STAR: TRAJECTORY-AWARE SENTENCE ROUTING FOR EFFICIENT MULTI-LLM ENSEMBLE

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-LLM ensembles exploit the complementary capabilities of heterogeneous LLMs to improve inference without further increasing model scale. However, existing ensemble methods incur substantial computational waste by invoking multiple models at every generation step, while their reliance on local signals can result in suboptimal reasoning trajectories. In this work, we find that multi-LLM integration is sparse yet consequential at the sentence level: semantic disagreement arises only at a few critical states, while decisions at these states can shape subsequent reasoning trajectories beyond what local signals capture. To exploit this, we propose *STAR*, a sentence-level trajectory-aware routing framework that triggers multi-LLM integration only when needed and routes candidates according to their long-term trajectory value. STAR combines Sentence-level Disagreement Prediction (SDP) to predict cross-model semantic disagreement before candidate generation with Tree-Advantage Routing Optimization (TARO) to evaluate alternative trajectories and optimize routing using counterfactual advantages. Extensive experiments show that STAR achieves state-of-the-art accuracy among multi-LLM ensemble methods while reducing inference cost by up to 88.04%. Our code is available in the supplementary material.

1 INTRODUCTION

The rapid progress of large language models (LLMs) has been largely driven by increasing model scale and computational budgets. Yet continued scaling is increasingly costly, and no single model consistently excels across diverse tasks and reasoning demands (Tang et al., 2026). Multi-LLM ensembles offer a promising alternative to this ever-escalating race for scale by coordinating heterogeneous LLMs to jointly address downstream queries (Zhang et al., 2026; Tang et al., 2026). By exploiting their complementary capabilities, such systems can achieve stronger and more flexible performance without relying solely on further model scaling.

Existing multi-LLM ensemble methods can be broadly categorized by when ensemble decisions are made: pre-generation, post-generation, and during-generation ensembles. Pre-generation methods select a single model before inference to generate the response (Wang et al., 2026a), whereas post-generation methods aggregate responses produced by multiple models (Tang et al., 2026). Both make ensemble decisions outside the generation process and therefore cannot adapt to evolving reasoning states. During-generation ensembles instead integrate multi-model collaboration into autoregressive decoding and can be further divided into token-level and process-level ensembles. Token-level methods combine next-token predictions from multiple models at each decoding step, but are limited to models with shared tokenizers due to the difficulty of aligning heterogeneous vocabularies and tokenization schemes (Yun et al., 2026). In contrast, process-level ensembles make routing decisions over higher-level semantic units rather than individual tokens. A common choice is the sentence level, where each model generates a complete candidate sentence before the ensemble selects the continuation (Park et al., 2025).

However, existing process-level ensemble methods invoke multiple models at every generation step, resulting in substantial redundant computation. Prior studies Kuhn et al. (2023); Fu et al. (2025) suggest that candidates with different wording may still be semantically equivalent, whereas genuine semantic differences can steer subsequent reasoning toward different trajectories. We define sentence-level generation positions exhibiting such cross-model semantic disagreement as critical

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

positions. Our empirical analysis shows that these positions are sparse in practice. Specifically, we randomly sample 1,000 tasks from GSM8K and MBPP and collect multiple sentence-level states along their reasoning trajectories. At each state, three heterogeneous LLMs independently generate the next candidate sentence from the same prefix, and GPT-5.5 serves as an independent semantic judge to assess their semantic consistency. We find that only about 12.6% of sentence-level positions exhibit disagreement, while the remaining 87.4% are semantically consistent.

Although critical positions are sparse, decisions made at these states can substantially shape subsequent reasoning trajectories. Since LLM generation is autoregressive, a locally preferred continuation need not lead to the best overall trajectory (Liao et al., 2026). Yet existing methods typically select candidates using local signals such as perplexity (Xu et al., 2025b) or process rewards (Park et al., 2025), which may fail to capture the downstream consequences of a current choice. Our experiments in Section 5.3 further support this observation: replacing trajectory-aware selection with one-step perplexity-based selection reduces average accuracy from 71.74% to 64.81%. Taken together, these observations motivate the central research question of this work: Can we learn a dynamic ensemble policy that invokes multiple models only at positions likely to exhibit semantic disagreement, while selecting candidates based on their long-term trajectory value?

To answer this question, we propose *STAR*, a novel **S**entence-level **T**rajectory-**A**ware **R**outing framework for multi-LLM ensembles. STAR triggers ensemble only at semantic disagreement positions likely to alter subsequent reasoning trajectories and routes candidates based on their long-term value. STAR consists of two core components: **Sentence-level Disagreement Prediction (SDP)** and **Tree-Advantage Routing Optimization (TARO)**. First, SDP leverages the base model's hidden representations at sentence boundaries to predict cross-model semantic disagreement before candidate generation, thereby avoiding redundant computation at non-critical states. Second, once ensemble is triggered, TARO performs budget-constrained tree rollouts to evaluate candidate continuations and constructs counterfactual advantage signals from their relative values under the same reasoning prefix to optimize the routing policy. Together, these components enable STAR to avoid unnecessary multi-model generation while optimizing candidate selection beyond local signals toward complete reasoning trajectories. Our contributions are summarized as follows:

- **New Perspective.** We revisit during-generation multi-LLM ensembling through the lens of sentence-level semantic disagreement, reframing it from local candidate selection at each step as sequential decision-making over sparse critical states along the reasoning trajectory.
- **Core Algorithm.** We propose *STAR*, which combines SDP to identify when ensemble is necessary with TARO for long-term candidate evaluation and fine-grained credit assignment via budget-constrained tree search and counterfactual advantage learning.
- **SOTA Performance.** Extensive experiments show that STAR achieves the highest average accuracy among multi-LLM ensemble methods with up to 88.04% lower inference cost. Moreover, by coordinating heterogeneous 2B–8B LLMs, STAR surpasses a 70B standalone model at 93.34% lower cost, demonstrating collective intelligence beyond model scale.

2 Related Work

Ensemble Before Inference. Before generation, existing approaches typically route each input query to an LLM based on its predicted suitability. Early methods such as Zooter (Lu et al., 2024) and ICL-Router (Wang et al., 2026a) focus on estimating model competence for individual queries, while subsequent approaches incorporate inference cost and other deployment objectives into routing decisions (Ong et al., 2025; Feng et al., 2025; Woisetschläger et al., 2026). More recent work further combines model routing with test-time scaling by jointly determining the model and inference budget (Ding et al., 2025; Zhang et al., 2026). Despite their efficiency, such methods commit to a routing decision before generation and therefore cannot adapt model selection as the reasoning state evolves.

Ensemble After Inference. After generation, existing approaches typically make cross-model decisions over complete responses through cascading, selection, or aggregation. AutoMix (Aggarwal et al., 2024) and Semantic Agreement (Soiffer et al., 2025) determine whether to invoke stronger models based on response reliability, whereas SLM-MUX (Wang et al., 2026b), SMoA (Li et al., 2025), and JiSi (Tang et al., 2026) select or aggregate responses produced by multiple models. Be-

cause cross-model decisions are made only after complete responses are available, these approaches cannot intervene in intermediate reasoning states and may incur redundant computation from generating full responses before selection or aggregation.

Ensemble During Inference. At the token level, multiple LLMs collaborate at individual decoding steps through probability aggregation or model selection. GAC (Yu et al., 2024) combines token-level probability distributions, while FusionRoute (Xiong et al., 2026) dynamically selects experts during decoding. To support heterogeneous LLMs, recent work further introduces vocabulary-alignment and logit-fusion strategies (Yao et al., 2025). Despite their fine-grained adaptivity, token-level methods require frequent cross-model coordination, while heterogeneous models additionally depend on explicit vocabulary or tokenizer alignment, resulting in considerable inference overhead.

At a coarser granularity, process-level approaches coordinate LLMs over semantically meaningful units such as sentences, text spans, or reasoning steps. Cool-Fusion (Liu et al., 2025) and SweetSpan (Xu et al., 2025a) select among model-generated segments through cross-model evaluation, while AdaFuse (Cui et al., 2026) adaptively expands candidate generation according to decoding uncertainty. For multi-step reasoning, LE-MCTS (Park et al., 2025) performs process-reward-guided tree search, and CBS (Xu et al., 2025b) combines multi-model step generation with consensus-based verification. Although these methods exploit richer semantic units, they often incur redundant computation by generating multiple candidates at each generation step and rely on local signals such as perplexity, process rewards, or consensus scores, which may fail to capture the downstream impact of each choice on the overall reasoning trajectory.

3 Problem Formulation

We consider sentence-level ensemble among K heterogeneous LLMs, denoted by $\mathcal{M} = M_1, \ldots, M_K$. A sentence-level generation unit is defined as a contiguous token sequence terminated by the first sentence-ending punctuation mark, newline, or end-of-sequence token, with a maximum length of 64 tokens. For simplicity, we refer to each unit as a sentence hereafter.

Given a query x, the system autoregressively generates a response $Y = (s_1, \ldots, s_T)$, where s_t denotes the sentence generated at step t and $h_t = (x, s_{<t})$ is the preceding context. At each sentence boundary, the system determines whether multi-model ensemble is required. At agreement states, the low-cost base model $M_b \in \mathcal{M}$ generates the next sentence, where b denotes its index in the candidate model pool. Otherwise, all K models are invoked to generate candidate continuations $\mathcal{S}_t = s_t^{(1)}, \ldots, s_t^{(K)}$ from the shared context h_t, and a routing policy selects one candidate $s_t^{(k_t)}$ as the next sentence. Repeating this process until termination induces a generation trajectory τ.

We formulate routing as a hierarchical decision process, where a gating function determines whether to trigger ensemble and a selection policy chooses among candidates when triggered. The objective is to maximize final response quality while minimizing inference cost:

$$\max_{\pi} \ \mathcal{J}(\pi) = \mathbb{E}_{\tau \sim \pi} \left[V(x, Y) - \lambda \sum_{t=1}^{T} C_t \right], \tag{1}$$

where $V(x, Y)$ measures the quality of the final response, C_t denotes the inference cost at sentence step t, and λ controls the trade-off between quality and cost.

4 Method

In this section, we introduce **STAR**, a sentence-level trajectory-aware routing framework for efficient multi-LLM ensemble. STAR contains two components: **Sentence-Level Disagreement Predictor (SDP)**, which predicts whether multi-model ensemble is required, and **Tree-Advantage Routing Optimization (TARO)**, which selects trajectory-aware candidate continuations. During training, SDP is optimized using offline disagreement labels, while TARO is independently trained with trajectory-level supervision from disagreement-triggered rollouts. During inference, SDP invokes the TARO-trained routing policy only at predicted disagreement states. Otherwise, the lowest-cost model directly generates the next sentence, avoiding unnecessary multi-model generation.

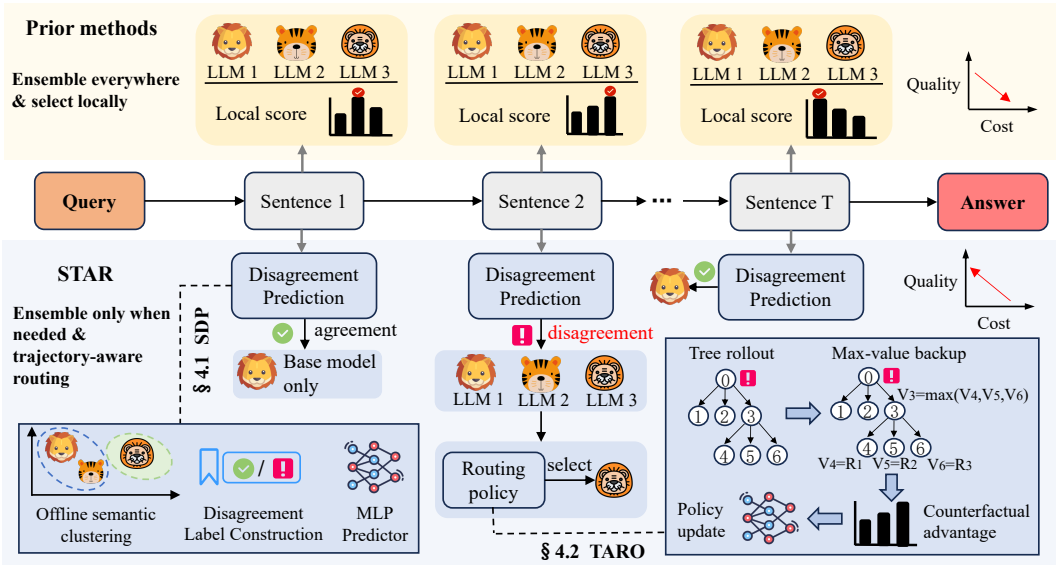

Figure 1: **Overview of the STAR framework.** STAR triggers multi-LLM integration only at predicted semantic disagreement states and performs trajectory-aware routing at critical sentence boundaries. SDP determines when integration is needed, while TARO learns the routing policy via tree rollout and counterfactual advantage.

4.1 Sentence-Level Disagreement Prediction

Invoking all candidate models at every sentence boundary introduces substantial redundant computation, since their next-sentence continuations are often semantically consistent. We predict whether candidate models will disagree before explicitly generating their continuations. Specifically, we construct disagreement labels offline based on the semantic consistency among candidate models and train a lightweight predictor to identify ensemble states.

Disagreement Label Construction. We employ a strong LLM as an offline semantic annotator following the LLM-as-a-judge paradigm. We collect sentence-boundary states along trajectories generated by the base model M_b. Given the shared prefix h_t and candidate continuations $\mathcal{S}_t$, the annotator groups candidates according to their underlying semantics. Candidates with the same semantics are assigned to the same cluster regardless of lexical variations, while semantically different continuations form different clusters. Let $c_t^{(k)}$ denote the semantic cluster assigned to $s_t^{(k)}$. We define the disagreement label as:

$$d_t = \mathbb{I}\Big(\Big|\text{Unique}\left(\{c_t^{(k)}\}_{k=1}^{K}\right)\Big| > 1\Big), \tag{2}$$

where $d_t = 0$ indicates semantic agreement among all candidate models, while $d_t = 1$ indicates the existence of multiple semantic clusters. Both candidate generation and annotation are performed offline and introduce no inference-time overhead.

To validate the reliability of the LLM-generated disagreement labels, we sample 500 sentence-boundary states and manually annotate them. The human annotations agree with the LLM-generated labels on 92.2% of the samples, demonstrating consistency between LLM and human judgments.

Agreement Predictor. We train a lightweight predictor to estimate cross-model agreement without explicitly invoking all candidate models. Although disagreement is defined by the semantic consistency among multiple candidate continuations, we use the lowest-cost model M_b as a proxy and extract its last-layer hidden state $\mathbf{z}_t \in \mathbb{R}^D$ at each sentence boundary. Following prior studies that leverage LLM hidden representations for prediction (Xuan et al., 2025; Zhou et al., 2024), the predictor learns to infer future ensemble states from $\mathbf{z}_t$. The predictor f_θ is a lightweight MLP that maps $\mathbf{z}_t$ to the agreement probability $p_t^{\text{agr}} = P_\theta(d_t = 0 \mid \mathbf{z}_t)$. The base model M_b remains frozen,

and only the predictor parameters θ are optimized using the offline disagreement labels. To address the imbalance between agreement and disagreement states, we minimize the class-weighted binary cross-entropy loss:

$$\mathcal{L}_{\text{agr}} = -\frac{1}{N}\sum_{t=1}^{N}\left[w_0(1-d_t)\log p_t^{\text{agr}} + w_1 d_t \log(1-p_t^{\text{agr}})\right], \tag{3}$$

where w_0 and w_1 denote the class weights for agreement and disagreement states, respectively. During inference, ensemble is triggered when:

$$g_t = \mathbb{I}(p_t^{\text{agr}} < \tau), \tag{4}$$

where τ controls the aggressiveness of ensemble triggering: a larger τ invokes multi-model routing more frequently, while a smaller τ favors cost-efficient generation.

4.2 Tree-Advantage Routing Optimization

The routing decision at each disagreement state should account for its downstream impact, as local candidate quality may not reflect the final trajectory outcome. We therefore propose Tree-Advantage Routing Optimization, which explores alternative trajectories through disagreement-triggered tree rollouts, estimates long-term candidate values via max-value backup, and learns routing policies with counterfactual advantages.

MDP Formulation. We formulate sentence-level multi-LLM ensemble as a finite-horizon Markov decision process (MDP). At step t, the **state** is the shared context $h_t = (x, s_{<t})$, and the **action** selects one candidate sentence $a_t = k_t$ from $\mathcal{S}_t$, resulting in $s_t = s_t^{(k_t)}$ and **transition** $h_{t+1} = (h_t, s_t)$. The **reward** decomposes into inference cost and final response quality:

$$r_t = -\lambda C_t + \mathbb{I}(t = T)V(x, Y). \tag{5}$$

Disagreement-Triggered Tree Rollout. Once semantic disagreement is detected, selecting a candidate based only on local information can be suboptimal, since different choices may alter future contexts and lead to different final outcomes. We therefore construct a trajectory tree to evaluate the long-term effects of candidate decisions.

At a disagreement state with shared prefix h_t, all models generate candidate sentences $\mathcal{S}_t$. Each candidate forms a child node, and future generation proceeds recursively: agreement states are advanced by the low-cost model, while disagreement states trigger further branching. Thus, the tree expands only at semantically divergent positions and sparsely explores alternative trajectories. We reuse the semantic clustering criterion from Sec. 4.1 to merge equivalent sibling candidates and retain the lowest-cost representative. To prevent exponential growth, each rollout is constrained by a trajectory budget B:

$$|\mathcal{L}(\mathcal{T})| \le B, \tag{6}$$

where $\mathcal{L}(\mathcal{T})$ denotes the leaf nodes of trajectory tree $\mathcal{T}$. When the budget is exhausted, further branching is terminated and the remaining trajectories are completed by the low-cost model. Each leaf trajectory is evaluated using Eq. 1, and the obtained returns are propagated backward through value backup.

Max-Value Backup. The rollout provides returns only for complete trajectories, whereas routing decisions are made at intermediate nodes. We therefore propagate leaf returns backward to estimate the long-term value of each candidate.

Unlike prior tree-based approaches (Liao et al., 2026; Kim et al., 2026) that average downstream values, we adopt maximum backup for intermediate nodes. A node is valuable if at least one downstream branch achieves a high return, whereas averaging may dilute promising branches with inferior continuations. Specifically, each leaf node is initialized with the return of its corresponding trajectory. For an internal node v with child set $\mathcal{C}(v)$, its value is recursively defined as:

$$V(v) = \max_{u \in \mathcal{C}(v)} V(u), \tag{7}$$

where an agreement state with a single child directly inherits its child's value. Applying this backup from leaves to the root yields trajectory-aware values for candidate selection.

Counterfactual Advantage Optimization. After value backup, each candidate branch at a disagreement state is assigned a long-term value. We transform these values into a local credit signal by comparing each candidate with the best alternative under the same shared prefix:

$$A_{\mathrm{cf}}(u) = V(u) - \max_{u' \in \mathcal{C}(v) \setminus \{u\}} V(u'), \tag{8}$$

where v denotes the corresponding disagreement node. This sibling-wise comparison measures the relative contribution of each routing decision and provides localized supervision, unlike trajectory-level credit assignment that assigns identical returns to all decisions along a rollout. Counterfactual advantages are computed only at disagreement nodes with multiple semantic branches. We optimize the routing policy using a PPO-style clipped objective (Schulman et al., 2017):

$$\mathcal{J}_{\mathrm{CAO}}(\theta) = \mathbb{E}\left[\min\left(r_t(\theta) A_t^{\mathrm{cf}}, \mathrm{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\, A_t^{\mathrm{cf}}\right)\right] + \alpha \mathcal{H}(\pi_\theta), \tag{9}$$

where $r_t(\theta) = \pi_\theta(a_t|h_t, \mathcal{S}_t) / \pi_{\theta_{\mathrm{old}}}(a_t|h_t, \mathcal{S}_t)$ is the policy ratio, ϵ is the clipping threshold, and α controls entropy regularization. We automatically adjust α toward a target entropy during training (Haarnoja et al., 2018).

5 Experiments

5.1 Setup

Models and Datasets. We evaluate our method on three heterogeneous open-source LLMs, including **Llama-3.1-8B-Instruct** (Grattafiori et al., 2024), **Gemma-2-2B-it** (Team et al., 2024), and **Mistral-7B-Instruct-v0.3** (Jiang et al., 2023). These models cover diverse model families and parameter scales and correspond to the latest publicly available versions at the time of our experiments. For in-distribution (ID) evaluation, we conduct experiments on five benchmarks spanning three task categories: (1) **Math**: GSM8K (Cobbe et al., 2021) and MATH500 (Hendrycks et al., 2021b); (2) **Code**: MBPP (Austin et al., 2021) and HumanEval (Chen et al., 2021); and (3) **Reasoning**: ARC-Challenge (Clark et al., 2018). For out-of-distribution evaluation, we additionally consider three unseen benchmarks: MathBench (Liu et al., 2024), APPS (Hendrycks et al., 2021a), and BIG-Bench Hard (BBH) (Suzgun et al., 2023).

Baselines. We compare STAR against a broad set of baselines: **(1) Static Baselines:** each participating LLM in the collaboration pool independently generates responses, serving as references for standalone performance. **(2) Single Large Model Baselines:** we additionally include two larger-scale LLMs, **Mistral-NeMo-12B-Instruct** (Mistral AI & NVIDIA, 2024) and **Llama-3.1-70B-Instruct** (Grattafiori et al., 2024), which are not involved in collaboration and serve as references for single-model performance. **(3) Ensemble Before Inference:** we include **Avengers** (Zhang et al., 2026), which performs query-level routing by selecting suitable LLMs based on input query semantics. **(4) Ensemble After Inference:** we compare with **SLM-MUX** (Wang et al., 2026b), which aggregates responses from multiple LLMs after independent generation. **(5) Ensemble During Inference:** we consider both token-level and process-level ensemble methods. Specifically, **FusionRoute** (Xiong et al., 2026) performs token-level routing among LLMs, while **AdaFuse** (Cui et al., 2026) and **CBS** (Xu et al., 2025b) represent process-level ensemble through adaptive ensemble decoding and collaborative search, respectively.

Metrics. We evaluate our method in terms of **Accuracy** and **Cost**. We report accuracy for math and reasoning tasks, and pass@1 for code generation. We measure the inference cost based on the latest input and output token prices of the evaluated models on OpenRouter (OpenRouter, 2026).

Implementation. Detailed experimental settings and implementation details are provided in the Appendix B. We have released the source code to facilitate reproducibility.

Table 1: Performance and inference cost comparison across five benchmarks. Best results are shown in **bold**, and second-best results are underlined.

Method	GSM8K	MATH500	MBPP	HumanEval	ARC-C	Avg. Acc.	Cost ($)
Candidate LLMs							
Gemma-2-2B-it	19.30	8.52	24.18	34.72	53.86	28.12	0.11
Mistral-7B-Instruct-v0.3	57.71	34.82	42.57	36.52	62.17	46.76	0.24
LLaMA-3.1-8B-Instruct	69.37	43.45	51.36	56.89	78.63	59.94	0.47
Single Large Model Baselines							
Mistral-NeMo-12B-Instruct	69.12	46.24	53.13	58.36	76.24	60.62	0.73
Llama-3.1-70B-Instruct	75.38	52.17	**70.84**	66.41	82.15	69.39	5.86
LLM Ensemble Baselines							
Avengers	69.85	44.63	54.72	57.18	78.42	60.96	1.24
SLM-MUX	72.54	46.82	58.35	60.27	79.11	63.42	2.35
FusionRoute	71.86	40.50	49.57	54.66	83.14	59.95	0.87
AdaFuse	75.32	49.16	62.21	67.74	**85.91**	68.07	0.75
CBS	75.26	51.34	60.87	67.92	80.47	67.17	3.26
STAR	**80.42**	**56.69**	68.51	**69.80**	83.26	**71.74**	**0.39**

5.2 Main Results

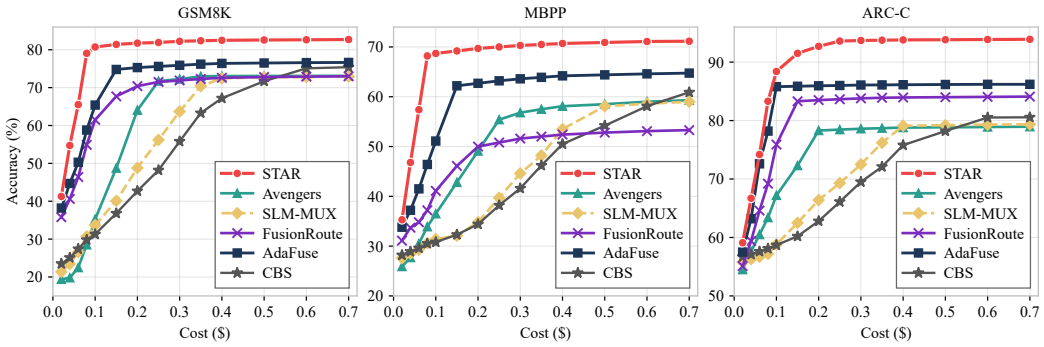

Figure 2: Accuracy–cost trade-off across three benchmarks. Higher accuracy at lower inference cost indicates better efficiency.

Superior Performance across Benchmarks. Table 1 compares the performance and inference cost across five benchmarks spanning mathematical reasoning, code generation, and general reasoning. STAR achieves the highest average accuracy of 71.74%, outperforming all individual candidate LLMs and all existing ensemble baselines. Compared with the strongest ensemble baseline (AdaFuse), STAR improves average accuracy by 3.67% while reducing inference cost by 48.0% ($0.75 $\rightarrow$ $0.39). STAR also achieves the best performance on GSM8K, MATH500, and HumanEval, while remaining competitive on MBPP and ARC-C. These results demonstrate that STAR achieves a better balance between predictive performance and inference efficiency, suggesting that its selective sentence-level ensemble strategy can exploit complementary model capabilities without incurring excessive ensemble cost.

Collective Intelligence beyond Larger Models. Table 1 further compares STAR with two substantially larger open-source LLMs that are not included in its candidate pool. STAR achieves an average accuracy of 71.74%, surpassing Mistral-NeMo-12B-Instruct and Llama-3.1-70B-Instruct by 11.12 and 2.35 percentage points, respectively. Notably, STAR outperforms the 70B model on four of the five benchmarks while reducing inference cost by 93.3% ($5.86 $\rightarrow$ $0.39). These results highlight the potential of *collective intelligence*: coordinating multiple smaller heterogeneous LLMs can outperform a much larger standalone model at substantially lower inference cost.

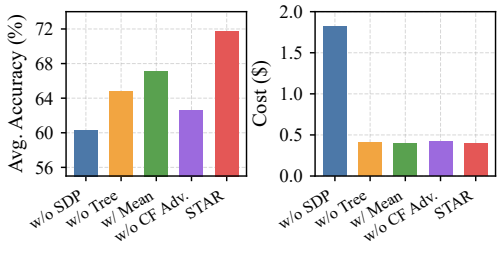

Figure 3: Ablation analysis on average accuracy (left) and inference cost (right).

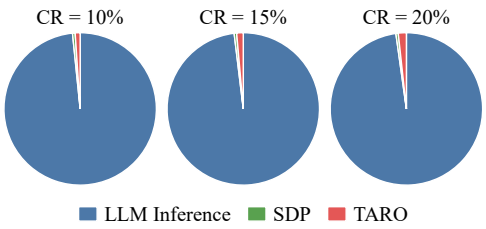

Figure 4: Runtime breakdown of STAR at different collaboration rates (CR).

Advancing the Pareto Frontier. Figure 2 illustrates the accuracy-cost trade-off on GSM8K, MBPP, and ARC-C under different inference budgets. Across all three benchmarks, STAR consistently achieves higher accuracy at the same cost, or comparable accuracy at substantially lower cost, establishing a stronger Pareto frontier than existing ensemble methods. Query-level routing methods such as Avengers remain relatively efficient, but their performance is constrained by making model-selection decisions only before generation. Post-inference aggregation methods such as SLM-MUX can improve accuracy through multiple complete responses, but incur rapidly increasing cost as all candidate models must finish generation. Token-level methods such as FusionRoute enable finer-grained collaboration, yet frequent multi-model coordination introduces considerable overhead. In contrast, STAR improves more rapidly in the low-cost regime and gradually saturates as the budget increases, indicating more effective use of additional inference budget.

5.3 Ablation Studies

We conduct four component-wise ablations to examine the major designs in STAR. For **w/o SDP**, we remove the disagreement predictor and retrain TARO with the same rollout budget $B = 32$. For **w/o Tree Rollout**, trajectory exploration is replaced by one-step selection using the continuation with the lowest perplexity. For **w/ Mean Backup**, max-value backup is replaced with mean-value backup. Finally, **w/o Counterfactual Advantage** replaces disagreement-aware counterfactual credit assignment with trajectory-level global credit assignment.

As shown in Fig. 3, all four variants reduce average accuracy from 71.74% to varying degrees. Removing SDP causes the largest drop to 60.26%, while also increasing the inference cost from \$0.39 to \$1.82. This indicates that SDP is not only an efficiency-oriented gate, but also helps concentrate the fixed rollout budget on informative disagreement states. Removing counterfactual advantage and tree rollout decreases the average accuracy to 62.59% and 64.81%, respectively, highlighting the importance of fine-grained credit assignment and long-horizon candidate evaluation. Mean backup causes a smaller drop to 67.13%, suggesting that trajectory exploration remains useful even with a less selective backup strategy. Detailed benchmark-wise results are reported in Appendix C.1.

5.4 System-Level Efficiency

Table 2 compares the system-level inference efficiency of different ensemble methods on 1,000 randomly sampled ARC-Challenge requests at the same accuracy of 78.3%. All methods are evaluated through end-to-end generation on a single NVIDIA A800 80GB GPU, avoiding the effects of inter-GPU communication and parallel scheduling. We evaluate efficiency from four complementary aspects: latency, inference throughput, memory usage, and computational cost.

STAR achieves the best overall efficiency across most system metrics, with the lowest latency of 2.57 s/task, the highest throughput of 102.84 TPS, and the smallest computational cost of 1.86 TFLOPs/task. This efficiency mainly stems from STAR's selective ensemble strategy: SDP reuses sentence-boundary representations already produced during generation, while TARO is activated only at predicted disagreement states, concentrating additional multi-model computation on a small number of critical sentence boundaries. In contrast, FusionRoute and AdaFuse require more fre-

quent multi-model coordination during decoding, while CBS and SLM-MUX incur higher overhead due to repeated candidate generation and aggregation. In terms of memory usage, STAR requires 37.43 GB, only slightly higher than Avengers at 36.91 GB and lower than the other ensemble baselines. Although Avengers naturally has a smaller memory footprint because it selects a single model before generation, STAR maintains comparable memory usage while supporting dynamic sentence-level multi-LLM ensemble. Overall, these results show that STAR improves inference efficiency without introducing substantial additional computation or memory overhead.

To further quantify the runtime overhead introduced by STAR itself, Fig. 4 presents the runtime breakdown over 1,000 requests under different ensemble rates. LLM inference consistently dominates the end-to-end runtime, accounting for 96.91–97.51%, whereas SDP contributes only about 0.5%. As the ensemble rate increases from 10% to 20%, the runtime share of TARO rises modestly from 1.04% to 1.70%. Overall, SDP and TARO together account for only 1.59–2.19% of the total runtime, demonstrating that STAR introduces negligible additional runtime overhead even under more frequent ensemble.

Table 2: System-level efficiency comparison on 1,000 ARC-Challenge requests at 78.3% accuracy.

Method	Cost ($)	Latency (s/Task)	TPS	Peak Memory (GB)	TFLOPs / Task
Avengers	0.11	<u>3.71</u>	<u>70.84</u>	**36.91**	9.72
SLM-MUX	0.21	8.63	30.16	41.25	25.44
FusionRoute	0.08	4.28	60.72	38.61	5.18
AdaFuse	<u>0.05</u>	3.94	65.99	39.74	<u>4.36</u>
CBS	0.30	11.46	22.69	46.82	28.75
STAR	**0.04**	**2.57**	**102.84**	<u>37.43</u>	**1.86**

5.5 Further Analysis

We further analyze STAR from multiple perspectives, with details provided in Appendix C.2–C.7:

- **Parameter Analysis.** Increasing τ beyond 0.3 or B beyond 32 yields marginal accuracy gains at substantially higher cost or rollout budgets, supporting the default settings.
- **Generalization Analysis.** STAR achieves the highest average accuracy of 57.77% on unseen tasks with the lowest ensemble inference cost of $0.15.
- **Annotator Robustness.** Across three semantic annotators, STAR maintains 70.80%-71.74% average accuracy with similar inference cost of $0.39-$0.41.
- **Ensemble Behavior.** STAR invokes multi-LLM integration for only 20.2% of sentence-level decisions, with only a 2.7-point average gap from the true disagreement rate.
- **Trajectory-aware Routing.** STAR achieves 82.6% Oracle agreement versus 61.3% for local selection, validating trajectory-aware evaluation.
- **Model Contribution.** Even the smallest Gemma-2-2B-it contributes 21.0% of selected continuations, demonstrating complementary capabilities across heterogeneous LLMs.

6 Conclusion

We introduce *STAR*, a sentence-level trajectory-aware routing framework that transforms multi-LLM ensemble from uniform collaboration at every generation step into selective routing over sparse semantic disagreement states. STAR combines disagreement prediction with trajectory-aware candidate selection to reduce redundant computation while accounting for the long-term impact of routing decisions. Extensive experiments show that STAR achieves state-of-the-art accuracy among multi-LLM ensemble methods while reducing inference cost by up to 88.04%. Moreover, STAR surpasses a 70B standalone model while requiring 93.34% lower inference cost. These results highlight the potential of selective, trajectory-aware collaboration to unlock collective intelligence from heterogeneous LLMs beyond model scaling.

AI USE STATEMENT

In this work, we used generative AI tools to assist with language editing and presentation, including improving grammar, clarity, conciseness, and organization of the manuscript, as well as refining the presentation of figures and captions. Generative AI tools were not used to generate experimental results, fabricate data, or make final scientific conclusions. All AI-assisted content was carefully reviewed, revised, and verified by the authors to ensure technical correctness, originality, and consistency with the underlying experiments. The authors take full responsibility for all content presented in this work, including text and artifacts produced with the assistance of generative AI.

REFERENCES

Pranjal Aggarwal, Aman Madaan, Ankit Anand, Srividya Pranavi Potharaju, Swaroop Mishra, Pei Zhou, Aditya Gupta, Dheeraj Rajagopal, Karthik Kappaganthu, Yiming Yang, et al. Automix: Automatically mixing language models. In *Advances in Neural Information Processing Systems (NeurIPS 2024)*, pp. 131000–131034, 2024.

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

Mark Chen, Jerry Tworek, Heewoo Jun, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Chengming Cui, Tianxin Wei, Ziyi Chen, Ruizhong Qiu, Zhichen Zeng, Zhining Liu, Xuying Ning, Duo Zhou, and Jingrui He. Adafuse: Adaptive ensemble decoding for large language models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (ACL 2026)*, 2026.

Dujian Ding, Ankur Mallick, Shaokun Zhang, Chi Wang, Daniel Madrigal, Mirian Del Carmen Hipolito Garcia, Menglin Xia, Laks VS Lakshmanan, Qingyun Wu, and Victor Rühle. Best-route: Adaptive llm routing with test-time optimal compute. In *Forty-second International Conference on Machine Learning (ICML 2025)*, 2025.

Tao Feng, Yanzhen Shen, and Jiaxuan You. Graphrouter: A graph-based router for llm selections. In *International Conference on Learning Representations (ICLR 2025)*, volume 2025, pp. 26186–26203, 2025.

Tianyu Fu, Yi Ge, Yichen You, Enshu Liu, Zhihang Yuan, Guohao Dai, Shengen Yan, Huazhong Yang, and Yu Wang. R2r: Efficiently navigating divergent reasoning paths with small-large model token routing. In *Advances in Neural Information Processing Systems (NeurIPS 2025)*, pp. 124108–124145. Curran Associates, Inc., 2025.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning (ICML 2018)*, pp. 1861–1870, 2018.

Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, et al. Measuring coding challenge competence with apps. *arXiv preprint arXiv:2105.09938*, 2021a.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021b.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

Hongbeen Kim, Juhyun Lee, Sanghyeon Lee, Kwanghoon Choi, and Jaehyuk Huh. Adaptive parallel monte carlo tree search for efficient test-time compute scaling. *arXiv preprint arXiv:2604.00510*, 2026.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *The Eleventh International Conference on Learning Representations (ICLR 2023)*, 2023.

Dawei Li, Zhen Tan, Peijia Qian, Yifan Li, Kumar Chaudhary, Lijie Hu, and Jiayi Shen. Smoa: Improving multi-agent large language models with s parse m ixture-o f-a gents. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining (KDD 2025)*, pp. 54–65. Springer, 2025.

Siteng Liao, Yuzhu Liang, Hengzhong Rao, Xizhao Luo, and Tian Wang. Alpharouter: Token-level routing between slm and llm with reinforcement learning and tree search. In *Forty-third International Conference on Machine Learning (ICML 2026)*, 2026.

Cong Liu, Xiaojun Quan, Yan Pan, Weigang Wu, Xu Chen, and Liang Lin. Cool-fusion: Fuse large language models without training. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL 2025)*, pp. 10617–10627, 2025.

Hongwei Liu, Zilong Zheng, Yuxuan Qiao, Haodong Duan, Zhiwei Fei, Fengzhe Zhou, Wenwei Zhang, Songyang Zhang, Dahua Lin, and Kai Chen. Mathbench: Evaluating the theory and application proficiency of llms with a hierarchical mathematics benchmark. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 6884–6915, 2024.

Keming Lu, Hongyi Yuan, Runji Lin, Junyang Lin, Zheng Yuan, Chang Zhou, and Jingren Zhou. Routing to the expert: Efficient reward-guided ensemble of large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL 2024)*, pp. 1964–1974, 2024.

Mistral AI and NVIDIA. Mistral nemo. `https://mistral.ai/news/mistral-nemo/`, 2024.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E Gonzalez, Mohammed Kadous, and Ion Stoica. Routellm: Learning to route llms from preference data. In *International Conference on Learning Representations (ICLR 2025)*, volume 2025, pp. 34433–34448, 2025.

OpenRouter. Openrouter: A unified interface for large language models. `https://openrouter.ai/`, 2026. Accessed: Aug 2026.

Sungjin Park, Xiao Liu, Yeyun Gong, and Edward Choi. Ensembling large language models with process reward-guided tree search for better complex reasoning. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics (NAACL 2025)*, pp. 10256–10277, 2025.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Duncan Soiffer, Steven Kolawole, and Virginia Smith. Semantic agreement enables efficient open-ended llm cascades. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track (EMNLP 2025)*, pp. 2499–2537, 2025.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed H Chi, Denny Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 13003–13051, 2023.

Shengji Tang, Weihao Lin, Peng Ye, Jingqi Ye, Hao Li, Yiqun Zhang, Xiaosong Wang, Bo Zhang, Shuyue Hu, Tao Chen, et al. Beyond gemini-3-pro: Revisiting llm routing and aggregation at scale. *arXiv preprint arXiv:2601.01330*, 2026.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

Chenxu Wang, Hao Li, Yiqun Zhang, Linyao Chen, Jianhao Chen, Ping Jian, Qiaosheng Zhang, and Shuyue Hu. Icl-router: In-context learned model representations for llm routing. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI 2026)*, volume 40, pp. 33413–33421, 2026a.

Chenyu Wang, Zishen Wan, Hao Kang, Emma Chen, Zhiqiang Xie, Tushar Krishna, Vijay Janapa Reddi, and Yilun Du. Slm-mux: Orchestrating small language models for reasoning. In *International Conference on Learning Representations (ICLR 2026)*, volume 2026, pp. 73697–73723, 2026b.

Herbert Woisetschläger, Ryan Zhang, Shiqiang Wang, and Hans Arno Jacobsen. Mess+: Dynamically learned inference-time llm routing in model zoos with service level guarantees. In *Advances in Neural Information Processing Systems (NeurIPS 2026)*, volume 38, pp. 54034–54067, 2026.

Nuoya Xiong, Yuhang Zhou, Hanqing Zeng, Zhaorun Chen, Furong Huang, Shuchao Bi, Lizhu Zhang, and Zhuokai Zhao. Token-level llm collaboration via fusionroute. In *Forty-third International Conference on Machine Learning (ICML 2026)*, 2026.

Yangyifan Xu, Jinliang Lu, and Jiajun Zhang. Bridging the gap between different vocabularies for llm ensemble. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2024)*, pp. 7140–7152, 2024.

Yangyifan Xu, Jianghao Chen, Junhong Wu, and Jiajun Zhang. Hit the sweet spot! span-level ensemble for large language models. In *Proceedings of the 31st International Conference on Computational Linguistics (COLING 2025)*, pp. 8314–8325, 2025a.

Yangyifan Xu, Shuo Ren, and Jiajun Zhang. Collaborative beam search: Enhancing llm reasoning via collective consensus. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP 2025)*, pp. 11398–11410, 2025b.

Zitao Xuan, Xiaofeng Mao, Da Chen, Xin Zhang, Yuhan Dong, and Jun Zhou. Shieldhead: Decoding-time safeguard for large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 18129–18143, 2025.

Yuxuan Yao, Han Wu, Mingyang Liu, Sichun Luo, Xiongwei Han, Jie Liu, Zhijiang Guo, and Linqi Song. Determine-then-ensemble: Necessity of top-k union for large language model ensembling. In *International Conference on Learning Representations (ICLR 2025)*, volume 2025, pp. 101533–101551, 2025.

Yao-Ching Yu, Chun Chih Kuo, Ye Ziqi, Chang Yucheng, and Yueh-Se Li. Breaking the ceiling of the llm community by treating token generation as a classification for ensembling. In *Findings of the Association for Computational Linguistics (EMNLP 2024)*, pp. 1826–1839, 2024.

Heecheol Yun, Kwangmin Ki, Jung Hyun Lee, and Eunho Yang. When to ensemble: Identifying token-level points for stable and fast llm ensembling. In *International Conference on Learning Representations (ICLR 2026)*, volume 2026, pp. 148437–148460, 2026.

Yiqun Zhang, Hao Li, Chenxu Wang, Linyao Chen, Qiaosheng Zhang, Peng Ye, Shi Feng, Daling Wang, Zhen Wang, Xinrun Wang, Jia Xu, Lei Bai, Wanli Ouyang, and Shuyue Hu. The avengers: A simple recipe for uniting smaller language models to challenge proprietary giants. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI 2026)*, 2026.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody H Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, et al. Sglang: Efficient execution of structured language model programs. In *Advances in neural information processing systems (NeurIPS 2024)*, volume 37, pp. 62557–62583, 2024.

Zhenhong Zhou, Haiyang Yu, Xinghua Zhang, Rongwu Xu, Fei Huang, and Yongbin Li. How alignment and jailbreak work: Explain llm safety through intermediate hidden states. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 2461–2488, 2024.

A Algorithm Pseudocode

This section summarizes the complete training and inference procedures of STAR. Algorithm 1 presents the offline disagreement-label construction and SDP training, Algorithm 2 details the trajectory-aware rollout and routing-policy optimization in TARO, and Algorithm 3 describes the overall STAR inference procedure that integrates the trained SDP and routing policy.

Algorithm 1 Offline Label Construction and SDP Training

Require: Request set $\mathcal{D}_{\text{req}}$; candidate models $\mathcal{M} = \{M_k\}_{k=1}^{K}$ with base model $M_b \in \mathcal{M}$; Semantic annotator A
Require: agreement predictor f_θ
Ensure: Trained SDP f_θ
Initialize supervision set $\mathcal{D}_{\text{sup}} \leftarrow \emptyset$
▷ **Stage 1: Disagreement label construction**
for each request $x \in \mathcal{D}_{\text{req}}$ **do**
  Initialize shared context $h_1 \leftarrow x$
  while h_t is not terminal **do**
    Obtain the sentence-boundary representation $\mathbf{z}_t$ from M_b at context h_t
    Generate one candidate sentence from each model: $\mathcal{S}_t \leftarrow \{s_t^{(k)}\}_{k=1}^{K}$
    Obtain semantic cluster assignments: $\{c_t^{(k)}\}_{k=1}^{K} \leftarrow A(h_t, \mathcal{S}_t)$
    Construct disagreement label d_t using Eq. 2
    $\mathcal{D}_{\text{sup}} \leftarrow \mathcal{D}_{\text{sup}} \cup \{(\mathbf{z}_t, d_t)\}$
    Advance along the base-model trajectory: $h_{t+1} \leftarrow (h_t, s_t^{(b)})$
  end while
end for
▷ **Stage 2: Agreement predictor training**
for each training epoch **do**
  for each minibatch $\mathcal{B} \subset \mathcal{D}_{\text{sup}}$ **do**
    Compute $p^{\text{agr}} = f_\theta(\mathbf{z})$ for $(\mathbf{z}, d) \in \mathcal{B}$
    Compute the class-weighted loss $\mathcal{L}_{\text{agr}}$ using Eq. 3
    Update θ by minimizing $\mathcal{L}_{\text{agr}}$
  end for
end for
return f_θ

Algorithm 2 Tree-Advantage Routing Optimization

Require: Request set $\mathcal{D}_{\text{req}}$; candidate models $\mathcal{M} = \{M_k\}_{k=1}^{K}$ with base model $M_b \in \mathcal{M}$
Require: Semantic annotator A; routing policy π_θ; trajectory budget B
Ensure: Trained routing policy π_θ
Initialize rollout trees $\mathfrak{T} \leftarrow \emptyset$ and routing set $\mathcal{D}_{\text{route}} \leftarrow \emptyset$
▷ **Stage 1: Disagreement-triggered tree rollout**
for each request $x \in \mathcal{D}_{\text{req}}$ **do**
  Initialize trajectory tree $\mathcal{T}$ with root context $h_1 \leftarrow x$
  while $\mathcal{T}$ contains an expandable nonterminal leaf **do**
    Select a leaf v with context $h(v)$
    Generate candidates $\mathcal{S}(v) \leftarrow \{s^{(k)}\}_{k=1}^{K}$ from $\mathcal{M}$

Cluster candidates with A and retain one lowest-cost representative per cluster as $\widetilde{\mathcal{S}}(v)$
if $|\widetilde{\mathcal{S}}(v)| = 1$ **then**
Extend v with the base-model continuation $s^{(b)}$
else
$m \leftarrow |\widetilde{\mathcal{S}}(v)|$
if $|\mathcal{L}(\mathcal{T})| + m - 1 \leq B$ **then**
Mark v as a disagreement node
for each $s^{(k)} \in \widetilde{\mathcal{S}}(v)$ **do**
Add child u with $h(u) \leftarrow (h(v), s^{(k)})$ and $a(u) \leftarrow k$
end for
else
Mark v as non-expandable
end if
end if
end while
Complete all remaining nonterminal leaves with M_b
for each terminal leaf $\ell \in \mathcal{L}(\mathcal{T})$ **do**
$V(\ell) \leftarrow V(x, Y_\ell) - \lambda \sum_t C_t$
end for
$\mathfrak{T} \leftarrow \mathfrak{T} \cup \{\mathcal{T}\}$
end for
▷ **Stage 2: Max-value backup and counterfactual credit**
for each trajectory tree $\mathcal{T} \in \mathfrak{T}$ **do**
for each internal node v in reverse tree order **do**
$V(v) \leftarrow \max_{u \in \mathcal{C}(v)} V(u)$
end for
for each disagreement node v in $\mathcal{T}$ **do**
for each child $u \in \mathcal{C}(v)$ **do**
$A_{\mathrm{cf}}(u) \leftarrow V(u) - \max_{u' \in \mathcal{C}(v) \setminus \{u\}} V(u')$
Add $(h(v), \widetilde{\mathcal{S}}(v), a(u), A_{\mathrm{cf}}(u))$ to $\mathcal{D}_{\mathrm{route}}$
end for
end for
end for
▷ **Stage 3: Routing policy optimization**
Set $\theta_{\mathrm{old}} \leftarrow \theta$
for each optimization epoch **do**
for each minibatch $\mathcal{B} \subset \mathcal{D}_{\mathrm{route}}$ **do**
Update θ by maximizing $\mathcal{J}_{\mathrm{CAO}}$ in Eq. 9
Adjust α toward the target entropy
end for
end for
return π_θ

Algorithm 3 STAR Inference

Require: Query x; candidate models $\mathcal{M} = \{M_k\}_{k=1}^K$ with base model $M_b \in \mathcal{M}$
Require: Trained SDP f_ϕ; routing policy π_θ; agreement threshold τ
Ensure: Generated response Y
Initialize shared context $h_1 \leftarrow x$ and response $Y \leftarrow \emptyset$
while h_t is not terminal **do**
Extract the sentence-boundary representation $\mathbf{z}_t$ from M_b
$p_t^{\mathrm{agr}} \leftarrow f_\phi(\mathbf{z}_t)$
if $p_t^{\mathrm{agr}} \geq \tau$ **then**
Generate next sentence $s_t \leftarrow M_b(\cdot \mid h_t)$
else
Generate candidates $\mathcal{S}_t \leftarrow \{s_t^{(k)}\}_{k=1}^K$ from $\mathcal{M}$
$k_t \leftarrow \arg\max_k \pi_\theta(k \mid h_t, \mathcal{S}_t)$

$s_t \leftarrow s_t^{(k_t)}$
end if
$Y \leftarrow (Y, s_t), \quad h_{t+1} \leftarrow (h_t, s_t)$
end while
return Y

B Experimental Setup Details

B.1 Candidate Models and Task Datasets

Models. Table 3 summarizes the models used in our experiments. STAR operates over a heterogeneous candidate pool consisting of Gemma-2-2B-it, Mistral-7B-Instruct-v0.3, and Llama-3.1-8B-Instruct. We use Gemma-2-2B-it, the smallest and lowest-cost model, as the base model M_b for default sentence generation, while all three models participate when ensemble is triggered. For comparison, Mistral-NeMo-12B-Instruct and Llama-3.1-70B-Instruct are evaluated independently as larger single-model references and are not included in the candidate LLM pool.

Table 3: Models used in our experiments.

Model	Params.	Role
Gemma-2-2B-it (Team et al., 2024)	2B	Base / Candidate
Mistral-7B-Instruct-v0.3 (Jiang et al., 2023)	7B	Candidate
Llama-3.1-8B-Instruct (Grattafiori et al., 2024)	8B	Candidate
Mistral-NeMo-12B-Instruct (Mistral AI & NVIDIA, 2024)	12B	Single-model Baseline
Llama-3.1-70B-Instruct (Grattafiori et al., 2024)	70B	Single-model Baseline

Task Datasets. Table 4 summarizes the datasets used in our experiments. We evaluate on five in-distribution benchmarks: GSM8K, MATH500, MBPP, HumanEval, and ARC-Challenge, comprising 8,000 examples in total. To assess out-of-distribution generalization, we further consider MathBench, APPS, and BIG-Bench Hard (BBH), randomly sampling 500 examples from each benchmark. These OOD datasets are used only for evaluation and are excluded from training, validation, and hyperparameter selection.

Table 4: Detailed information of the task datasets.

Dataset	Category	#Examples
In-distribution Datasets		
GSM8K (Cobbe et al., 2021)	Math	3,772
MATH500 (Hendrycks et al., 2021b)	Math	500
MBPP (Austin et al., 2021)	Code	974
HumanEval (Chen et al., 2021)	Code	164
ARC-Challenge (Clark et al., 2018)	Reasoning	2,590
Total	–	**8,000**
Out-of-distribution Datasets		
MathBench (Liu et al., 2024)	Math	500
APPS (Hendrycks et al., 2021a)	Code	500
BBH (Suzgun et al., 2023)	Reasoning	500
Total	–	**1,500**

B.2 Baselines

Avengers. As an ensemble-before-inference method, Avengers (Zhang et al., 2026) performs query-level routing based on input semantics. We implement it using the same candidate pool as STAR. Queries are encoded with `gte-Qwen2-7B-instruct` and grouped into 32 clusters using K-Means. Candidate models are ranked by their calibration performance within each cluster. During

inference, each query is routed to the top-ranked model, which generates 3 samples, and the final answer is selected by majority voting.

SLM-MUX. Designed for ensemble-after-inference, SLM-MUX (Wang et al., 2026b) selects outputs by aggregating independently generated responses from multiple LLMs. We implement it using the same three candidate models as STAR. Each model independently generates 3 responses per query with a temperature of 0.3. For math and reasoning tasks, model confidence is estimated by self-consistency through majority voting. For code generation, we follow the embedding-based consistency estimator and encode generated solutions using `Salesforce/codet5p-110m-embedding` with cosine similarity. The output from the model with the highest consistency score is selected as the final answer.

FusionRoute. As a state-of-the-art token-level routing method, FusionRoute (Xiong et al., 2026) dynamically selects experts and complements their predictions during decoding. Since they use different tokenizers, we follow EVA (Xu et al., 2024) to align their output distributions in a shared vocabulary space, with Gemma-2-2B-it as the pivot model. The router is initialized from Gemma-2-2B-it, while all experts remain frozen. At each decoding step, it selects the highest-scoring expert and combines its aligned output distribution with complementary router logits for next-token prediction. Apart from vocabulary alignment, we follow the original training procedure with supervised fine-tuning (SFT) and Complemented Direct Preference Optimization (CDPO).

AdaFuse. We compare against AdaFuse (Cui et al., 2026), a state-of-the-art adaptive ensemble decoding method that dynamically coordinates multiple LLMs during generation. Following the original implementation, each model greedily generates word-level continuations and adaptively extends them according to the confidence margin between the top-1 and top-2 probabilities of the first token. We set the confidence threshold to $\tau_\Delta = 0.7$ and allow at most $M = 3$ consecutively committed words per decoding round. Candidate spans from all models are evaluated using normalized negative log-likelihood (NLL), averaged across the three models, and the span with the lowest score is appended to the shared prefix. Each model uses its native tokenizer for generation and scoring.

CBS. We further include CBS (Xu et al., 2025b), a representative process-level collaborative search method that coordinates multiple LLMs through collective consensus during reasoning. At each reasoning step, the models generate multiple candidate continuations, which are scored by the average perplexity across all models, and the top-B trajectories are retained for subsequent expansion. Following the original implementation, we set the beam size to $B = 4$, the total sampling budget per round to $K = 32$, and the quota-allocation temperature to $\tau = 0.1$. Candidate continuations are generated with nucleus sampling using temperature 0.6 and top-$p = 0.9$. The sampling quota of each model is dynamically adjusted according to its contribution to the retained beams.

B.3 Evaluation Metrics

Task Performance. For mathematical reasoning benchmarks, including GSM8K, MATH500, and MathBench, we report exact-match accuracy based on the normalized final answer. For ARC-Challenge (ARC-C) and BIG-Bench Hard (BBH), we report accuracy by comparing the predicted answer with the corresponding ground truth. For code-generation benchmarks, including MBPP, HumanEval, and APPS, we report pass@1 using the official test cases, where a solution is considered correct only if it passes all associated tests. For the in-distribution evaluation, the average accuracy is the unweighted mean across the five ID benchmarks; for OOD evaluation, it is the unweighted mean across MathBench, APPS, and BBH.

Inference Cost. We estimate monetary inference cost according to the input and output token prices on OpenRouter (OpenRouter, 2026). Table 5 summarizes the token prices used for all evaluated models. For model M_i, the cost is computed as

$$C_i = \frac{N_i^{\text{in}}}{10^6} P_i^{\text{in}} + \frac{N_i^{\text{out}}}{10^6} P_i^{\text{out}}, \tag{10}$$

where N_i^{in} and N_i^{out} denote the numbers of input and output tokens, and P_i^{in} and P_i^{out} are their prices per million tokens. Unless otherwise specified, the reported Cost denotes the total inference cost over all evaluated requests.

System-Level Efficiency. For the system-level comparison in Section 5.4, we report the following metrics:

- **Cost ($):** total monetary inference cost over the 1,000 requests, computed using the pricing protocol above.
- **Latency (s/Task):** average end-to-end wall-clock inference time per request, including model generation and method-specific routing or ensemble overhead.
- **TPS:** inference throughput measured as the number of generated tokens per second over the complete evaluation.
- **Peak Memory (GB):** average per-request peak GPU memory consumption during inference.
- **TFLOPs/Task:** average number of floating-point operations required per request, expressed in 10^{12} FLOPs.

Table 5: Token prices used for inference-cost estimation.

Model	Input ($/M tokens)	Output ($/M tokens)
Gemma-2-2B-it	0.01	0.01
Mistral-7B-Instruct-v0.3	0.02	0.02
Llama-3.1-8B-Instruct	0.02	0.04
Mistral-NeMo-12B-Instruct	0.06	0.06
Llama-3.1-70B-Instruct	0.40	0.40

B.4 Implementation Details

B.4.1 Model Architecture

We implement our framework with two lightweight modules: **SDP** and **TARO**. The **SDP** takes the last-layer hidden state of the base model M_b at each sentence boundary as input, where M_b denotes the lowest-cost candidate model (Gemma-2-2B-it in our experiments). The predictor adopts a two-layer MLP with layer normalization, a linear projection to a 512-dimensional hidden space, GELU activation, and a sigmoid output head. The base model remains frozen during training.

The **TARO** routing policy adopts a candidate scoring network. The sentence-boundary hidden state of M_b serves as the context representation, while the final hidden states of candidate models serve as candidate representations. Representations from different LLMs are independently projected into a shared 512-dimensional space, concatenated, and processed by a two-layer MLP with GELU activation, followed by softmax normalization to obtain π_θ. All candidate LLMs remain frozen during training, and only the routing network parameters are optimized.

B.4.2 Training Hyperparameters

We train the disagreement predictor and routing policy in two separate stages. Both modules share the same offline candidate generation and semantic annotation pipeline but use different supervision signals: binary disagreement labels for **SDP** and trajectory-level advantages for **TARO**. The two modules are optimized independently.

For **SDP**, sentence-level disagreement labels are constructed using an offline LLM-as-a-judge procedure with GPT-5.5 as the semantic annotator. Given a shared prefix h_t, candidate continuations $\mathcal{S}_t$ are generated by all K LLMs with greedy decoding ($\mathrm{temperature} = 0$) and a maximum length of 64 tokens. The annotator produces binary disagreement labels based on semantic equivalence among candidates. The prompt template is provided in Appendix B.5. The predictor is trained on 20,000 sentence-level samples with a train/validation/test split of 0.6/0.1/0.3. We use the AdamW optimizer (Adam with decoupled weight decay) with a learning rate of 1×10^{-5}, weight decay of 0.01, batch size of 256, and 10 training epochs. The weighted binary cross-entropy loss uses class weights $w_0 = 1.0$ and $w_1 = 2.3$. The ensemble threshold is set to $\tau = 0.3$ based on the validation set.

For **TARO**, trajectory trees are constructed from offline-identified disagreement states. Candidate generation follows the same decoding configuration as **SDP**, and each rollout explores at most $B = 32$ trajectories. The terminal reward combines task correctness and normalized inference cost with

$\lambda = 0.1$. The routing policy is optimized using AdamW with a learning rate of 1×10^{-5}, weight decay of 0.01, and batch size of 128. PPO optimization uses $\epsilon = 0.2$, four policy updates per batch, normalized counterfactual advantages, and gradient clipping with a maximum norm of 1.0. The entropy coefficient is initialized as $\alpha = 0.01$ with a target entropy of $\mathcal{H}_{\text{target}} = 0.15$.

B.4.3 Hardware and Software Environment

Unless otherwise specified, all experiments are conducted on 8 $\times$ NVIDIA V100 (32GB) GPUs. The system efficiency experiments in Section 5.4 are conducted separately on a single NVIDIA A800 (80GB) GPU to evaluate end-to-end inference efficiency under a unified single-GPU setting. The ensemble inference framework is implemented using PyTorch and SGLang Zheng et al. (2024) for efficient LLM inference and token generation.

B.5 Disagreement Annotation Prompt

We use the following prompt to instruct the LLM annotator to group candidate sentence-level continuations according to their reasoning semantics. The annotator is explicitly instructed to assess semantic equivalence rather than correctness or candidate quality.

Prompt for Disagreement Label Construction

```
You are an expert annotator for sentence-level multi-LLM reasoning.

Given a shared reasoning prefix and several candidate next sentences
generated independently by different language models, determine
    whether
the candidates represent the same next reasoning step.

Two candidates should be considered equivalent if:
1. they express the same key fact, inference, operation, or
    conclusion;
2. differences are limited to wording, style, or non-essential
    details;
3. substituting one candidate for another would not materially
    change
  the subsequent reasoning direction.

Candidates should NOT be considered equivalent if:
1. they make conflicting factual or numerical claims;
2. they perform materially different reasoning operations;
3. they pursue different solution directions;
4. they are merely non-contradictory but advance different reasoning
     steps.

Do NOT evaluate which candidate is better.
Do NOT evaluate whether the candidates are correct.
Judge only their reasoning-semantic equivalence under the given
    prefix.

Shared prefix:
<prefix>
{prefix}
</prefix>

Candidates:
A: {sentence_A}
B: {sentence_B}
C: {sentence_C}

Group the candidates into semantic reasoning clusters.
```

```
Return only:
{
  "clusters": [["A", "B"], ["C"]],
  "consistent": false
}
```

C Experiment Results

C.1 Detailed Ablation Results

Table 6 reports the complete results on all five benchmarks. The overall trend is consistent with the main-text analysis, but the sensitivity varies across tasks. MATH500 is particularly sensitive to the ablations: its accuracy drops from 56.69% with full STAR to 41.35% without SDP, 44.78% without counterfactual advantage, and 48.23% without tree rollout. This is consistent with the stronger long-range dependency of mathematical reasoning, where errors at intermediate routing decisions can propagate through subsequent steps. In contrast, ARC-C shows relatively smaller degradation, remaining between 77.52% and 78.41% for most ablated variants compared with 83.26% for STAR, suggesting that its shorter reasoning trajectories depend less on long-horizon trajectory optimization. The code benchmarks exhibit an intermediate pattern: MBPP drops from 68.51% to 55.92–64.70%, while HumanEval decreases from 69.80% to 58.67–66.12%. Overall, these results indicate that STAR provides the largest gains on tasks requiring longer and more tightly coupled reasoning trajectories.

Table 6: Component-wise ablation results of STAR across five benchmarks. We report task-specific accuracy, average accuracy, and total inference cost.

Method	GSM8K	MATH500	MBPP	HumanEval	ARC-C	Avg. Acc. (%)	Cost ($)
w/o SDP	67.84	41.35	55.92	58.67	77.52	60.26	1.82
w/o Tree Rollout	72.96	48.23	61.37	63.54	77.95	64.81	0.41
w/ Mean Backup	75.48	51.62	64.70	66.12	77.73	67.13	0.40
w/o Counterfactual Advantage	70.21	44.78	58.63	60.92	78.41	62.59	0.42
STAR	**80.42**	**56.69**	**68.51**	**69.80**	**83.26**	**71.74**	**0.39**

C.2 Parameter Analysis

Figure 5 studies the effects of the SDP threshold τ and trajectory budget B. Increasing τ from 0.1 to 0.3 improves accuracy from 63.29% to 71.74% with only a moderate cost increase from \$0.26 to \$0.39. Beyond $\tau = 0.3$, the accuracy gain becomes marginal, while the cost rises rapidly to \$0.68 at $\tau = 0.5$. For B, Avg. Explored Trajectories denotes the average number of rollout trajectories actually explored per request under the given budget. Increasing B from 4 to 32 substantially improves accuracy from 57.19% to 71.74%, whereas increasing it further to 64 yields only a 1.41-point gain while raising the average explored trajectories from 25.6 to 48.3. These results reveal clear diminishing returns for both parameters and support $\tau = 0.3$ and $B = 32$ as the default settings.

C.3 Generalization Analysis

Table 7 evaluates whether STAR transfers its ensemble policy to unseen task distributions. STAR achieves the best overall accuracy (57.77%) with the lowest ensemble inference cost (\$0.15), showing that its advantage is not limited to in-distribution data. The gains are most pronounced on MathBench and APPS, where STAR outperforms the second-best methods by 5.20 and 4.90 points, respectively. This is consistent with the trajectory-aware design of STAR: mathematical reasoning and code generation involve long-range dependencies, making local confidence, consensus, or token-level decisions less reliable for identifying continuations that lead to correct final solutions. By contrast, query-level routing methods such as Avengers make a fixed model choice before generation and cannot adapt to uncertainty that emerges during reasoning, which becomes more prob-

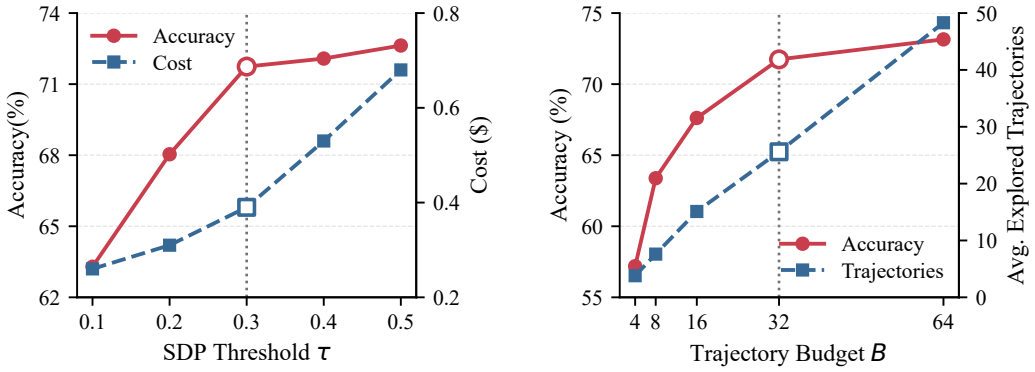

Figure 5: Parameter sensitivity of STAR to the SDP threshold τ and trajectory budget B.

lematic under distribution shift. FusionRoute also shows weaker transfer on MathBench and APPS, suggesting that token-level signals may not sufficiently capture long-horizon reasoning quality.

Interestingly, AdaFuse slightly outperforms STAR on BBH (72.40% vs. 71.60%), indicating that local confidence-based ensemble can be effective for shorter general-reasoning tasks, where trajectory optimization provides less additional benefit. Overall, the results suggest that sentence-level disagreement and downstream trajectory value provide transferable ensemble signals, enabling STAR to maintain a favorable accuracy-cost trade-off under distribution shift.

Table 7: Generalization performance and inference cost comparison across three out-of-distribution benchmarks. Best results are shown in **bold**, and second-best results are underlined.

Method	MathBench	APPS	BBH	Avg. Acc. (%)	Cost ($)
Candidate LLMs					
Gemma-2-2B-it	24.90	9.80	41.70	25.47	0.04
Mistral-7B-Instruct-v0.3	38.70	18.60	54.90	37.40	0.09
LLaMA-3.1-8B-Instruct	47.60	27.90	65.10	46.87	0.18
LLM Ensemble Baselines					
Avengers	49.10	29.10	65.80	48.00	0.46
SLM-MUX	51.60	32.40	67.20	50.40	0.88
FusionRoute	47.90	26.80	69.50	48.07	0.32
AdaFuse	54.80	35.90	**72.40**	54.37	0.29
CBS	55.70	35.20	68.90	53.27	1.21
STAR	**60.90**	**40.80**	71.60	**57.77**	**0.15**

C.4 Robustness to Semantic Annotators

As shown in Table 8, we examine whether the effectiveness of SDP depends on the choice of semantic annotator. We replace GPT-5.5 with two strong LLM annotators (*Claude Opus 5* and *Gemini 3.1 Pro*) to independently construct disagreement labels while keeping all other settings unchanged. STAR maintains stable overall performance across different annotators, achieving 70.80%-71.74% average accuracy with inference costs of $0.39-$0.41. Despite variations on individual benchmarks, the overall performance remains comparable, indicating limited sensitivity to the specific LLM used for semantic annotation.

C.5 Ensemble Behavior Analysis

Figure 6 examines how frequently STAR activates multi-LLM ensemble and whether these activations align with actual model disagreement. Overall, ensemble is invoked for only 20.2% of

Table 8: Robustness of STAR to different semantic annotators.

Annotator	GSM8K	MATH500	MBPP	HumanEval	ARC-C	Avg. Acc. (%)	Cost ($)
Claude Opus 5	78.06	55.72	65.24	**70.81**	84.17	70.80	0.41
Gemini 3.1 Pro	79.14	**56.84**	67.92	67.28	**86.47**	71.53	**0.39**
GPT-5.5	**80.42**	56.69	**68.51**	69.80	83.26	**71.74**	**0.39**

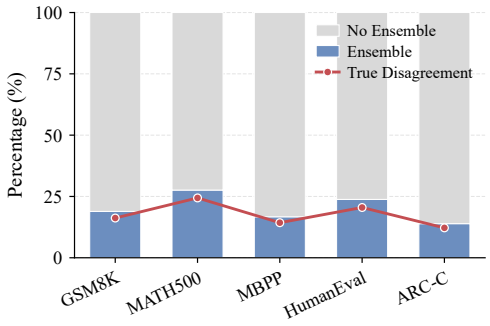

Figure 6: Ensemble behavior of STAR across five benchmarks. Stacked bars show collaborative and non-collaborative decisions, while the line shows the true disagreement rate.

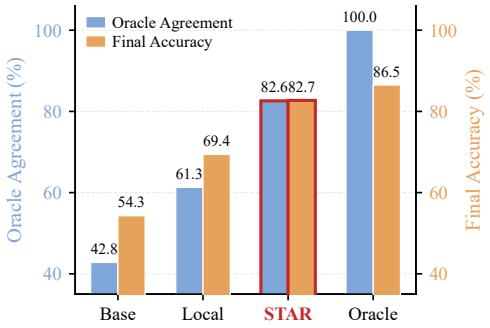

Figure 7: Trajectory-aware routing analysis on 1,000 ARC-C tasks, comparing Oracle agreement and final accuracy across different routing strategies.

sentence-level decisions on average, confirming that most generation states can be handled by the lowest-cost model without redundant multi-model computation. More importantly, the ensemble rate closely tracks the true disagreement rate across all benchmarks, with only a small average gap of 2.7 percentage points. This indicates that SDP concentrates additional computation around genuinely ambiguous states while maintaining a modest safety margin to avoid missing potential disagreements.

The triggering behavior also varies systematically with task characteristics. MATH500 exhibits the highest ensemble and true disagreement rates (27.6% vs. 24.4%), suggesting that complex mathematical reasoning creates more divergent intermediate trajectories and therefore benefits more from multi-model intervention. HumanEval shows a similar tendency (23.8% vs. 20.5%), where alternative implementation choices can lead candidate models toward different continuations. In contrast, ARC-C requires ensemble on only 13.9% of states, consistent with its lower true disagreement rate of 12.2%. These results suggest that STAR does not apply a fixed ensemble pattern across tasks, but adaptively allocates multi-model computation according to the intrinsic uncertainty of the reasoning trajectory.

C.6 Trajectory-aware Routing Analysis

Figure 7 compares four routing strategies on 1,000 randomly sampled ARC-C tasks. *Base* always uses the lowest-cost model to generate the next sentence; *Local Selection* chooses the candidate with the lowest sentence-level perplexity; *STAR* applies our trajectory-aware routing strategy; and *Oracle* selects the candidate that yields the highest final task reward after complete rollout, serving as an empirical upper bound. Base achieves only 42.8% Oracle agreement and 54.3% final accuracy, while Local Selection improves them to 61.3% and 69.4%, respectively. This shows that comparing candidate continuations is beneficial, but also reveals the limitation of purely local criteria: a sentence with lower perplexity does not necessarily lead to a better downstream reasoning trajectory.

STAR substantially increases Oracle agreement to 82.6% and achieves 82.7% final accuracy, approaching the Oracle accuracy of 86.5%. This indicates that trajectory-level evaluation provides a more reliable routing signal than local generation quality and better captures the long-term effect of each intermediate decision. Moreover, the smaller gap in final accuracy than in Oracle agree-

ment suggests that some non-Oracle intermediate choices can still converge to correct solutions. Overall, these results validate the key motivation of STAR: effective sentence-level routing requires reasoning about long-term trajectory outcomes rather than making purely local decisions.

C.7 Model Contribution Analysis

Figure 8 examines which candidate model contributes the selected continuation once ensemble is triggered. The results reveal clear task-dependent specialization rather than a uniform preference for the largest model. On mathematical reasoning tasks, LLaMA-3.1-8B-Instruct is selected most frequently, accounting for 49.5% on GSM8K and 56.8% on MATH500, suggesting that its stronger reasoning capability is particularly useful at difficult disagreement states. In contrast, Mistral-7B-Instruct-v0.3 contributes more on code generation, reaching 43.7% on MBPP and 46.9% on HumanEval, while the three models are selected more evenly on ARC-C (31.6%–35.6%), indicating that no single model consistently dominates general reasoning. Gemma-2-2B-it is still selected in 21.0% of ensemble-triggered states on average, despite being the smallest and lowest-cost candidate. This suggests that weaker models can still contain superior reasoning trajectories at particular intermediate states, and that model quality cannot be determined solely by global capability or parameter scale. Even when ensemble is necessary, always switching to a stronger model may therefore discard useful branches from the weaker model's search space. Overall, these results show that STAR exploits complementary reasoning paths across models rather than simply favoring larger models.

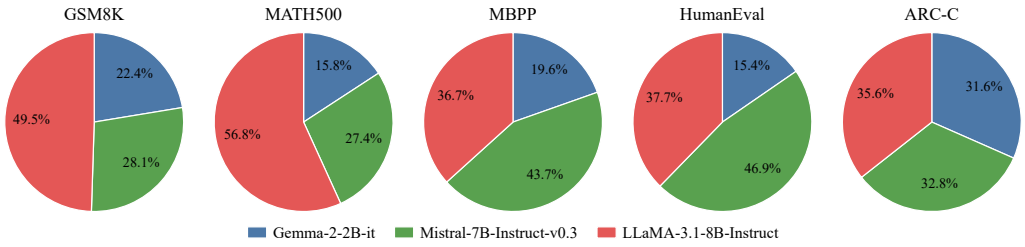

Figure 8: Model selection ratios at ensemble-triggered sentence boundaries across five benchmarks.