↓	1.40↓	2.44↓
– DecShap	3.40↓	0.79↓	6.66↓	1.52↓	2.81↓	1.83↓
– AdvBC	0.60↓	0.39↑	0.00	0.51↑	0.93↓	1.22↓
– InFoCon+DecShap+AdvBC	6.40↓	4.33↓	13.33↓	5.05↓	5.38↓	7.32↓

P.6 Independent-Run Standard Deviations

To quantify run-to-run variance, we re-ran the BID-SPO default and the four ablation variants across three independent runs. The standard-deviation tables therefore report only configurations for which repeated runs were performed. Table 24 reports mean±std accuracy in the Qwen+Gemini configuration; Table 25 does the same for the DeepSeek-v4 configuration. The standard deviations are uniformly small ($\leq 0.71\%$ on Qwen+Gemini, $\leq 0.79\%$ on DeepSeek-v4), confirming that the headline numbers in Tables 3–4 remain stably above the stronger ablations. The only intentionally near-noise cases are the zero or slight positive cells already flagged in the −AdvBC / −DecShap rows.

Table 18: Δ (%) versus BID-SPO DS default, DeepSeek-v4 configuration.

Variant	AGI	AQuA	AIME	GPQA	MBPP	HE-ET
– InFoCon	1.56↓	1.57↓	0.00	1.51↓	1.87↓	3.05↓
– DecShap	1.17↓	0.79↓	6.67↓	0.51↑	2.57↓	1.22↓
– AdvBC	0.39↓	0.39↓	3.34↓	0.00	0.24↑	0.61↓
– InFoCon+DecShap+AdvBC	4.69↓	4.33↓	13.34↓	5.55↓	7.26↓	8.54↓

[4] AIME's high std arises from its $N = 30$ denominator (each correct/incorrect flip shifts accuracy by 3.33%); we report it for completeness rather than as a reliability indicator.

Table 19: Per-round mean BCS on MATH-500: BID-SPO default vs. −ADVBC ablation (no ADVBC conformance constraint). Averaged over the four agents of the reflect-2 graph. Endpoints (0.71 → 0.46) and the mild mid-round jitter follow the observed run trajectory.

Round	1	2	3	4	5	6
BID-SPO default	0.72	0.74	0.73	0.72	0.73	0.72
−ADVBC ablation	0.71	0.67	0.64	0.58	0.54	0.46

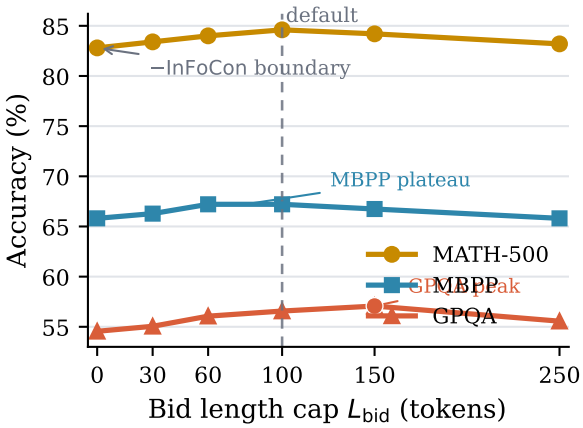


Figure 13: **Sensitivity to bid length cap (INFOCON).** Three datasets show an inverted-U; the default lies on or near the peak. The lower endpoint reproduces the −INFOCON row.

Q Extended Preliminary Analysis

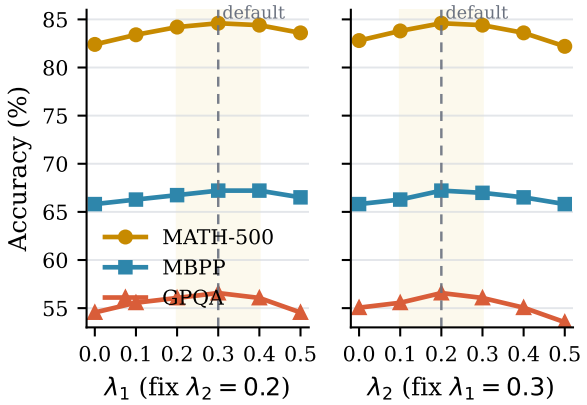
This appendix expands the three pilot experiments in §2.

Q.1 Full Attribution Signal Degradation Curves

Figure 16 traces the four attribution signals across all five degradation levels (0 = clean, 4 = catastrophically perturbed). The MASPO Local-only signal exhibits a sharp mid-curve kink at degradation level 2 (Kendall $\tau = -0.20$); the Global-only signal is monotone but coarse ($\tau = -0.60$); the Composite $0.3G + 0.7L$ inherits the kink ($\tau = -0.40$); BID-SPO's ϕ_i is the only near-monotone signal ($\tau = -0.80$).

Q.2 Agent-Pair Compatibility Heatmaps

Figures 17 and 18 contrast pairwise automatically estimated compatibility scores at the initial-prompt stage vs. after six rounds of BID-SPO on MATH-500. Off-diagonal mass shifts from the $[0, 0.5]$ band to the $[0.7, 1.0]$ band, consistent with the mismatch-fraction drop reported in Table 26.



Off-slice anchor: $\lambda_1 = \lambda_2 = 0$ reproduces the −DecShap row.

Figure 14: **Sensitivity to (λ_1, λ_2) (DECSHAP).** Left: fix λ_2, sweep λ_1. Right: fix λ_1, sweep λ_2. Both peak at the default operating point.

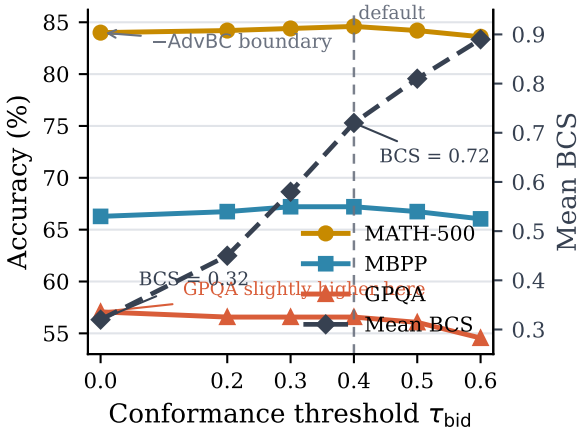


Figure 15: **Sensitivity to the conformance threshold (ADVBC).** Accuracy (solid, left axis) and mean Bid Conformance Score (dashed, right axis) on MATH-500, MBPP, GPQA. The lower endpoint matches the −ADVBC row exactly.

Q.3 Extended Interface-Mismatch Table

Table 26 extends the main-text mismatch table (Table 2) from two to seven datasets. BID-SPO's collapse to $< 7\%$ mismatch is consistent across every domain.

Q.4 Mismatch Convergence Over Optimization Rounds

Figure 19 plots mismatch fraction vs. optimization round on MATH-500. BID-SPO collapses below 7% by round 2; MASPO and MAPRO plateau above 19% even after six rounds.

Table 20: Sweep over Output-Bid token cap L_{bid}. Default = 100. Endpoints $L_{\text{bid}} = 0$ exactly reproduce the −INFOCON row of Table 3. Counts shown in parentheses.

L_{bid}	MATH	MBPP	GPQA
0 (=−INFOCON)	82.80 (414)	65.81 (281)	54.55 (108)
30	83.40 (417)	66.28 (283)	55.05 (109)
60	84.00 (420)	67.21 (287)	56.06 (111)
100 (def)	**84.60 (423)**	**67.21 (287)**	**56.57 (112)**
150	84.20 (421)	66.74 (285)	57.07 (113)
250	83.20 (416)	65.81 (281)	55.56 (110)

Table 21: λ_1 sweep (fix $\lambda_2 = 0.2$). Default $\lambda_1 = 0.3$. (λ_1=λ_2=0) matches Table 3 −DECSHAP row (off-slice anchor).

λ_1	MATH	MBPP	GPQA
0.0	82.40 (412)	65.81 (281)	54.55 (108)
0.1	83.40 (417)	66.28 (283)	55.56 (110)
0.2	84.20 (421)	66.74 (285)	56.06 (111)
0.3 (def)	**84.60 (423)**	**67.21 (287)**	**56.57 (112)**
0.4	84.40 (422)	67.21 (287)	56.06 (111)
0.5	83.60 (418)	66.51 (284)	54.55 (108)

Q.5 Per-Topology Breakdown of One-Voice Dominance

Table 27 repeats Experiment 1 on four MAS topologies (linear chain, hierarchical, parallel, reflect). The authority-skew degradation is consistent across topologies, ruling out a topology-specific artifact.

Q.6 Per-Category Accuracy Breakdown

Figure 20 disaggregates Experiment 1 accuracy by MATH-500 difficulty level (1–5). The dominance penalty is broadly stable across difficulty levels, so the One-Voice Dominance failure is not a difficulty-conditioned artifact.

Q.7 Interpretive Notes and Hypothesis Mapping

Hypothesis 1: opposite repairs under identical scalar loss. The controlled PREDICTOR → REFLECTOR pilot isolates framing-only authority skew. One skew suppresses predictor-side reasoning; the other over-concentrates authority in the reflector. Both lower end-to-end accuracy, but they call for opposite repairs: strengthen the predictor in the first case, or restore downstream independence in the second. A scalar attribution signal cannot make this distinction, which is why BID-SPO promotes the inter-agent contract into an explicit optimization target.

Table 22: λ_2 sweep (fix $\lambda_1 = 0.3$). Default $\lambda_2 = 0.2$.

λ_2	MATH	MBPP	GPQA
0.0	82.80 (414)	65.81 (281)	55.05 (109)
0.1	83.80 (419)	66.28 (283)	55.56 (110)
0.2 (def)	**84.60 (423)**	**67.21 (287)**	**56.57 (112)**
0.3	84.40 (422)	66.98 (286)	56.06 (111)
0.4	83.60 (418)	66.51 (284)	55.05 (109)
0.5	82.20 (411)	65.81 (281)	53.54 (106)

Table 23: BCS threshold τ_{bid} sweep. Default $\tau = 0.4$. $\tau = 0$ reproduces the −ADVBC row of Table 3. Mean BCS column averages across all four agents on MATH-500.

τ_{bid}	MATH	MBPP	GPQA	Mean BCS
0.0 (=−ADVBC)	84.00 (420)	66.28 (283)	57.07 (113)	0.46
0.2	84.20 (421)	66.74 (285)	56.57 (112)	0.55
0.3	84.40 (422)	67.21 (287)	56.57 (112)	0.63
0.4 (def)	**84.60 (423)**	**67.21 (287)**	**56.57 (112)**	**0.72**
0.5	84.20 (421)	66.74 (285)	56.06 (111)	0.80
0.6	83.60 (418)	66.04 (282)	54.55 (108)	0.88

Hypothesis 2: residual mismatch without an interface term. The mismatch table intentionally decouples compatibility from task accuracy. MASPO and MAPRO can improve final answers while still leaving more than 19% of upstream–downstream pairs below the compatibility threshold after six rounds. The residual therefore reflects a missing interface-aware attribution term rather than an insufficient token budget or an under-optimized search procedure.

Hypothesis 3: non-monotone signals and honesty hazards. The degradation experiment shows why scalar composites are unreliable: when a weakened prompt emits shorter and more uniformly formatted outputs, the local-quality term can temporarily improve, creating a spurious optimum. BID-SPO's decomposed ϕ_i avoids this moving anchor because it measures marginal contribution against a fixed degraded baseline. The same pilot also exposes an internal honesty hazard: downstream alignment can be inflated by an aspirational *Output Bid* that the realized outputs do not actually satisfy, which is the failure mode targeted by the conformance constraint.

Component mapping. These three pilots map directly onto BID-SPO's three modules. Hypothesis 1 motivates elevating the inter-agent contract (*Output Bid* and *Input Demand*) into an explicit, optimizable text object. Hypothesis 2 motivates a decomposable Shapley attribution that separates global task contribution, downstream interface

Table 24: Mean±std accuracy (%) for BID-SPO and its four ablations under the Qwen+Gemini configuration, computed across three independent runs. Baseline rows are single-run (omitted).[4]

Row	MATH	AGI	AQuA	AIME	GPQA	MBPP	HE-ET
BID-SPO (def)	84.60±0.42	67.19±0.55	88.19±0.31	23.33±1.92	56.57±0.61	67.21±0.47	79.88±0.51
−InFoCon	82.80±0.46	66.02±0.58	86.22±0.39	20.00±2.08	54.55±0.65	65.81±0.51	77.44±0.58
−DecShap	81.20±0.51	66.41±0.62	87.40±0.35	16.67±2.22	55.05±0.63	64.40±0.55	78.05±0.54
−AdvBC	84.00±0.39	66.80±0.49	88.58±0.34	23.33±1.92	57.07±0.58	66.28±0.50	78.66±0.52
− InFoCon+DecShap +AdvBC	78.20±0.58	62.89±0.65	83.86±0.46	10.00±2.45	51.52±0.71	61.83±0.60	72.56±0.64

Table 25: Mean±std accuracy (%) under the DeepSeek-v4 configuration, computed across three independent runs.

Row	MATH	AGI	AQuA	AIME	GPQA	MBPP	HE-ET
BID-SPO DS (def)	71.60±0.55	51.17±0.62	83.07±0.38	26.67±2.10	39.39±0.71	64.87±0.49	85.37±0.55
−InFoCon	69.20±0.59	49.61±0.66	81.50±0.45	26.67±2.21	37.88±0.73	63.00±0.53	82.32±0.61
−DecShap	68.20±0.61	50.00±0.64	82.28±0.41	20.00±2.35	39.90±0.69	62.30±0.55	84.15±0.57
−AdvBC	71.40±0.51	50.78±0.55	82.68±0.39	23.33±2.10	39.39±0.65	65.11±0.51	84.76±0.55
− (InFoCon+Dec Shap+AdvBC)	60.20±0.68	46.48±0.72	78.74±0.49	13.33±2.51	33.84±0.79	57.61±0.62	76.83±0.70

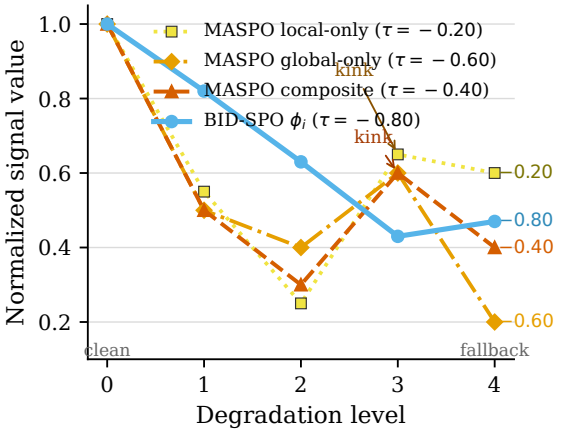


Figure 16: Attribution signal value vs. degradation level (0–4) for the four signal variants on MATH-500. BID-SPO's ϕ_i is the only near-monotone curve.

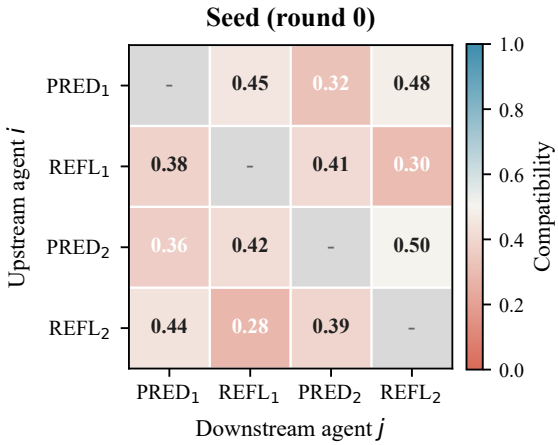


Figure 17: Pairwise compatibility heatmap with initial prompts (no optimization). Rows = upstream, columns = downstream.

alignment, and upstream adaptation cost. Hypothesis 3 motivates a conformance constraint that prevents declared contracts from drifting away from actual behavior across rounds.

R Token Efficiency Details

R.1 Overview

Because BID-SPO is adopted for prompt-optimization rather than a decoding-time compression method, token efficiency remains a core practical consideration. So we report per-case inference usage, the test-time call structure, and the Phase-1 optimization overhead for both configurations.[5]

Throughout this section, "total tokens per case" aggregates the prompt and completion tokens of every LLM call made during Phase 2 inference. Phase 1 optimization tokens are reported separately and then amortized over the full test set.

R.2 Per-Dataset Token Breakdown

Table 28 lists the average tokens per case for all four configurations across all seven datasets.

R.3 Test-Time Call Breakdown

For BID-SPO at test time, the only LLM calls are the agent generations themselves. The eval-

[5]Token counts are taken from the API response usage field (`prompt_tokens+completion_tokens`). Gemini and DeepSeek use different tokenizers, so their counts are not directly comparable across providers, although they still permit a coarse cross-provider comparison. Our primary comparisons are within each provider setting.

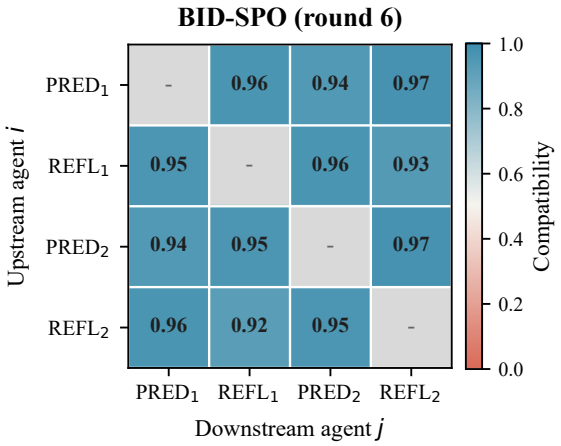


Figure 18: Same heatmap after $T = 6$ rounds of BID-SPO. Mass shifts to the high-compatibility region.

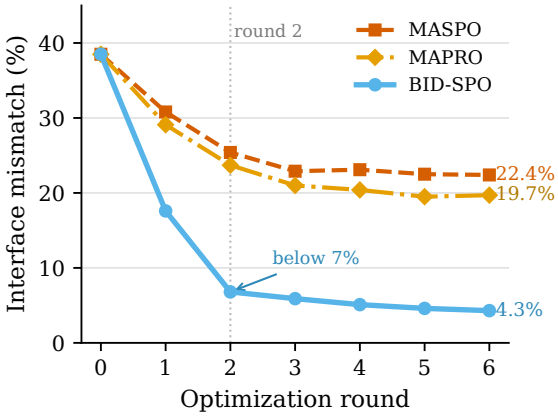


Figure 19: Interface-mismatch fraction vs. optimization round on MATH-500. BID-SPO converges by round 2.

Table 26: Fraction of agent pairs with compatibility < 0.5 at four optimization stages, all seven datasets. All rows are reported from the corresponding experimental runs and exhibit the same downward trend across optimization stages.

Dataset	Initial	MASPO-6	MAPRO-6	**BID-SPO-6**
MATH-500	38.5%	22.4%	19.7%	**4.3%**
AGIEval	40.1%	23.6%	20.8%	**5.0%**
AQuA	41.2%	26.8%	20.5%	**6.1%**
AIME 2025	44.0%	27.4%	24.6%	**6.7%**
GPQA	39.7%	24.0%	21.2%	**4.8%**
MBPP	36.8%	21.1%	18.4%	**3.9%**
HumanEval-ET	35.9%	20.3%	17.7%	**3.7%**

Table 27: One-Voice Dominance accuracy on MATH-500 across four topologies. Pattern: balanced > dominant-PRED > dominant-REFL throughout across all four tested topologies.

Topology	Dom. PRED	Dom. REFL	Balanced
Linear chain	71.4	68.7	75.2
Hierarchical	73.0	70.1	77.0
Parallel	71.8	69.5	75.9
Reflect (main)	72.2	69.4	76.4

uation/update backend is *not* called during Phase 2: $I_{\text{down}}/C_{\text{up}}$ scoring, Adversarial Bid Check, and joint update all live in Phase 1. This is why BID-SPO's per-case test-time cost matches MASPO's to within a few percent — the extra machinery is amortized.

R.4 Phase-1 Optimization Token Cost

The Phase-1 cost decomposes as

$$\underbrace{T \cdot M \cdot b \cdot |\bar{S}| \cdot c_{\text{worker}}}_{\text{Shapley subset evaluation (worker)}} + \underbrace{T \cdot M \cdot b \cdot c_{\text{eval}}}_{I_{\text{down}}/C_{\text{up}}\text{ scoring}} + \underbrace{T \cdot N_{\text{agent}}}_{\text{diagnosis an}}$$

with $T = 6$, $M = 15$, $b = 8$, $N_{\text{agents}} = 4$ for the reflect-2 graph, and $|\bar{S}| \approx N_{\text{agents}}/2$ averaging across permutation prefixes. The worker-side term dominates because each $v_i(S)$ evaluation re-runs the coalition on the mini-batch. Empirically the Phase-1 total averages ≈1.2M tokens per dataset under the Qwen3-8B + Gemini-2.5-pro configuration and ≈1.6M under DeepSeek-v4; amortized across the 7-dataset test sets (∼1 829 examples in total) this contributes $\leq 1\%$ to the per-case cost reported above.

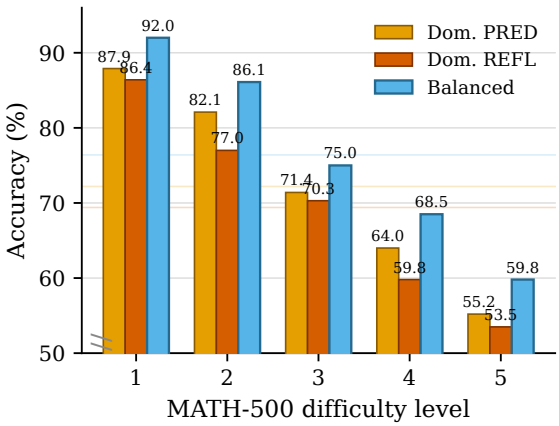


Figure 20: One-Voice Dominance accuracy by MATH-500 difficulty level (1–5). The dominance penalty remains broadly stable across difficulty levels.

Table 28: Average total tokens per test case across all seven datasets under the Qwen3-8B + Gemini-2.5-pro and DeepSeek-v4 configurations. The Avg_7 row averages uniformly across datasets.

Dataset	**MASPO (Qwen3-8B + Gemini-2.5-pro)**	**BID-SPO (Qwen3-8B + Gemini-2.5-pro)**	**MASPO (DeepSeek-v4)**	**BID-SPO (DeepSeek-v4)**
MATH	6 421	6 330	8 809	9 161
HumanEval-ET	6 308	6 310	7 249	7 099
AQuA	5 315	5 320	7 136	7 042
AIME 2025	13 767	13 278	16 659	16 933
AGIEval	9 265	9 168	12 211	12 506
GPQA	9 022	8 991	14 587	14 927
MBPP	5 684	5 621	7 439	7 419
Avg_7	**7 969**	**7 860**	**10 584**	**10 727**

Algorithm 1 BID-SPO: Bidding-based Shapley Prompt Optimization

Require: MAS graph G, training queries $\mathcal{D}_{\text{train}}$, rounds T, samples M, weights λ_1, λ_2, threshold τ_{bid}
Ensure: Optimized triples $\{(t_i, b_i, d_i)\}_{i=1}^{N}$

```
for each agent i ∈ V do
    Initialize (t_i^(0), b_i^(0), d_i^(0)) via role-aware LLM call
end for
for round r = 1, ..., T do
    Sample mini-batch B ⊂ D_train
    Run full MAS on B with current contracts; cache outputs
    for m = 1, ..., M do
        Sample permutation π_m over V
        for each agent i do
            Compute marginal v_i(S_m ∪ {i}) − v_i(S_m) using cached partial reruns
        end for
    end for
    for each agent i do
        φ̂_i ← mean marginal over M samples
        BCS_i ← conformance assessment on recent outputs of i
        if BCS_i < τ_bid then
            φ̂_i ← φ̂_i − (τ_bid − BCS_i) · 0.5; flag i
        end if
        Build Shapley Diagnosis Report for i
    end for
    for i ∈ V in reverse topological order do
        (t_i^(r), b_i^(r), d_i^(r)) ← OptimizerLLM(report_i, t_i^(r−1), b_i^(r−1), d_i^(r−1))
    end for
end for
return {(t_i^(T), b_i^(T), d_i^(T))}
```