T*: Progressive Block Scaling for MDM Through Trajectory Aware RL

Anonymous ACL submission

Abstract

We present **T***, a simple TRACERL-based training curriculum for progressive block-size scaling in masked diffusion language models (MDMs). Starting from an AR-initialized small-block MDM, **T*** transitions smoothly to larger blocks, enabling higher-parallelism decoding with minimal performance degradation on math reasoning benchmarks. Moreover, further analysis suggests that **T*** can converge to an alternative decoding schedule $\hat{\mathrm{S}}$ that achieves comparable performance.

1 Introduction

Before the current wave of large language models (LLMs), bidirectional Transformers trained with masked language modeling were a widely adopted backbone for NLP systems, with BERT and its optimized variants as canonical examples (Devlin et al., 2019; Liu et al., 2019). Today, autoregressive (AR) modeling via next-token prediction dominates both scaling practice and deployed systems (Brown et al., 2020; Touvron et al., 2023).

In parallel, diffusion language models have begun to emerge as viable alternatives or complements to the autoregressive decoding paradigm. Masked diffusion models stochastically mask a subset of tokens under a ratio-parameterized corruption process and optimize cross-entropy on masked positions to recover the original sequence (Sahoo et al., 2024). For scalability, recent work initializes diffusion LMs from pretrained autoregressive LLMs and trains them with random-mask diffusion objectives (Ye et al., 2025; Cheng et al., 2025). At inference time, they often adopt blockwise decoding that denoises tokens within each block while generating blocks autoregressively to preserve global coherence (Arriola et al., 2025). In this setting, the block size is a control parameter that interpolates between stronger AR-like causality and higher-parallel masked updates.

Within each block, the denoising schedule is typically determined by model confidence. Let $\mathcal{M}^{(s)}$ be the set of masked positions at denoising step s, and let the model predict a token distribution at each masked position. A common heuristic defines a confidence score

$$c_i^{(s)} = \max_{v \in \mathcal{V}} p_\theta(x_i = v \mid x^{(s)}, Q), \quad i \in \mathcal{M}^{(s)},$$
$$U^{(s)} = \{i \in \mathcal{M}^{(s)} : c_i^{(s)} \geq \eta\}, \tag{1}$$

and then materializes tokens in $U^{(s)}$ (e.g., via argmax or sampling), while leaving the rest masked for subsequent refinement.

When examining the **SDAR** series models across scales (1.7B–30B) and block sizes (4–64), we find that math-centric reasoning becomes increasingly sensitive to larger blocks: accuracy generally degrades as block size B grows, with more pronounced drops for smaller models, which is also reported by Cheng et al. (2025). Under an absorbing-state corruption, the negative-ELBO objective reduces to a reweighted cross-entropy on masked positions:

$$\mathcal{L}(\theta) = \mathbb{E}_{x_0 \sim p_{\text{data}}, x_t \sim q(x_t|x_0), t \sim \mathrm{U}(0,1)}$$
$$\left[-\frac{1}{t} \sum_{\ell=1}^{L} \mathbf{1}_{\{x_{t,\ell} = \texttt{[/MASK]}\}} \cdot \log p_\theta(x_{0,\ell} \mid x_t) \right], \tag{2}$$

where $t \sim \mathrm{U}(0,1)$ controls the masking ratio and x_t is obtained by independently masking tokens. For a block of size B, the expected number of `[/MASK]` positions is tB, so larger blocks contain more masked tokens to resolve within each denoising stage. Since we scale block sizes in powers of two ($B = 2^n$), the number of tokens involved per step—and hence the degree of within-block reordering—grows exponentially with the stage index n. Moreover, standard corpora for supervised fine-tuning (SFT) only specify token targets but do not provide supervision for "correct" unmasking

schedule.

In this work, we propose **T***, an easy-to-implement yet effective strategy for progressive block-size scaling that increases block size with minimal performance degradation. **T*** offers a practical route for masked diffusion models (MDMs) to preserve the strong reasoning capability inherited from AR-initialized small-block models while moving toward higher-parallel decoding. Further analysis suggests that **T*** can induce an alternative decoding schedule, rather than reverting to the canonical left-to-right schedule.

2 Methodology

2.1 Trajectory-aware RL

We adopt TRACERL as our trajectory-aware reinforcement learning backbone, and build our method on top of it (Wang et al., 2025). TraceRL views diffusion decoding as a multi-step denoising trajectory and performs policy optimization on the same trajectory used at inference. Given a prompt Q, a diffusion LM produces a trajectory $\tau = \tau(1) \cup \cdots \cup \tau(T)$, where T is the number of denoising steps and $\tau(t)$ denotes the set of tokens decoded (unmasked) at step t. For brevity, we denote the trajectory prefix by $\tau_{<t} := \tau(1{:}t{-}1)$ and suppress the dependence on Q when it is clear.

TraceRL applies a PPO-style objective over all decoded tokens along the trajectory:

$$J(\theta) = \mathop{\mathbb{E}}_{\tau\sim\pi_{\theta_{\text{old}}}} \left[\sum_{t=1}^{T} \frac{1}{|\tau(t)|} \sum_{o\in\tau(t)} C_\epsilon(\rho_t(o),\, A(o)) \right] - \beta\, \mathrm{KL}(\pi_\theta \,\|\, \pi_{\theta_{\text{old}}}), \tag{3}$$

where $C_\epsilon(r, A) = \min\{rA,\ \mathrm{clip}(r, 1-\epsilon, 1+\epsilon)A\}$ is the clipped surrogate and

$$\rho_t(o) = \frac{\pi_\theta(o \mid \tau_{<t}, Q)}{\pi_{\theta_{\text{old}}}(o \mid \tau_{<t}, Q)}. \tag{4}$$

In the simplest verifiable-reward setting, a single sequence-level reward (e.g., correctness of the final answer) is broadcast to the trajectory and used to form the advantages in Eq. 3.

To enable finer credit assignment over denoising steps, TRACERL aggregates token-level rewards (and value predictions) into step-level quantities by averaging within each denoising step, and computes step-wise advantages via TD/GAE (Schulman et al., 2015). These step advantages are then assigned back to all tokens decoded at the corresponding step, so that learning signals propagate through the entire denoising trajectory rather than only the final output (Lightman et al., 2023b).

Algorithm 1: **T***: Progressive Block Scaling

Input: Base model θ_0, $\mathcal{D}$, Initial B_0, Target $\hat{B}$
Output: Optimized model θ

// Initialization
$\theta \leftarrow \theta_0$; $\quad B \leftarrow B_0$;
// Progressive Scaling Training
while $B \le \hat{B}$ **do**
  Sample a batch d from $\mathcal{D}$;
  // Data Preparation
  $d_1, d_2 \leftarrow \textsc{Split}(d)$;
  // First Update Step
  $\theta \leftarrow \textsc{TraceRL}(\theta,\ d_1,\ B)$;
  // Shift Mechanism
  $\Delta \leftarrow B/2$;
  $d_2' \leftarrow \textsc{Shift}(d_2,\ \Delta)$;
  // Second Update Step
  $\theta \leftarrow \textsc{TraceRL}(\theta,\ d_2',\ B)$;
  $B \leftarrow 2B$;
return θ

2.2 Progressive Block Scaling

We propose a progressive block-scaling strategy, **T***, which uses trajectory-aware RL as a catalyst to adapt the denoising policy under the current block size and then relaxes the constraint by enlarging blocks. Concretely, for each stage with block size B, we run one epoch consisting of three steps. (I) RL: on the first 50% of the training batches, we perform TRACERL updates under the current block partition. (II) SHIFTING: on the remaining 50%, we keep the same block size B but shift block boundaries by an offset $\Delta = B/2$, so that each block straddles two neighboring blocks from the original partition; this reduces boundary artifacts and helps positional representations adapt to cross-block interactions. (III) EXPANDING: we merge adjacent blocks and double the block size, $B \leftarrow 2B$, and proceed to the next stage. Algorithm 1 summarizes the training loop.

3 Experiments

3.1 Setup

We conduct experiments with the masked diffusion models **`SDAR-1.7B-Chat`** and **`SDAR-4B-Chat`**. These models are trained via block-diffusion strategy using different block sizes $B \in \{4, 8, 16, 32\}$. Our training dataset consists of 8K high-quality mathematical problems with difficult levels 3-5 from Openr1math. In each step, we randomly sample 128 problems from the

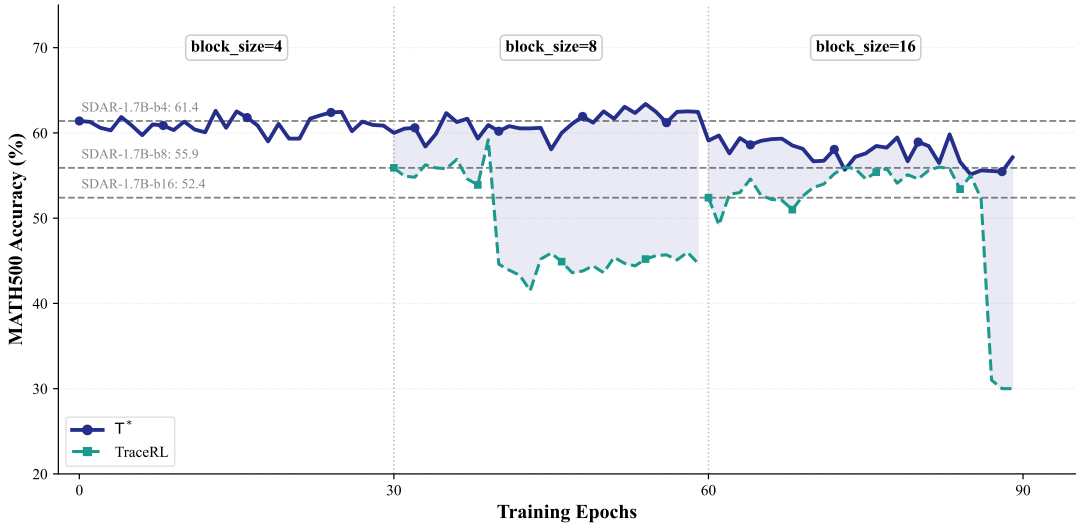

Figure 1: **Validation accuracy during block scaling (1.7B).** MATH500 validation accuracy over training epochs for $\mathbf{T}^\star$ and a direct TRACERL baseline (dashed). Vertical dotted lines indicate stage transitions ($B{=}4 \rightarrow 8 \rightarrow 16$). Horizontal dashed lines show the accuracies of the original **SDAR** checkpoints trained at each block size.

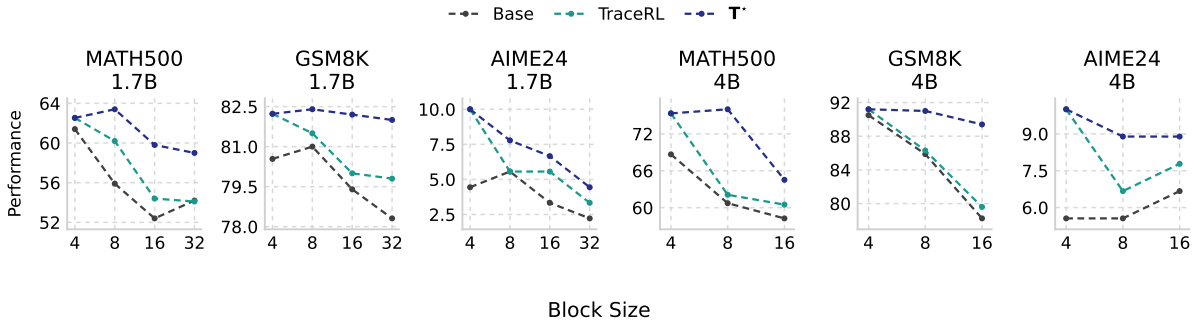

Figure 2: **Performance vs. block size across model scales.** Performance on MATH500, GSM8K, and AIME24 as a function of block size B for **SDAR** models at 1.7B (left) and 4B (right), comparing the Base model, TRACERL trained at the same B, and our progressive strategy $\mathbf{T}^\star$.

dataset and generate 16 responses per problem using the static sampling strategy. Training process is performed on an 8-GPU H200 cluster using the AdamW optimizer with a learning rate of 1×10^{-6}. To prevent the policy from collapsing or drifting far from the base model, we apply a KL-divergence penalty with $\beta = 0.01$.

Baselines: for each block size $B \in \{8, 16, 32\}$ of **SDAR**, we directly apply 30 epochs of TRACERL training.

Evaluation We evaluate the models on MATH500 (Hendrycks et al., 2021; Lightman et al., 2023a), GSM8K (Cobbe et al., 2021) and AIME24 (Art of Problem Solving, 2024a,b). During sampling, we use the same block size for inference as used in training stage and report the Pass@3 accuracy.

3.2 General Performance

Figure 1 plots MATH500 validation accuracy throughout training for the 1.7B model. While $\mathbf{T}^\star$ remains relatively stable across stages, the direct TRACERL baseline exhibits abrupt collapses: a sharp drop occurs during the $B{=}8$ stage (from ∼56% to the low-40% range), and another collapse appears near the end of the $B{=}16$ stage (down to ∼30%). We find this instability is highly sensitive to initialization at the target block size: applying TRACERL directly on the **SDAR-1.7B-Chat-b8** checkpoint collapses, whereas continuing TRACERL at $B{=}8$ starting from a TRACERL-trained $B{=}4$ diffusion policy (our stage transition) remains stable. A plausible explanation is that larger-block **SDAR** checkpoints operate under weaker conditioning contexts (cf. Eq. 2) and thus start from a lower-

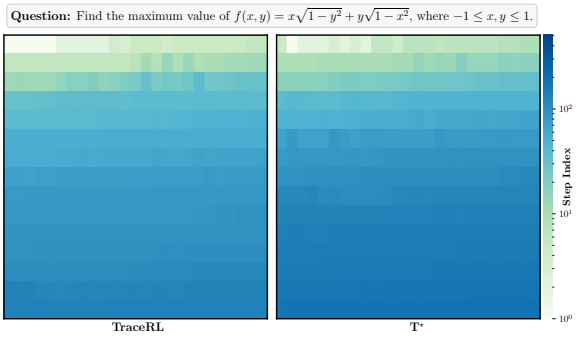

Figure 3: Decoding schedule under TRACERL vs. $\mathbf{T}^\star$. More results can be found in Appendix A.2

confidence regime, yielding noisier rollouts and higher-variance advantage estimates; when such advantages are broadcast to many tokens per denoising step, ratio-based updates can trigger likelihood drift and collapse, consistent with the Lazy Likelihood-Displacement "death spiral" analysis for GRPO-style training (Deng et al., 2025; Gao et al., 2025).

Figure 2 shows that, across different model sizes, $\mathbf{T}^\star$ consistently matches or exceeds the performance of the base models and TRACERL at the same block size on MATH500, GSM8K, and AIME24. When we expand the block size, the base models generally show a downward trend, while $\mathbf{T}^\star$ remains more stable and achieves the strongest results at most evaluated block sizes; TRACERL often improves over the base model at smaller blocks but is typically below $\mathbf{T}^\star$ at larger blocks. Unless otherwise noted, all scores are reported for the checkpoint that attains the best validation accuracy on MATH500 during training (and are then evaluated on MATH500, GSM8K, and AIME24). The exact scores can be found in Table 2 and Appendix A.1.

3.3 Schedule

We compute LOCALSTRICT (Gong et al., 2025). Let $\pi = (\pi_1, \ldots, \pi_n)$ denote the linearized unmasking order obtained by sorting token positions by their first-unmask step (ties broken by smaller positions). LOCALSTRICT is defined as the fraction of events that decode the leftmost remaining masked position:

$$\text{LOCALSTRICT} = \frac{1}{n} \sum_{k=1}^{n} \mathbb{1}\left[\pi_k = \min_{j \ge k} \pi_j\right]. \quad (5)$$

Higher values indicate a schedule closer to the canonical left-to-right order S_0, while lower values reflect more non-monotone masked updates.

Model	LocalStrict	Accuracy
`Qwen3-1.7B`	1.000	70.2
`Qwen2.5-1.5B`	1.000	55.0
`SDAR-1.7B-b32`	0.743	54.2
+ TRACERL	0.704	54.1
+ $\mathbf{T}^\star$	0.730	59.0
`SDAR-1.7B-b16`	0.766	52.4
+ TRACERL	0.824	54.4
+ $\mathbf{T}^\star$	0.804	59.8
`SDAR-1.7B-b8`	0.915	55.9
+ TRACERL	0.984	60.2
+ $\mathbf{T}^\star$	0.854	63.4

Table 1: **LocalStrict vs. accuracy on MATH500.** LocalStrict is computed by Eq. 5; higher values indicate a decoding order closer to the canonical left-to-right schedule.

Figure 3 visualizes token-level first-unmask step indices under TRACERL and $\mathbf{T}^\star$. Table 1 reports LOCALSTRICT and accuracy under different block sizes. Overall, both methods retain largely monotone unmasking behavior (i.e., LOCALSTRICT remains high), but neither collapses to a strictly deterministic left-to-right schedule; instead, the learned step-wise schedules differ under the target block size (see Appendix A.2 for more examples).

4 Conclusion

Experiments show that $\mathbf{T}^\star$ stably scales block size with minimal performance loss, providing a practical recipe for diffusion language models to inherit strong reasoning ability from AR-initialized small-block checkpoints. We analyze the collapse of direct TraceRL at larger block size models and present a potential mitigation perspective related to Lazy Likelihood Displacement. Finally, our schedule analysis suggests that trajectory-aware RL can induce a non-canonical denoising schedule under a fixed block size.

Recent work encourages non-linear reasoning via explicit external scaffolds such as tree/graph-structured search over intermediate thoughts (Yao et al., 2023; Besta et al., 2024; Yao et al., 2024). In contrast, our experiments show that trajectory-aware RL can modify the model's internal denoising policy (i.e., token-finalization schedule) and improve reasoning performance without introducing an external search procedure, suggesting internal schedule learning as a complementary direction.

Limitations

The limitations of this work can be summarized as:

- $\mathbf{T}^{\star}$ mitigates but does not fully eliminate degradation under block expansion; we suspect residual drops are partly due to the lack of a high-quality "cold-start" stage.
- We did not scale to very large blocks (e.g., B=64 or 128) in our $\mathbf{T}^{\star}$ curriculum, because the inference engine becomes unstable at large block sizes.

Ethical Statements

In this paper, we propose strategies to improve the SQL generation capabilities of LLMs. To the best of our knowledge, we do not expect our system would have negative impacts on society.

References

Marianne Arriola, Aaron Gokaslan, Justin T. Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. 2025. Block diffusion: Interpolating between autoregressive and diffusion language models. In *International Conference on Learning Representations*.

Art of Problem Solving. 2024a. 2024 AIME i. AoPS Wiki. Accessed: 2026-01-05.

Art of Problem Solving. 2024b. 2024 AIME ii. AoPS Wiki. Accessed: 2026-01-05.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, Vancouver, Canada*, pages 17682–17690.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Shuang Cheng, Yihan Bian, Dawei Liu, Yuhua Jiang, Yihao Liu, Linfeng Zhang, Wenhai Wang, Qipeng Guo, Kai Chen, Biqing Qi, and Bowen Zhou. 2025. SDAR: A synergistic diffusion–autoregression paradigm for scalable sequence generation.

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

Wenlong Deng, Yushu Li, Boying Gong, Yi Ren, Christos Thrampoulidis, and Xiaoxiao Li. 2025. On grpo collapse in search-r1: The lazy likelihood-displacement death spiral. *arXiv preprint arXiv:2512.04220*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Chang Gao, Chujie Zheng, Xiong-Hui Chen, Kai Dang, Shixuan Liu, Bowen Yu, An Yang, Shuai Bai, Jingren Zhou, and Junyang Lin. 2025. Soft adaptive policy optimization. *arXiv preprint arXiv:2511.20347*.

Shansan Gong, Ruixiang Zhang, Huangjie Zheng, Jiatao Gu, Navdeep Jaitly, Lingpeng Kong, and Yizhe Zhang. 2025. Diffucoder: Understanding and improving masked diffusion models for code generation.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset. In *Advances in Neural Information Processing Systems*.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023a. Let's verify step by step. *arXiv preprint arXiv:2305.20050*. Defines the nonstandard MATH-500 evaluation split released in the PRM800K repository.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023b. LetâĂŹs verify step by step.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach.

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T. Chiu, Alexander Rush, and Volodymyr Kuleshov. 2024. Simple and effective masked diffusion language models. In *Advances in Neural Information Processing Systems*.

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. 2015. High-dimensional continuous control using generalized advantage estimation.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and efficient foundation language models.

Yinjie Wang, Ling Yang, Bowen Li, Ye Tian, Ke Shen, and Mengdi Wang. 2025. Revolutionizing reinforcement learning framework for diffusion large language models.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*.

Yao Yao, Zuchao Li, and Hai Zhao. 2024. GoT: Effective graph-of-thought reasoning in language models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 2901–2921, Mexico City, Mexico. Association for Computational Linguistics.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. 2025. Dream 7b: Diffusion large language models.

A Appendix

A.1 Full Results

Base model	Method	MATH500 ↑	GSM8K ↑	AIME24 ↑
`SDAR-1.7B-Chat-b4`	–	61.40	80.54	4.44
	TRACERL	62.53	82.23	10.00
`SDAR-1.7B-Chat-b8`	–	55.90	81.00	5.56
	TRACERL	60.20	81.50	5.56
	$\mathbf{T}^\star$	63.40	82.40	7.78
`SDAR-1.7B-Chat-b16`	–	52.40	79.40	3.33
	TRACERL	54.40	80.00	6.66
	$\mathbf{T}^\star$	59.80	82.20	6.66
`SDAR-1.7B-Chat-b32`	–	54.20	78.31	2.22
	TRACERL	54.10	79.80	3.33
	$\mathbf{T}^\star$	59.00	82.00	4.44
`SDAR-4B-Chat-b4`	–	68.67	90.50	5.56
	TRACERL	75.33	91.20	10.00
`SDAR-4B-Chat-b8`	–	60.73	85.87	5.56
	TRACERL	62.10	86.30	6.67
	$\mathbf{T}^\star$	76.00	91.00	8.89
`SDAR-4B-Chat-b16`	–	58.26	78.24	6.67
	TRACERL	60.50	79.60	7.78
	$\mathbf{T}^\star$	64.53	89.40	8.89

Table 2: **Reasoning performance under different block sizes.** "–" denotes the original `SDAR-·-Chat-b`B checkpoint, TRACERL applies trajectory-aware RL at the same block size B, and $\mathbf{T}^\star$ denotes our progressive block-size scaling.

A.2 Case Study

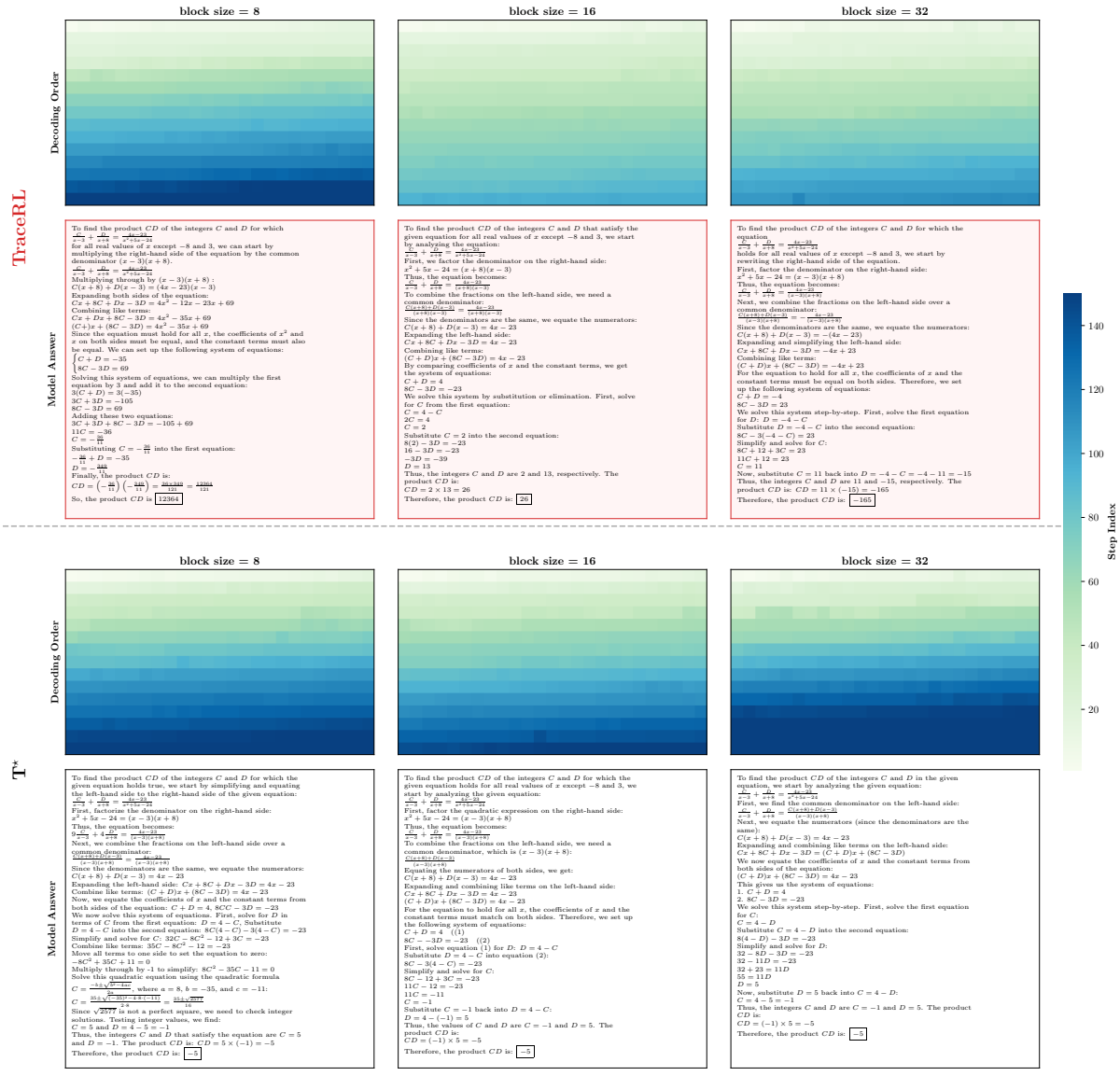

Figure 4: **Case study: decoding schedule under TRACERL vs. T⋆.** We visualize the token-level first-unmask step index (heatmaps; darker means decoded later) and the corresponding model solutions for a representative algebra problem, evaluated with block sizes $B \in \{8, 16, 32\}$. The top row shows a model trained with direct TRACERL at the same block size, and the bottom row shows the model obtained by **T⋆**.