FIDELITYFLOW-PO: BANDIT-GUIDED ALLOCATION OF CONTEXT FIDELITY FOR LONG-CONTEXT DPO

Anonymous authors
Paper under double-blind review

ABSTRACT

For long-context Direct Preference Optimization (DPO), the quality of preference pairs depends critically on the evidence available when candidate responses are generated. One adaptive strategy is to use response feedback to revise the context shown across candidate-generation rounds. A recent method, LongMab, instantiates this strategy by partitioning the long context into chunks and using a multi-armed bandit to adapt which chunks are shown during candidate-response generation. However, because decisions are made at the whole-chunk level, each chunk must either be included in full, potentially introducing irrelevant content, or be omitted entirely, potentially discarding critical bridging evidence. This couples source coverage with reading cost. We introduce FidelityFlow-PO, which advances LongMab's chunk-as-arm formulation from binary source selection to feedback-driven control of reading cost. The method preserves a compact, query-conditioned core from each chunk as a starting point for deeper reading, while response feedback adaptively controls how much surrounding context is included. This decouples source coverage from reading depth: access to a source no longer requires full-chunk inclusion, allowing broad coverage to coexist with selective, deeper reading during preference construction. Experiments across the Qwen and Llama model families show that FidelityFlow-PO improves both candidate-response quality and downstream long-context question answering. Across five benchmarks, it increases macro-F1 over the LongMab baselines on Qwen2.5-7B and Llama-3.1-8B. All adaptive reading remains offline, requiring no additional inference-time search.

1 INTRODUCTION

A long context window does not guarantee that a language model can reliably use the information it contains (Liu et al., 2024). Even when an input fits within the supported window, relevant evidence may be overlooked and irrelevant material can distract response generation (Shi et al., 2023). This limitation also affects long-context Direct Preference Optimization (DPO) (Rafailov et al., 2023). The central challenge is to organize evidence so that the generator can use it more effectively, rather than simply accommodate more text. LongMab addresses this challenge through adaptive exploration of context combinations (Duan et al., 2026). It treats context chunks as arms in a multi-armed bandit (MAB), balancing exploration and exploitation. This process searches for informative combinations, with the aim of producing high-quality and diverse preference candidates. However, its action remains binary: each selected chunk is included in full. This granularity leaves a trade-off between retaining useful evidence and introducing unnecessary context. Omitting the first can break the evidence chain; including it also exposes the generator to surrounding material that may be irrelevant or distracting. Whole-chunk actions thus couple source access to reading depth, restricting which evidence combinations can be presented without also introducing unnecessary text (Figure 1).

We introduce **FidelityFlow-PO**, which advances LongMab's chunk-as-arm formulation from binary source selection to feedback-driven control of reading depth. Its guiding principle is *coverage–depth decoupling*: maintain access to a source without committing to its full contents. Specifically, the method preserves a compact, query-conditioned core in each context chunk as a persistent starting

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

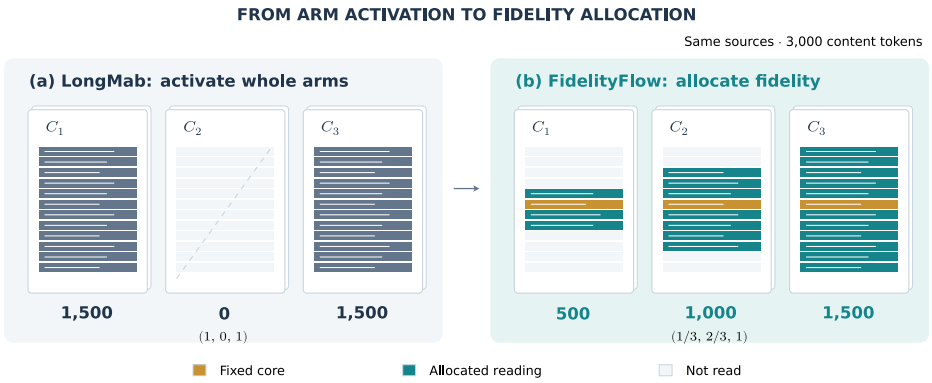

Figure 1: **Flexible Reading under the Same Budget.** Both panels read 3,000 content tokens from the same three 1,500-token chunks. (a) Whole-chunk selection admits at most two complete chunks and leaves the third unread. (b) FidelityFlow-PO keeps a fixed core in every chunk and allocates the remaining tokens unevenly, so all three sources stay accessible while reading depth varies across chunks. Numbers give tokens read per chunk; tuples give the fraction of each chunk read.

point for deeper reading. At each round, the controller allocates additional reading across chunks within a shared budget, using their current priorities augmented by exploration bonuses that favor less-exposed chunks. The generator then produces a response from the resulting input, and the response is scored against the reference answer. The controller subtracts the question's historical reward baseline from the observed reward to obtain an advantage. It distributes this advantage across chunks in proportion to their realized shares of additional reading and uses the assigned feedback to update their priorities for the next round. Response feedback thus controls how deeply each chunk is read rather than which chunks are included.

FidelityFlow-PO generates candidate responses from views that combine persistent source access with selective detail, then forms preferred–dispreferred response pairs. We subsequently fine-tune the language model with DPO using these pairs and training inputs reconstructed from the original long contexts. This allows the model to learn from responses elicited under more focused evidence conditions while being trained to answer from long-context inputs. The resulting model processes long-context questions directly, without an adaptive reading controller or iterative context search at inference. Experiments across the Qwen and Llama model families show that FidelityFlow-PO improves candidate quality and achieves higher five-benchmark macro-F1 than LongMab baselines under matched training identities and protocols.

2 Related Work

Long-context instruction and preference learning. Supporting longer inputs does not ensure that a language model can reliably use the evidence they contain (Liu et al., 2024; 2025). Instruction tuning and preference optimization seek to improve this capability through different forms of response supervision. LongAlpaca, released with LongLoRA (Chen et al., 2024), and LongAlign (Bai et al., 2024a) provide long-context instruction examples for supervised fine-tuning. LongMab-SFT likewise learns from individual responses, using only the chosen outputs in LongMab's preference data (Duan et al., 2026). Preference optimization offers a complementary signal based on relative response quality, with DPO learning directly from preferred–dispreferred response pairs (Rafailov et al., 2023). The quality of this supervision depends on the candidates available for comparison and, in long-context settings, on the evidence presented when those candidates are generated.

Constructing long-context preference data. Preference construction involves two complementary decisions: how to elicit candidate responses and how to assess their relative quality. LongReward uses LLM-generated feedback to evaluate responses (Zhang et al., 2025), while SeaLong estimates candidate quality through semantic consistency across sampled outputs (Li et al., 2024). Complementing response assessment, LongFaith-PO guides preference synthesis with ground-truth

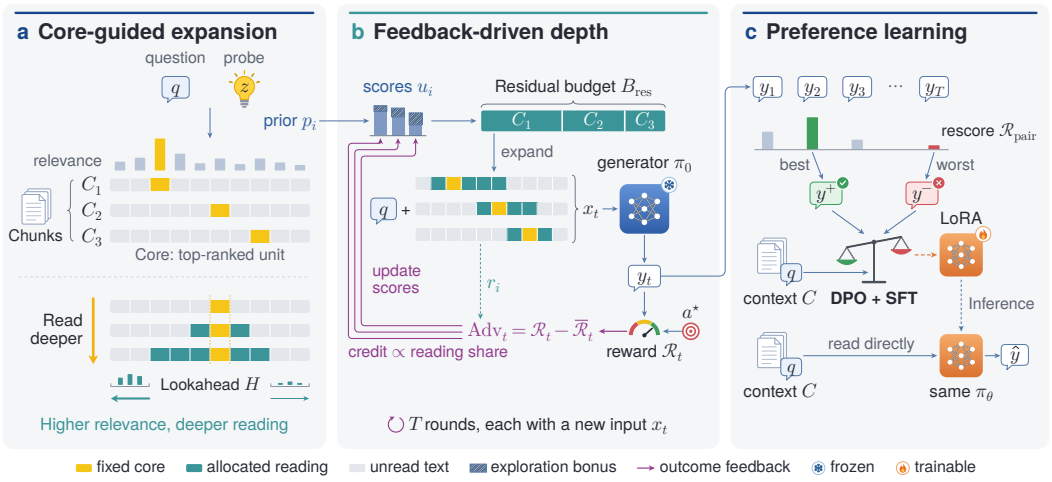

Figure 2: FidelityFlow-PO decouples source access from reading depth. (a) Each chunk keeps a fixed core around its unit most relevant to q and z, so every source stays accessible. Nested views grow outward toward higher relevance density; relevance thus fixes *where* each chunk is read. (b) Allocation scores u_i, combining the prior p_i, outcome feedback, and an exploration bonus, split the residual budget B_{res}, setting *how much* of each chunk is read. The frozen generator π_0 answers from x_t; the answer's advantage Adv_t over a historical baseline is credited to chunks in proportion to their realized reading shares, updating u_i. (c) After T rounds, the best and worst candidates under $\mathcal{R}_{\text{pair}}$ form a preference pair (y^+, y^-) for LoRA training with DPO plus SFT on y^+, using the original context C as input. At inference, the trained model π_θ reads C directly.

answers and citation-based reasoning (Yang et al., 2025). LOGO intervenes in the context itself, varying the inclusion of relevant and irrelevant chunks to elicit contrasting responses (Tang et al., 2025). Selective text retention, as studied in prompt compression (Jiang et al., 2024; Pan et al., 2024), provides a related means of shaping model inputs. Our focus is using response outcomes to adapt the evidence shown in subsequent generation rounds.

Bandit-guided evidence construction. LongMab (Duan et al., 2026) brings response feedback into context construction by treating chunks as arms of a multi-armed bandit. A reasoning probe initializes chunk values, upper confidence bound (UCB) selection explores complementary chunk combinations; the resulting candidates then provide preference pairs for DPO on full-context inputs. In this formulation, however, access to a source is tied to reading it in full: including a useful chunk also brings in all of its surrounding text, whereas excluding the chunk to avoid that text also discards its evidence. FidelityFlow-PO instead decouples source access from reading depth. It anchors persistent access in compact, query-conditioned cores and adapts additional reading to answer outcomes, so exploration can redistribute contextual detail without removing any source's entry point, letting broad coverage coexist with selective, deeper reading.

3 Method

Given a training question q, context C, and reference answer $a^\star$, LongMab treats the ordered chunks $\{C_i\}_{i=1}^n$ as bandit arms, initializing their values through similarity to a filtered, answer-conditioned reasoning probe z. At each round, an upper confidence bound (UCB) rule selects K complete chunks, and the resulting response reward updates the selected arms' value estimates. This feedback adapts source combinations, but each selected source still enters at full depth. FidelityFlow-PO builds on this formulation by decoupling source access from reading depth: every chunk keeps a core as its entry point, while reading depth becomes a feedback-controlled decision. Scoring and pairing follow LongMab. Figure 2 and Algorithm 1 summarize the procedure. Theoretical analysis is provided in Appendix B.

Algorithm 1 FidelityFlow-PO

Require: Training item $(q, C, a^\star)$, chunks $\{C_i\}$, probe z, frozen generator π_0, rounds T.
Ensure: A preference sample (x, y^+, y^-), when available.
1: Prepare fixed cores, nested views, priors, and reading allowance.
2: Initialize outcome corrections $\Delta_i = 0$ and exposures $N_i = 0$.
3: **for** $t = 1, \ldots, T$ **do**
4: Allocate additional reading using Eqs. (2) and (5).
5: Construct x_t; record actual additional reading $r_i^{(t)}$.
6: $y_t \sim \pi_0(\cdot \mid x_t)$; $\mathcal{R}_t \leftarrow \mathcal{R}_{\text{roll}}(y_t, a^\star)$.
7: **if** $t = 1$ **then**
8: $\overline{\mathcal{R}} \leftarrow \mathcal{R}_t$.
9: **end if**
10: Compute Adv_t, $\omega_i^{(t)}$, and $e_i^{(t)}$ using Eq. (3).
11: Update Δ_i, N_i, and $\overline{\mathcal{R}}$ using Eq. (4).
12: **end for**
13: Select (y^+, y^-) using Eq. (8).
14: Reconstruct x from q and C; form a pair when scores differ.
15: **return** the preference sample (x, y^+, y^-).

3.1 Establishing persistent cores

Each chunk is further split into sentence-based units s_{ij}, the j-th unit of chunk i. Establishing a core first requires measuring how relevant each unit is to the answering goal. The question q, however, specifies what to answer but not necessarily where the required text lies. We therefore also score units against the auxiliary reasoning probe z, which is generated conditioned on the training question and reference answer and whose reasoning cues help locate useful source text. Given a frozen encoder E, unit relevance combines question and probe relevance with equal weights:

$$g_{ij} = \tfrac{1}{2}\psi(q, s_{ij}) + \tfrac{1}{2}\psi(z, s_{ij}), \qquad \psi(x, y) = \frac{\cos\big(E(x), E(y)\big) + 1}{2}, \tag{1}$$

where ψ maps cosine similarity to $[0, 1]$.

Based on Eq. (1), the highest-scoring unit in each chunk serves as an anchor, and a short original-text span around it forms the core C_i^{core}. The core stays fixed throughout candidate generation, so every chunk always retains its highest-scoring entry point in the input. Response feedback only adjusts how much surrounding text is read. The remainder of each chunk defines its capacity for additional reading, $R_i = L_i - A_i$, where L_i and A_i denote the token lengths of the chunk and its core. Because cores and expansion paths are both fixed in advance (Section 3.3), each subsequent round only needs to decide how much additional text to read from each chunk within this capacity.

3.2 Feedback-driven reading-depth control

After all cores are reserved, let B_{res} denote the remaining per-question allowance for additional reading. Candidate generation runs for T rounds, and $r_i^{(t)} \in [0, R_i]$ denotes the actual additional reading of chunk i in round t. In each round, we redistribute B_{res} according to chunk-level allocation scores. Each chunk is then expanded as described in Section 3.3, the retained spans are assembled with q in source order to form the input x_t, and the frozen generator π_0 samples a response $y_t \sim \pi_0(\cdot \mid x_t)$. The quality of this response then corrects the allocation scores for the next round.

Unit relevance g_{ij} determines where to read within a chunk, whereas priorities across chunks start from the probe–chunk similarity prior $p_i = \psi(z, C_i)$. On top of this prior, each chunk maintains two states: an outcome correction $\Delta_i^{(t)}$, which accumulates the response feedback assigned to the chunk by reading share, and a cumulative exposure $N_i^{(t)}$, which records how much of the chunk has been read in earlier rounds. Following the UCB principle (Auer et al., 2002), the allocation score in round t is

$$\mu_i^{(t)} = p_i + \Delta_i^{(t)}, \qquad u_i^{(t)} = \mu_i^{(t)} + \alpha\sqrt{\frac{2\log t}{N_i^{(t)} + \lambda_N}}, \tag{2}$$

where $\mu_i^{(t)}$ exploits past feedback, the second term encourages reading less-exposed chunks, α controls exploration strength, and λ_N is a smoothing term. Feedback comes from comparing each response with the training reference $a^\star$. The reward $\mathcal{R}_t = \mathcal{R}_{\text{roll}}(y_t, a^\star)$ averages substring exact match (SubEM), which checks for the reference in the full response, and normalized word-overlap F1 on the extracted final answer. In the first round, Δ_i and N_i are initialized to zero, and the exploration term vanishes because $\log 1 = 0$, so the initial allocation is determined entirely by the prior p_i. From the second round onward, each response is compared with the previous baseline, and the result corrects subsequent allocations. Because reward levels depend on question difficulty and a single response evaluates an input jointly formed by many chunks, we subtract the question's historical baseline $\overline{\mathcal{R}}_t$ and distribute the relative outcome according to each chunk's actual share of additional reading:

$$\text{Adv}_t = \mathcal{R}_t - \overline{\mathcal{R}}_t, \qquad \omega_i^{(t)} = \frac{r_i^{(t)}}{\sum_j r_j^{(t)}}, \qquad e_i^{(t)} = \frac{r_i^{(t)}}{R_i}, \tag{3}$$

where $e_i^{(t)}$ is normalized by the chunk's own capacity, making exposure comparable across chunks of different lengths. The states are then updated as

$$\begin{aligned} \Delta_i^{(t+1)} &= \lambda_\Delta \Delta_i^{(t)} + \eta_c \,\text{Adv}_t \; \omega_i^{(t)}, \\ N_i^{(t+1)} &= N_i^{(t)} + e_i^{(t)}, \\ \overline{\mathcal{R}}_{t+1} &= (1-\beta_R)\overline{\mathcal{R}}_t + \beta_R \mathcal{R}_t, \end{aligned} \tag{4}$$

where λ_Δ, η_c, and β_R control correction decay, update step size, and baseline smoothing, respectively. Scores in later rounds therefore combine the prior with both past outcomes and exploration. Allocation scores are converted into planned reading amounts through capacity-constrained exponential weights:

$$\tilde{r}_i^{(t)} = \min\left\{ R_i,\ \kappa_t \exp\left(\frac{u_i^{(t)}}{\tau_t}\right) \right\}, \qquad \sum_i \tilde{r}_i^{(t)} = B_{\text{res}}, \tag{5}$$

where allocation exceeding a chunk's capacity is redirected to other chunks and κ_t matches the total to B_{res}. The temperature τ_t controls whether the reading budget is concentrated on a few chunks or distributed more broadly. We select τ_t adaptively in each round from the current allocation scores. Temperature selection and boundary cases are detailed in Appendix A.3.

3.3 Expanding cores to the allocated depth

For each chunk, we construct an expansion sequence that progressively incorporates source text from the left and right boundaries of its core. Each round then determines how far to read along this sequence. The expansion sequence is fixed before candidate generation. Each step of the expansion sequence determines how much text to incorporate on either side. Comparing only the units adjacent to each boundary can be sensitive to local noise: an adjacent sentence may be irrelevant even when useful content lies slightly farther away. We therefore look ahead up to H tokens from the current boundary and compare the relevance density, or relevance per token, on the two sides. For direction $d \in \{\text{L}, \text{R}\}$, let $\ell_{ij,d}(h)$ denote the number of tokens from unit j within the first h tokens read outward, and let $D_{ij,d}$ denote the distance from the current boundary to unit j. The direction scores and the leftward expansion share are:

$$\begin{aligned} J_{i,d} &= \max_{0<h\le H} \frac{1}{h} \sum_j g_{ij}\, \ell_{ij,d}(h) \exp\left(-\frac{D_{ij,d}}{\tau_D}\right), \\ \pi_{i,\text{L}} &= \frac{J_{i,\text{L}}}{J_{i,\text{L}} + J_{i,\text{R}}}. \end{aligned} \tag{6}$$

Here, τ_D is the distance decay scale. The normalization $1/h$ accounts for all text that must be read to reach relevant content, so distant content increases the score only when its relevance outweighs the cost of reading the intervening text. At each step, both sides expand in proportions $\pi_{i,\text{L}}$ and $1-\pi_{i,\text{L}}$, and the scores are recomputed after the boundaries move. In each round, chunk i expands outward from its core along the precomputed sequence, incorporating as much source text as possible without

exceeding the planned reading amount $\widetilde{r}_i^{(t)}$. The resulting contiguous span is retained in the input, and the number of tokens beyond the core defines the actual reading amount $r_i^{(t)}$. Let $W_i(a)$ denote the view obtained with an allowance of a additional tokens. These views satisfy

$$W_i(0) = C_i^{\text{core}}, \qquad W_i(R_i) = C_i, \qquad a \le a' \implies W_i(a) \subseteq W_i(a'). \tag{7}$$

3.4 Preference construction and long-context learning

After T rounds, the pairing score $\mathcal{R}_{\text{pair}}$ averages SubEM and F1 on each candidate's extracted final answer. We denote the indices of the highest- and lowest-scoring candidates by t^+ and t^-, respectively:

$$\begin{aligned} t^+ &\in \arg\max_{1\le t\le T} \mathcal{R}_{\text{pair}}(y_t, a^\star), \qquad y^+ = y_{t^+}, \\ t^- &\in \arg\min_{1\le t\le T} \mathcal{R}_{\text{pair}}(y_t, a^\star), \qquad y^- = y_{t^-}. \end{aligned} \tag{8}$$

The preferred response y^+ and dispreferred response y^- form a preference pair when their scores differ. During training, we reconstruct the input x from the question q and the original context C. We combine DPO, which learns a relative preference for y^+ over y^-, with a supervised objective that directly maximizes the likelihood of y^+ given x. In this way, responses generated through selective reading provide supervision for learning to answer questions from the original long context.

4 Experiments

Our evaluation links adaptive reading to both preference-data production and downstream learning. We compare FidelityFlow-PO with published long-context adaptation methods and with LongMab, examining candidate quality, pair yield, and whether downstream gains persist once training questions and pair counts are matched (Section 4.2). We then isolate the contribution of each component (Section 4.3) and test whether the advantage reduces to finer whole-chunk selection or simply to reading more text (Section 4.4).

4.1 Experimental setup

We use Qwen2.5-7B-Instruct (Qwen et al., 2025) and Llama-3.1-8B-Instruct (Grattafiori et al., 2024) with LongMab's MuSiQue–Wikipedia construction procedure, which produces 8K–16K-token training contexts. Within each backbone, methods share the construction pool, probes, scoring and pairing rules, and 30 candidate calls per question. The default LongMab policy selects four complete 1,500-token chunks using its original updates. FidelityFlow sets its reading reference from LongMab's first prompt, without changing LongMab's subsequent selections. *Natural-Pair* trains on each method's complete valid-pair output. *Matched-ID* retains their common valid question IDs, with identical ordering and a shared 90/10 training/validation split. Both protocols use rank-8 LoRA (Hu et al., 2022) DPO for two epochs at learning rate 2×10^{-5}, retaining the mixed DPO/chosen-response objective and original-context reconstruction in Method. Optimization settings are shared within each backbone. Natural-Pair allows dataset size and composition to vary; consequently, equal epochs need not imply equal update counts. The remaining hyperparameters are provided in Appendix A.5.

The five tasks are MuSiQue (Trivedi et al., 2022), 2WikiMultihopQA (2Wiki) (Ho et al., 2020), MultiFieldQA-En (MFQA), NarrativeQA (Narr.) (Kočiský et al., 2018), and InfiniteBench English book QA (En.QA), with 1,101 questions in total (Bai et al., 2024b; Zhang et al., 2024). Exact match (EM), substring exact match (SubEM), and normalized word-overlap F1 score extracted answers, with missing answer markers receiving zero. Macro averages weight the five tasks equally. Paired 95% intervals use 10,000 within-task question resamples. Reward scores use 0–1, with point differences expressed on the 0–100 scale.

4.2 Preference yield and downstream performance

FidelityFlow-PO outperforms all compared methods on both backbones. FidelityFlow-PO attains the highest macro-F1 and macro-SubEM on both backbones (Table 1), improving over

Table 1: Five-task benchmark comparison (F1). Macro columns weight tasks equally. LongMab and FidelityFlow-PO correspond to the Natural-Pair results in Table 2A; other baselines are reported by Duan et al. (2026). Bold marks the best value per backbone and does not indicate significance.

	MuSiQue	2Wiki	MFQA	Narr.	En.QA	Macro	
Method						F1	SubEM
Qwen2.5-7B-Instruct							
Vanilla LLM	30.52	50.14	45.12	18.29	22.38	33.29	32.03
LongAlpaca	33.90	50.75	45.88	19.72	23.82	34.81	31.90
LongAlign	31.09	52.02	**49.76**	21.98	25.61	36.09	28.88
LongReward-PO	33.37	50.75	44.89	17.67	21.77	33.69	34.51
SeaLong-PO	22.09	36.39	41.08	16.27	16.95	26.56	36.71
Logo-PO	36.48	57.20	49.18	18.28	28.47	37.92	34.43
LongFaith-PO	43.38	53.93	38.08	17.03	18.55	34.19	34.85
LongMab	43.43	59.16	46.70	24.74	30.23	40.85	36.43
FidelityFlow	**47.32**	**62.27**	48.87	**25.11**	**30.45**	**42.80**	**36.87**
Llama-3.1-8B-Instruct							
Vanilla LLM	36.56	65.45	44.34	26.61	32.19	41.03	33.23
LongAlpaca	38.36	63.64	40.67	26.88	30.58	40.03	32.80
LongAlign	37.62	59.35	48.49	26.38	24.85	39.34	33.99
LongReward-PO	41.41	63.23	44.70	25.92	24.20	39.89	34.34
SeaLong-PO	40.52	67.07	44.12	26.45	32.06	42.04	33.58
Logo-PO	46.58	61.74	44.39	28.05	**32.69**	42.69	34.13
LongFaith-PO	49.23	**72.80**	48.10	23.26	22.35	43.15	34.79
LongMab	51.40	61.10	51.64	27.70	30.83	44.53	40.01
FidelityFlow	**54.36**	63.12	**51.98**	**30.66**	30.38	**46.10**	**40.72**

LongMab by 1.95 macro-F1 points on Qwen and 1.57 on Llama. The gains are most pronounced on multi-hop tasks, whose answers depend on composing evidence dispersed across sources. This is precisely the regime that whole-chunk selection handles poorly, since admitting one bridging source forces another out of the input, whereas persistent cores keep every source within reach. The improvements also extend beyond the construction source, most notably to NarrativeQA on Llama, indicating that the learned behavior transfers rather than specializing to the training distribution.

Stronger candidates from the same generation budget. With an identical budget of 30 candidate calls per question, FidelityFlow raises Best-of-30, the mean highest rollout reward per question, by 5.66 points on Qwen and 2.40 on Llama, and increases pair yield by 3.84 and 1.23 percentage points (Table 2A). These gains reflect a more effective use of each call. Redistributing reading depth, rather than toggling entire sources, enlarges the set of evidence configurations the search can reach, so stronger candidates emerge at no additional generation cost.

The advantage stems from pair quality. Natural-Pair reflects the full pipeline and therefore conflates the quality of each pair with the number of pairs produced. Matched-ID separates the two by training both methods on their common question IDs with identical ordering and pair counts (Table 2B). Under this control the margin widens rather than narrows. The decomposition is informative: restricting to shared questions costs LongMab 4.40 and 4.03 points, but FidelityFlow-PO only 0.91 and 2.00. FidelityFlow's pairs thus carry more learning signal per example, making its downstream performance far less dependent on data volume. Paired differences are positive on all five tasks for both backbones, and the macro-level 95% intervals exclude zero on both (Figure 3).

4.3 Ablation studies

We ablate each component separately (Table 3). 30 calls per question. *Any@30* measures complete evidence coverage in at least one prompt. Each variant's pairs are used for DPO, followed by five-task evaluation. All ablations reduce all three metrics, and Best-of-30 ranks the variants identically to post-DPO macro-F1, linking candidate quality to downstream performance in these experiments.

Table 2: andidate generation and downstream learning. (A) Natural-Pair, with full-pool construction statistics. (B) Matched-ID: both methods use their common valid question IDs with identical ordering and pair counts. Best-of-30 is the mean highest rollout reward per question (0–1 scale). Pair yield is the percentage of pool questions yielding a valid pair. F1 is the five-task macro average. Bold marks the higher Best-of-30 or F1 per backbone and panel.

A. Natural-Pair

Backbone	Policy	Best-of-30	Pair yield (%)	F1	Peak (GiB)
Qwen	LongMab	0.7015	78.06	40.85	70.94
	FidelityFlow	**0.7581**	81.90	**42.80**	71.12
Llama	LongMab	0.8574	91.42	44.53	67.68
	FidelityFlow	**0.8814**	92.65	**46.10**	69.51

B. Matched-ID

Backbone	Policy	Best-of-30	Pair yield (%)	F1	Peak (GiB)
Qwen	LongMab	0.8653	74.36	36.45	69.83
	FidelityFlow	**0.8907**	74.36	**41.89**	70.61
Llama	LongMab	0.8934	88.97	40.50	67.12
	FidelityFlow	**0.9076**	88.97	**44.10**	68.45

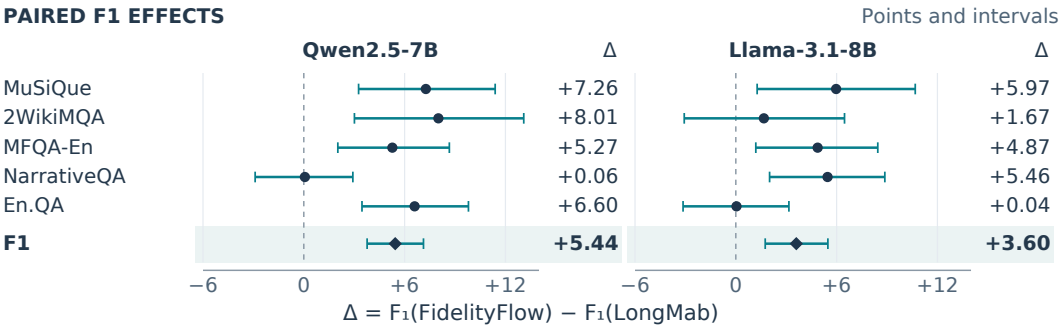

Figure 3: Per-task F1 gains under Matched-ID. Paired differences (FidelityFlow-PO − LongMab) with 95% intervals from question-level resampling; the bottom row is the macro average. Task-level intervals are wider, reflecting smaller per-task sample sizes.

Access and depth are complementary. The two structural ablations each remove one of the axes that FidelityFlow decouples. *Without core reservation* re-couples access to depth: chunks that receive no additional reading vanish from the input, recovering the exclusion behavior of whole-chunk selection. Macro-F1 drops by 4.51 points. *Uniform residual allocation* does the opposite, preserving every core but spreading the remaining allowance evenly, and incurs an even larger loss of 6.66 points. Coverage is therefore necessary but not sufficient. The benefit arises from pairing persistent access with depth concentrated where the evidence lies.

Outcome feedback drives adaptation. *Reward-blind* fixes $\Delta_i = 0$ while retaining every other component; it lowers Best-of-30 from 0.758 to 0.634 and macro-F1 by 4.96 points. *Static-prior*, which samples all responses from a single prior-based view, is the weakest variant on every metric, trailing by 7.30 points. The prior determines where the search begins, but outcome feedback determines where it converges; even a well-informed initialization cannot substitute for the closed loop between generation and evaluation.

Credit assignment sharpens attribution. *Equal-share outcome credit* splits each update equally among expanded chunks. The resulting 2.78-point drop is the smallest among the ablations yet consistent across all three metrics. Because a single reward evaluates an input composed of many chunks, weighting credit by realized reading share attributes each outcome more precisely to the text that produced it.

Table 3: Component ablations and a static-prior baseline. Effects of FidelityFlow components on preference generation and downstream five-task macro-F1 after DPO.

	Preference generation		After DPO
Variant	Any@30 (%) ↑	Best-of-30 ↑	Macro-F1 ↑
FidelityFlow	**62.21**	**0.7581**	**42.80**
Equal-share outcome credit	60.46	0.7103	40.02
Without core reservation	58.91	0.6762	38.29
Reward-blind	56.48	0.6340	37.84
Uniform residual allocation	56.93	0.5998	36.14
Static-prior	54.13	0.5753	35.50

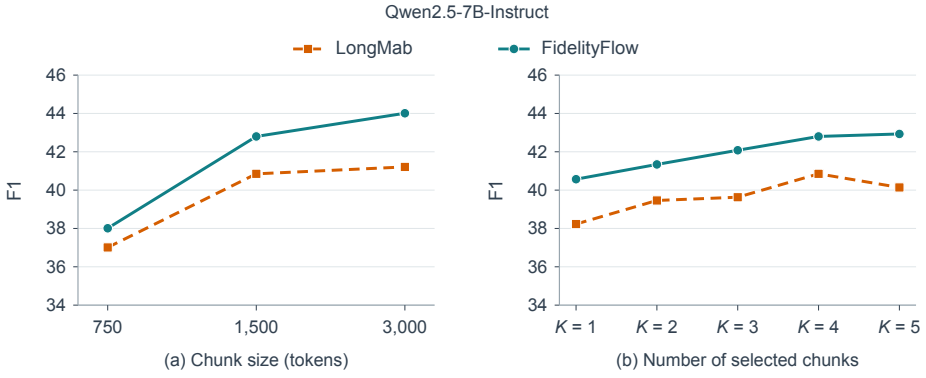

Figure 4: Chunk configuration and downstream performance on Qwen. (a) Chunk sizes of 750, 1,500, and 3,000 tokens with $K = 4$. (b) $K = 1, \ldots, 5$ with 1,500-token chunks. Here, K is the number of chunks selected by LongMab.

4.4 Robustness to chunk granularity and reading budget

A natural question is whether FidelityFlow's advantage is an artifact of the chunk partition or of reading more text. Figure 4 examines both factors. All runs use Qwen and question IDs common to both methods and all settings. With $K = 4$, 750-, 1,500-, and 3,000-token chunks correspond to nominal budgets of 3K, 6K, and 12K tokens. FidelityFlow-PO outperforms LongMab throughout, and its margin grows from 1.00 to 2.80 points as chunks become coarser. Coarser chunks can increase the amount of irrelevant text admitted by whole-chunk selection, whereas finer partitions require more persistent cores. With 1,500-token chunks and $K = 1, \ldots, 5$ (1.5K–7.5K tokens), FidelityFlow-PO leads at every budget. At 3K tokens ($K = 2$), it already surpasses LongMab's best result, attained at 6K tokens ($K = 4$). The two methods also respond differently to additional context. A fifth complete chunk lowers LongMab from 40.85 to 40.14, whereas FidelityFlow-PO continues to improve, from 42.80 to 42.93. These results suggest that selectively allocating additional reading can be more effective than including more complete chunks.

5 Conclusion

We introduced FidelityFlow-PO, advancing LongMab's chunk-as-arm formulation from binary source selection to feedback-driven control of reading depth. FidelityFlow-PO decouples source coverage from reading depth by retaining compact, query-conditioned cores and selectively expanding them using response feedback. This relaxes the trade-off imposed by whole-chunk selection, where excluding a source can break an evidence chain, while including it can expose candidate generation to distracting context. Experiments with Qwen and Llama show improved candidate quality and higher downstream macro-F1 across five long-context benchmarks. Controlled comparisons and ablations support the need for both mechanisms. All adaptive reading occurs during offline preference construction, requiring no additional context search at inference.

AI USE STATEMENT

During manuscript preparation, generative AI tools were used solely for language polishing, including grammar, wording, and readability. The authors reviewed all AI-assisted revisions and take full responsibility for the final manuscript.

REFERENCES

Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47:235–256, 2002. doi: 10.1023/A:1013689704352. URL https://link.springer.com/article/10.1023/A:1013689704352.

Yushi Bai, Xin Lv, Jiajie Zhang, Yuze He, Ji Qi, Lei Hou, Jie Tang, Yuxiao Dong, and Juanzi Li. LongAlign: A recipe for long context alignment of large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 1376–1395. Association for Computational Linguistics, 2024a. doi: 10.18653/v1/2024.findings-emnlp.74. URL https://aclanthology.org/2024.findings-emnlp.74/.

Yushi Bai, Xin Lv, Jiajie Zhang, et al. LongBench: A bilingual, multitask benchmark for long context understanding. In *ACL*, pp. 3119–3137, 2024b.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. LongLoRA: Efficient fine-tuning of long-context large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/211ab571cc9f3802afa6ffff52ae3e5b-Abstract-Conference.html.

Shaohua Duan, Pengcheng Huang, Xinze Li, et al. Chunks as arms: Multi-armed bandit-guided sampling for long-context LLM preference optimization. In *ACL*, pp. 3559–3575, 2026. URL https://aclanthology.org/2026.acl-long.161/.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. URL https://arxiv.org/abs/2407.21783.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 6609–6625. International Committee on Computational Linguistics, 2020. doi: 10.18653/v1/2020.coling-main.580. URL https://aclanthology.org/2020.coling-main.580/.

Edward J. Hu, Yelong Shen, Phillip Wallis, et al. LoRA: Low-rank adaptation of large language models. In *ICLR*, 2022.

Huiqiang Jiang, Qianhui Wu, Xufang Luo, et al. LongLLMLingua: Accelerating and enhancing LLMs in long context scenarios via prompt compression. In *ACL*, pp. 1658–1677, 2024.

Tomáš Kočiský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. The NarrativeQA reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328, 2018. doi: 10.1162/tacl_a_00023. URL https://aclanthology.org/Q18-1023/.

Siheng Li, Cheng Yang, Zesen Cheng, Lemao Liu, Mo Yu, Yujiu Yang, and Wai Lam. Large language models can self-improve in long-context reasoning. *arXiv preprint arXiv:2411.08147*, 2024. URL https://arxiv.org/abs/2411.08147.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024. doi: 10.1162/tacl_a_00638.

Xiaoran Liu, Ruixiao Li, Mianqiu Huang, Zhigeng Liu, Yuerong Song, Qipeng Guo, Siyang He, Qiqi Wang, Linlin Li, Qun Liu, Ziwei He, Yaqian Zhou, Xuanjing Huang, and Xipeng Qiu. Thus spake long-context large language model. *arXiv preprint arXiv:2502.17129*, 2025. URL https://arxiv.org/abs/2502.17129.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, et al. LLMLingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. In *Findings of ACL*, pp. 963–981, 2024.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. YaRN: Efficient context window extension of large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=wHBfxhZu1u.

Qwen, An Yang, Baosong Yang, Beichen Zhang, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2025. URL https://arxiv.org/abs/2412.15115.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *NeurIPS*, 2023. URL https://arxiv.org/abs/2305.18290.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H. Chi, Nathanael Schärli, and Denny Zhou. Large language models can be easily distracted by irrelevant context. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 31210–31227. PMLR, 2023.

Zecheng Tang, Zechen Sun, Juntao Li, Qiaoming Zhu, and Min Zhang. LOGO—long context alignment via efficient preference optimization. In *ICML*, volume 267 of *PMLR*, pp. 58944–58968, 2025.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022. doi: 10.1162/tacl_a_00475. URL https://aclanthology.org/2022.tacl-1.31/.

Cehao Yang, Xueyuan Lin, Chengjin Xu, et al. LongFaith: Enhancing long-context reasoning in LLMs with faithful synthetic data. In *Findings of ACL*, pp. 3236–3256, 2025.

Jiajie Zhang, Zhongni Hou, Xin Lv, Shulin Cao, Zhenyu Hou, Yilin Niu, Lei Hou, Yuxiao Dong, Ling Feng, and Juanzi Li. LongReward: Improving long-context large language models with AI feedback. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3718–3739. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.187. URL https://aclanthology.org/2025.acl-long.187/.

Xinrong Zhang, Yingfa Chen, Shengding Hu, et al. ∞Bench: Extending long context evaluation beyond 100k tokens. In *ACL*, pp. 15262–15277, 2024.

A Implementation and Evaluation Details

This appendix describes the reading controller, preference construction, training, and evaluation, using the notation of the main text. We write $a^\star$ for the set of accepted reference answers. The language-model policies π_0 and π_θ should not be confused with $\pi_{i,L}$, the leftward expansion share of chunk i.

A.1 Pipeline and supervision

FidelityFlow-PO has three stages: preprocessing, offline preference construction, and LoRA DPO. During preprocessing, LongMab generates a reasoning probe from the question, the original context, and the reference answer, and discards items whose probe does not reach the reference. FidelityFlow-PO reuses these probes together with LongMab's stored probe–chunk similarities. The reference answer therefore influences context selection indirectly through the probe, in addition to its direct roles in reward computation and pair selection.

During preference construction, each prompt fills the shared response template with the question and the selected source text only. Neither the reference answer nor the probe is added to the prompt. Each question has its own controller, and the generator π_0 remains frozen. Model parameters are updated only in the DPO stage. At test time, the merged model π_θ reads ordinary long-context inputs without the controller, the probe, or any feedback.

A.2 Cores, views, and the reading budget

Cores and views. Unit relevance is computed with a frozen `openbmb/MiniCPM-Embedding` encoder. Units follow sentence boundaries aligned with tokenizer offsets, and every cut, including one inside a unit, falls between tokens. Overly long sentences are split recursively, preferably at internal punctuation. The core of a chunk is its most relevant unit (the anchor), extended by the following unit when the anchor is only a heading, a label, or a short fragment. The expansion schedule of each chunk is fixed per question, so a larger allowance always yields a view that contains every smaller view of the same chunk, and the full capacity recovers the entire chunk. Nesting is defined over allowances, not rounds, so feedback may shorten one chunk's view in a later round while lengthening another's. Views are verbatim substrings of their chunks and are assembled in source order. Cores give every chunk an entry point but do not guarantee that complete supporting evidence is visible.

Reading budget. Let P_{ref} be the token count of LongMab's first-round Top-K prompt, and let P_{core} be that of the identically formatted prompt containing only the cores. Both counts include the question, the instructions, and the chat template. For a chunk of length L_i with a core of length A_i,

$$R_i = L_i - A_i, \qquad B_{\mathrm{res}} = \min\Big\{\sum_i R_i,\ \max\big(0,\ P_{\mathrm{ref}} - P_{\mathrm{core}}\big)\Big\}. \tag{9}$$

An item with $P_{\mathrm{core}} > P_{\mathrm{ref}}$ is treated as infeasible, and cores are never dropped to meet the cap. Otherwise, continuous allocations are rounded to integer token allowances and converted into source spans that end at token boundaries.

Prompt repair. Token counts are not additive across joined spans, so the assembled prompt is retokenized. If it exceeds the cap, repair shrinks the frontier with the lowest marginal relevance gain. If space remains, it applies the feasible expansion with the largest gain per added token. A prompt that violates the cap, or falls short of it by more than a small tolerance, is rejected. Budget matching therefore fixes only the prompt cap, while source-token counts, output lengths, and compute can still differ.

A.3 Allocation

Support target. For a nonnegative vector $\boldsymbol{v}$, the effective support is the exponentiated entropy of its normalized mass,

$$S_{\mathrm{eff}}(\boldsymbol{v}) = \exp\Big(-\sum_{i:\, w_i>0} w_i \log w_i\Big), \qquad w_i = \frac{v_i}{\sum_j v_j}, \tag{10}$$

with $S_{\text{eff}}(\mathbf{0}) = 0$. Uniform mass over k chunks gives $S_{\text{eff}} = k$. It measures how widely residual reading is spread. The number of visible chunks is fixed, since every prompt retains all cores. Let $\mathcal{I} = \{i : R_i > 0\}$ and $n_{\mathcal{I}} = |\mathcal{I}|$. For $B_{\text{res}} > 0$, the target support in round t is

$$k_{\text{base}} = \text{clip}\Big(\frac{B_{\text{res}}}{\text{median}_{i\in\mathcal{I}}\, R_i},\, 1,\, n_{\mathcal{I}}\Big),$$
$$k_t = \text{clip}\Big(k_{\text{base}} + s_0\Big(1 - \frac{t-1}{T-1}\Big),\, 1,\, n_{\mathcal{I}}\Big), \tag{11}$$

where $s_0 \geq 0$ is the initial extra support and the fraction is taken as zero when $T = 1$. The target decreases linearly to the budget-equivalent support k_{base}, so early rounds spread reading more widely.

Temperature selection. Let $\hat{u}_i^{(t)}$ denote the allocation score, which equals $u_i^{(t)}$ except under complete ties (see below), and let $\widetilde{\boldsymbol{r}}_t(\tau)$ be the planned allocation produced by the capped exponential rule of the main text at temperature τ. Bisection on the support of the uncapped softmax provides an initial temperature, which is added to a fixed log-spaced grid to form the candidate set $\mathcal{T}_t$. We then select

$$\tau_t \in \arg\min_{\tau\in\mathcal{T}_t} \big|S_{\text{eff}}\big(\widetilde{\boldsymbol{r}}_t(\tau)\big) - k_t\big|, \tag{12}$$

breaking ties by the uncapped-support error and then by the lower temperature. The grid is needed because capacity saturation creates plateaus in the capped support. The allocation temperature τ_t is distinct from the sampling temperature of π_0.

Ties and boundary cases. When all active scores are tied, a uniform split cannot realize targets below $n_{\mathcal{I}}$. The allocator then uses rank-spaced surrogate scores ordered by the prior p_i and then by chunk index, while the recorded UCB scores remain unchanged. A zero budget yields no residual reading, and a budget equal to the total capacity reads every chunk in full.

A.4 Scoring and preference pairs

Metrics. Let $\text{norm}(\cdot)$ lowercase a string, remove ASCII punctuation and the articles *a*, *an*, and *the*, and collapse whitespace. For a prediction b and nonempty normalized references,

$$\text{EM}(b, a^\star) = \max_{a\in a^\star} \mathbf{1}\big\{\text{norm}(b) = \text{norm}(a)\big\},$$
$$\text{SubEM}(b, a^\star) = \max_{a\in a^\star} \mathbf{1}\big\{\text{norm}(a) \text{ is a substring of } \text{norm}(b)\big\}, \tag{13}$$

and, with $o(b, a)$ the multiset overlap between the normalized words of b and a,

$$\text{F1}(b, a^\star) = \max_{a\in a^\star} \frac{2\, o(b, a)}{|\,\text{norm}(b)|_{\text{w}} + |\,\text{norm}(a)|_{\text{w}}}, \tag{14}$$

where $|\cdot|_{\text{w}}$ counts whitespace-delimited words rather than subword tokens. An empty prediction scores zero. The three metrics capture different properties: against the reference "New York", the prediction "New York City" has EM $= 0$, F1 $= 0.8$, and SubEM $= 1$, whereas "York New" has EM $= 0$, F1 $= 1$, and SubEM $= 0$.

Answer extraction and stage scores. Extraction is case-insensitive and searches the response for three markers in order of precedence: `answer:`, `the answer is:`, and `final answer is:`. It returns the text after the last occurrence of the first marker present and fails if no marker appears or nothing follows it. Let $\chi(y) \in \{0, 1\}$ indicate successful extraction from a response y, and let $b(y)$ be the extracted answer. The rollout reward and the pairing score are

$$R_{\text{roll}}(y, a^\star) = \tfrac{1}{2}\big[\text{SubEM}(y, a^\star) + \chi(y)\,\text{F1}\big(b(y), a^\star\big)\big],$$
$$R_{\text{pair}}(y, a^\star) = \tfrac{1}{2}\,\chi(y)\big[\text{SubEM}\big(b(y), a^\star\big) + \text{F1}\big(b(y), a^\star\big)\big]. \tag{15}$$

Since the rollout reward evaluates SubEM on the full response, a missing marker removes only its F1 term, whereas the pairing score removes both. Test-time evaluation computes EM, F1, and SubEM on the extracted answer and assigns zero to all three when extraction fails.

Table 4: Hyperparameters not stated in the main text. The backbone-specific FidelityFlow-PO settings were selected in preliminary tuning; the LongMab baselines use their released settings.

Setting	Qwen2.5-7B	Llama-3.1-8B
Exploration strength α	1.0	1.0
Exposure smoothing λ_N	1.0	1.0
Initial extra support s_0	1	2
Update step size η_c	0.5	0.294
Lookahead H (tokens)	256	256
Distance decay scale τ_D (tokens)	128	128
Expansion step size (tokens)	64	64
FidelityFlow-PO sampling temperature	1.0	0.936
Maximum new tokens (generation and evaluation)	1,024	1,024
LoRA scaling / dropout	16 / 0	16 / 0
LoRA target modules	all linear layers	all linear layers
Batch size $\times$ gradient accumulation	1×4	1×4
Training sequence cutoff (tokens)	8,500	8,500
Evaluation input limit (tokens)	100,000	100,000

Valid pairs. For question j with candidates $y_{j1}, \ldots, y_{jT}$, a pair is formed only if

$$g_j = \max_t R_{\text{pair}}(y_{jt}, a_j^\star) - \min_t R_{\text{pair}}(y_{jt}, a_j^\star) > 0. \tag{16}$$

The constructor selects one maximizing and one minimizing candidate, breaking ties at random, and uses the complete responses, including their reasoning, as the chosen and rejected targets. If all candidates receive the same score, no pair is formed, even when all of them are correct. The gap determines eligibility only and does not weight the loss.

A.5 Training

The training prompt x is rebuilt from the original question and context rather than from either candidate's reading view. The pair constructor applies its own prompt-length limit: it reserves space for the question and the longer response and, when necessary, truncates the context by keeping its beginning and end. Such truncation can remove evidence that was visible during generation. This limit is separate from the trainer's sequence cutoff. For a pair (x, y^+, y^-), let

$$z_\theta = \log \frac{\pi_\theta(y^+ \mid x)}{\pi_{\text{ref}}(y^+ \mid x)} - \log \frac{\pi_\theta(y^- \mid x)}{\pi_{\text{ref}}(y^- \mid x)}. \tag{17}$$

We minimize the sigmoid DPO loss with an auxiliary supervised term on the chosen response,

$$\mathcal{L} = \mathbb{E}_{(x,y^+,y^-)\sim\mathcal{D}}\big[-\log \sigma(\beta z_\theta)\big] + \gamma\, \mathcal{L}_{\text{chosen}}, \tag{18}$$

where σ is the logistic function. Token masking and loss reduction follow the training backend. FidelityFlow-PO keeps LongMab's training objective and performs no online reward-gradient updates. Each compared model is trained independently from the original backbone: only the LoRA adapters are updated, the reference policy stays frozen, and the adapters are merged before evaluation. We set $\beta = 0.1$ for both backbones, and $\gamma = 0.2$ for Qwen and 0.1 for Llama. Optimization settings are shared by all methods within a backbone, whereas generation settings are backbone-specific. Table 4 lists the remaining hyperparameters.

A.6 Comparison protocols

Let $\mathcal{G}$ be the generation pool of a backbone, i.e., the items retained after preprocessing, and let $\mathcal{V}_m \subseteq \mathcal{G}$ contain the items for which method m yields a valid pair. *Natural-Pair* trains each method on its own $\mathcal{V}_m$ and therefore reflects both pair quality and yield. *Matched-ID* trains both methods on $\mathcal{M} = \mathcal{V}_1 \cap \mathcal{V}_2$, each with its own responses, sharing the item order, the training/validation assignment, and the training schedule. Response lengths and reconstructed prompts may still differ, so the number of training tokens is not matched.

Matched statistics reuse the original rollouts restricted to $\mathcal{M}$. This conditioning changes the population, so a higher Best-of-30 on $\mathcal{M}$ does not indicate better generation.

A.7 GENERATION METRICS

For a question set $\mathcal{S}$, let $M_j = \max_t R_{\text{roll}}(y_{jt}, a_j^\star)$ and define

$$\text{Best-of-30}(\mathcal{S}) = \frac{1}{|\mathcal{S}|}\sum_{j\in\mathcal{S}} M_j, \qquad \text{Any}_{\text{reward}}@30(\mathcal{S}) = \frac{100}{|\mathcal{S}|}\sum_{j\in\mathcal{S}} \mathbf{1}\{M_j = 1\}. \tag{19}$$

Best-of-30 is on the 0–1 reward scale and is not comparable to test-time F1. Replacing $M_j = 1$ with $M_j > 0$ gives the partial-success rate $\text{Any}_{>0}@30$. Given annotated support spans $\mathcal{E}_j$ and the source text V_{jt} visible in round t, evidence coverage is

$$\text{Any}_{\text{evidence}}@30(\mathcal{S}) = \frac{100}{|\mathcal{S}|}\sum_{j\in\mathcal{S}} \mathbf{1}\big\{\exists\, t:\ e \subseteq V_{jt} \text{ for all } e \in \mathcal{E}_j\big\}, \tag{20}$$

which requires all support to be visible within a single round. For the reward-based variants, $\mathbf{1}\{M_j = 1\} \le M_j \le \mathbf{1}\{M_j > 0\}$ implies the consistency check

$$\frac{\text{Any}_{\text{reward}}@30}{100} \le \text{Best-of-30} \le \frac{\text{Any}_{>0}@30}{100}. \tag{21}$$

Pair yield is $\text{PairYield}_m = 100\,|\mathcal{V}_m|/|\mathcal{G}|$, computed on the generation pool before the training/validation split. For Matched-ID, $100\,|\mathcal{M}|/|\mathcal{G}|$ is a shared retention rate. Yield counts only questions with a positive score gap, so a question whose candidates all attain the maximum pairing score yields no pair.

A.8 EVALUATION AND UNCERTAINTY

Test prompts that exceed the input limit (Table 4) are truncated before the chat template by keeping their beginning and end, and responses are decoded greedily. Qwen2.5-7B-Instruct is evaluated with YaRN RoPE scaling (Peng et al., 2024) (factor 4), enabled only at evaluation time; Llama-3.1-8B-Instruct uses its native RoPE scaling. Published baseline results may rely on different evaluation settings.

Let $s_{dj} \in [0, 1]$ be the EM, F1, or SubEM score of question j in task d, which has n_d questions. The macro score weights tasks equally,

$$\bar{s}_d = \frac{1}{n_d}\sum_{j=1}^{n_d} s_{dj}, \qquad \text{Macro}(s) = \frac{100}{5}\sum_{d=1}^{5} \bar{s}_d, \tag{22}$$

and is therefore not the pooled mean over all questions. Each macro metric is computed from its own per-question scores using unrounded values. For a paired comparison, $D = \text{Macro}(s^{\text{FF}}) - \text{Macro}(s^{\text{control}})$. Each bootstrap replicate resamples question IDs with replacement within every task, applies the same IDs to both models, and recomputes D. We report the 95% percentile interval, which captures evaluation-sample uncertainty for the fixed trained models.

B THEORETICAL ANALYSIS

We analyze two properties of FidelityFlow-PO: the evidence access provided by persistent cores, and an optimization view of its continuous allocation rule. Neither result assumes that the generator uses visible evidence correctly.

B.1 A COVERAGE LIMITATION OF WHOLE-CHUNK SELECTION

Let $C_1, \ldots, C_n$ be the source chunks and $C_i^{\text{core}} \subseteq C_i$ the fixed core of chunk i. For a set $\mathcal{E}$ of required support spans, a view V, i.e., the source text visible in one prompt, has complete coverage if $\text{Cov}(V) = \mathbf{1}\{e \subseteq V \text{ for all } e \in \mathcal{E}\} = 1$. A hard-$K$ policy shows at most K complete chunks per round, whereas a FidelityFlow-PO view retains every core and expands selected chunks within the prompt cap.

Proposition B.1 (Conditional coverage separation) *Suppose there is a set $\mathcal{J}$ of more than K chunk indices such that each C_i with $i \in \mathcal{J}$ contains a required span $e_i \in \mathcal{E}$ not covered by the other chunks, i.e., $e_i \not\subseteq \bigcup_{l \neq i} C_l$. Suppose further that every required span lies in some core and that the core-only prompt fits the cap. Then every hard-K view V^{hard} and every valid FidelityFlow-PO view V^{FF} satisfy*

$$\mathrm{Cov}(V^{\mathrm{hard}}) = 0, \qquad \mathrm{Cov}(V^{\mathrm{FF}}) = 1. \tag{23}$$

Proof. A hard-K view is a union of at most K chunks and thus omits some C_i with $i \in \mathcal{J}$. It is then contained in $\bigcup_{l \neq i} C_l$, which does not cover e_i, so $\mathrm{Cov}(V^{\mathrm{hard}}) = 0$. Every valid FidelityFlow-PO view contains all cores, and every required span lies in some core, so $\mathrm{Cov}(V^{\mathrm{FF}}) = 1$; such views exist because the core-only prompt fits the cap. □

Exclusivity is defined with respect to the union of the other chunks because adjacent chunks may overlap. The limitation persists under perfect chunk ranking and any number of rounds, since each round generates from its own prompt and evidence is not pooled across rounds. The result concerns only the visibility of the required support and makes no claim about the correctness of the generated answer.

B.2 An optimization view of continuous allocation

Fix a round and write $B = B_{\mathrm{res}}$. Let $\mathcal{I} = \{i : R_i > 0\}$, assume $0 < B < \sum_{i \in \mathcal{I}} R_i$, let $\hat{u}_i$ be the finite allocation score (Appendix A.3), and let $\tau > 0$. Writing continuous allocations as $r_i = B w_i$, define

$$\mathcal{W} = \Big\{ \boldsymbol{w} : \sum_{i \in \mathcal{I}} w_i = 1, \;\; 0 \le w_i \le R_i / B \Big\}, \qquad H(\boldsymbol{w}) = -\sum_{i \in \mathcal{I}} w_i \log w_i, \tag{24}$$

with the convention $0 \log 0 = 0$.

Proposition B.2 (Entropy-regularized allocation) *In exact arithmetic, capped proportional filling returns the unique solution of*

$$\boldsymbol{w}^\star = \arg\max_{\boldsymbol{w} \in \mathcal{W}} \Big\{ \sum_{i \in \mathcal{I}} \hat{u}_i w_i + \tau H(\boldsymbol{w}) \Big\}, \tag{25}$$

whose token allocation is

$$r_i^\star = \min\big\{ R_i, \; \kappa \exp(\hat{u}_i / \tau) \big\}, \qquad \sum_{i \in \mathcal{I}} r_i^\star = B, \tag{26}$$

for the unique constant $\kappa > 0$ that meets the budget.

Proof. The objective is strictly concave and $\mathcal{W}$ is convex and compact, so the maximizer exists and is unique. The point $w_i = R_i / \sum_{l \in \mathcal{I}} R_l$ satisfies all inequality constraints strictly, so Slater's condition holds and the KKT conditions characterize $\boldsymbol{w}^\star$. Since $\partial H / \partial w_i \to +\infty$ as $w_i \downarrow 0$ and every upper bound R_i / B is positive, moving a small amount of mass to a zero coordinate would increase the objective; hence $w_i^\star > 0$ and the lower bounds are inactive. With multipliers ν for the equality constraint and λ_i for the upper bounds, the KKT conditions read

$$\hat{u}_i - \tau\big(1 + \log w_i^\star\big) - \nu - \lambda_i = 0, \qquad \lambda_i \ge 0, \qquad \lambda_i\big(w_i^\star - R_i / B\big) = 0. \tag{27}$$

Let $\kappa = B \exp(-1 - \nu/\tau)$. If $w_i^\star < R_i / B$, then $\lambda_i = 0$ and $r_i^\star = \kappa \exp(\hat{u}_i / \tau) < R_i$; if $w_i^\star = R_i / B$, then $\lambda_i \ge 0$ gives $R_i \le \kappa \exp(\hat{u}_i / \tau)$. Both cases yield Eq. (26). The map $\kappa \mapsto \sum_{i \in \mathcal{I}} \min\{R_i, \kappa \exp(\hat{u}_i / \tau)\}$ is continuous, vanishes as $\kappa \to 0$, and increases strictly until it reaches $\sum_{i \in \mathcal{I}} R_i > B$, so exactly one κ meets the budget.

Capped proportional filling repeatedly distributes the remaining budget over the unsaturated coordinates in proportion to $\exp(\hat{u}_i / \tau)$ and saturates every coordinate whose share reaches its capacity. A saturated coordinate absorbs no more than its proportional share, so the proportionality constant never decreases across passes. At termination, the unsaturated coordinates share a common constant, and every saturated coordinate satisfies $R_i \le \kappa \exp(\hat{u}_i / \tau)$ for that constant. The output thus has the form of Eq. (26) and meets the budget, so it equals $\boldsymbol{r}^\star$. □

The proposition also explains how temperature controls the spread of reading. For fixed scores, capacities, and budget, let $\boldsymbol{w}^{(1)}$ and $\boldsymbol{w}^{(2)}$ be the maximizers at $\tau_1 < \tau_2$. Adding the two optimality inequalities, each stating that a maximizer is at least as good as the other under its own temperature, gives

$$(\tau_2 - \tau_1)\big[H(\boldsymbol{w}^{(2)}) - H(\boldsymbol{w}^{(1)})\big] \geq 0. \tag{28}$$

Raising the temperature therefore never decreases the continuous effective support $\exp H(\boldsymbol{w})$, although capacities can create plateaus. This justifies calibrating τ_t against a target spread, but a bounded temperature range need not attain every target.