TERSE: TRACE-EFFICIENCY REWARD VIA SQUEEZING REDUNDANCY FOR EFFICIENT LARGE REASONING MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large reasoning models perform well on complex problems by generating long chains of thought, but these traces are often redundant, raising inference cost and latency without improving accuracy. Reinforcement learning methods curb this by penalizing length, which captures how much a model writes but not how much of it merely repeats: two correct traces of equal length can carry very different amounts of redundancy. Prior methods address this by adding a second signal, derived from external information such as a reflection lexicon, an auxiliary model, or the model's internals. We instead read a second signal from the trace itself, the gzip compressibility of its raw bytes, which depends on bytes rather than meaning and is therefore not tied to a domain or model family. Building on it, we propose Trace-Efficiency Reward via Squeezing rEdundancy (TERSE), a reward for group-relative reinforcement learning. TERSE residualizes compressibility against length so that it rewards only the redundancy length does not already explain, and standardizes both signals within difficulty buckets to remain stable on hard prompts where correct rollouts are scarce. On five mathematical reasoning benchmarks with DeepSeek-R1-Distill-Qwen at 1.5B and 7B, TERSE improves accuracy over the base model by 7.8 and 3.4 points while cutting response length by 54.4% and 37.4%, improving the accuracy–length trade-off at both scales.

1 INTRODUCTION

Recently, Large Reasoning Models (LRMs) such as OpenAI o1 (OpenAI, 2024), QwQ (Qwen Team, 2025), and DeepSeek-R1 (DeepSeek-AI, 2025) have drawn intense research attention for their strong performance on complex tasks, scaling test-time computation into long chains that self-assess and self-correct (Wei et al., 2022; Shao et al., 2024). These chains, however, often overthink (Chen et al., 2024; Sui et al., 2025; Yue et al., 2025): models spend far more tokens than a problem requires, inflating cost and latency without a matching gain in accuracy. The inefficiency lies not in reasoning itself but in how a trace is written. The same answer can be reached by a concise derivation or by a repetitive one that revisits steps it has already settled. This makes efficient reasoning challenging, because trimming tokens too aggressively can also remove the steps that reach the answer. A natural remedy is to reward shorter traces, and most methods take this route.

Length, however, is an imperfect proxy for redundancy. It counts how much a model writes, not how much of that writing repeats, so two traces of equal length receive the same penalty even when one is far more repetitive. Existing work refines the penalty in two ways. One line keeps length as the only signal and scales its strength by problem difficulty, estimated from the online solve rate or from group statistics, so that hard problems are penalized less (Peng et al., 2026; Xiang et al., 2025). The other adds a second signal that measures redundancy more directly, drawn from a reflection lexicon (Yu et al., 2026), an auxiliary model (Hu et al., 2026), the model's internal states (Cai et al., 2026; Liu et al., 2025a), or the reference answer (Cheng et al., 2025). These signals track redundancy more faithfully than length alone, but each pays for it with a dependency outside the trace, such as a hand-built lexicon or a separate model maintained alongside the policy (Table 1). What is missing is a signal that reads redundancy from the trace itself, with no such dependency.

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.

Table 1: What the redundancy signal of each method depends on to be computed (✓ = depends on it). The first four rows list resources outside the trace, and the last indicates whether the signal needs an additional forward pass beyond generating the trace. All five prior methods depend on at least one such resource, whereas TERSE depends on none and is computed from the raw bytes of the trace.

Dependency	ARLCP	LC-R1	ENTRA	BINGO	ConMax	**TERSE**
Hand-crafted lexicon	✓					
Auxiliary model		✓			✓	
Model internals			✓	✓		
Reference answer		✓			✓	
Extra inference		✓	✓	✓	✓	

We measure this redundancy from the trace with a general-purpose compressor such as gzip (Ziv & Lempel, 1977), which encodes a repeated substring as a short reference to its earlier occurrence. A trace that loops back over content it has already covered then compresses into far fewer bytes than one that keeps adding new content. We take the compressibility c of a trace, one minus its gzip compression ratio, as a proxy for how much of it is redundant (Delétang et al., 2024; Jiang et al., 2023). Because c reads bytes rather than meaning, it is content-agnostic and depends on none of the external resources the second signals above require (Table 1), so it is not tied to a particular domain or model family. When a redundant trace is also longer, c flags it just as length does. The harder case is two correct traces of similar length: length scores them alike, but the more repetitive one is more compressible, and c marks it as the more redundant of the two (Figure 1). In this case c supplies information that a length penalty alone cannot.

Building on this signal, we propose Trace-Efficiency Reward via Squeezing rEdundancy (TERSE), a reward-shaping method for group-relative RL that scores reasoning efficiency from two complementary signals, the length of a trace and its compressibility c. TERSE has three components. First, we add compressibility to the reward as a second efficiency signal alongside the usual length term, so the reward can penalize repetition that a length term alone would miss. Second, a more compressible trace tends to be longer, so adding both signals directly would let length act through the compressibility term and be penalized twice. We therefore residualize compressibility against length and keep only the residual, so the reward charges just the redundancy that length does not already explain. Third, both signals must be standardized before they are combined, but a single group can be too small for this, since a hard prompt may yield only one or two correct rollouts. We therefore standardize within difficulty buckets, pooling correct rollouts across prompts of similar difficulty and tracking the bucket means and variances as exponential moving averages across steps.

We evaluate TERSE on five math-reasoning benchmarks, ranging from grade-school GSM8K to competition-level AIME, with DeepSeek-R1-Distill-Qwen at 1.5B and 7B. Relative to the base model, TERSE improves the accuracy–efficiency trade-off at both scales, cutting average response length by 54.4% and 37.4% while raising accuracy by 7.8 and 3.4 points. Our contributions are:

- **We identify gzip compressibility as a content-agnostic redundancy signal.** Computed from the decoded bytes of a trace alone, it captures redundancy that a length penalty misses, without depending on any resource outside the trace (Section 3.2).
- **We turn this signal into a reward, TERSE.** It adds compressibility to a length term but keeps only the part that length does not already explain, and standardizes both within difficulty buckets so the reward stays reliable on hard prompts (Section 3.2).
- **We show the gains hold beyond the training setting.** TERSE exceeds strong length-penalty baselines under a common protocol, and its gains carry to a different model family and an out-of-domain benchmark (Section 4).

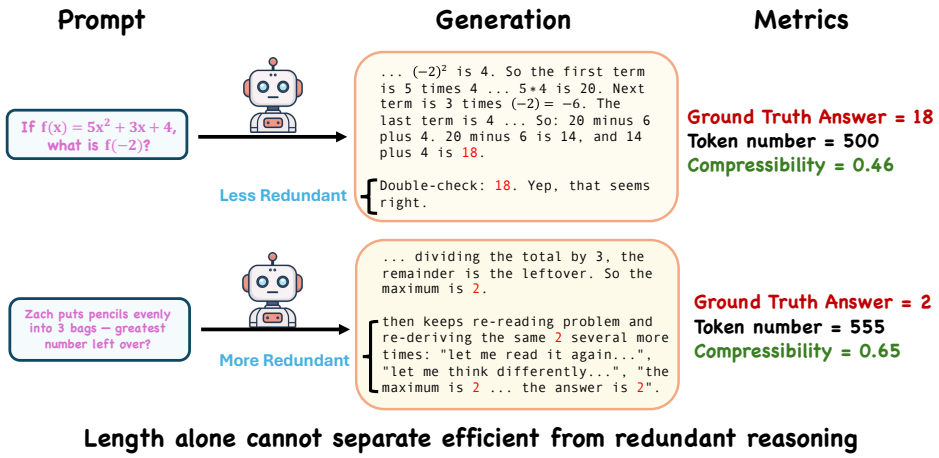


Figure 1: Two correct rollouts of DeepSeek-R1-Distill-Qwen-1.5B on different prompts, with verbatim excerpts. Both reach the right answer at nearly the same length (500 and 555 tokens), but differ in how much they repeat: the first adds only a short verification, while the second loops over the same answer, re-reading the problem and re-deriving it several times in near-identical wording. A length penalty scores the two almost equally, yet the repeated phrasing in the second compresses into far fewer bytes, giving it a higher compressibility (0.65 against 0.46, higher meaning more redundant).

2 Related Work

Efficient reasoning. Work on curbing redundant reasoning falls into three lines. Truncation-based methods cap the generation budget or prune traces to a fixed length (Muennighoff et al., 2025; Han et al., 2024; Hou et al., 2025), which reliably cuts computation but risks discarding necessary intermediate steps. Supervised fine-tuning on shortened or difficulty-pruned traces (Xia et al., 2025; Qiao et al., 2025; Waheed et al., 2025) distills brevity into a student model, but depends on curated compression data and can degrade reasoning quality when the removed steps were load-bearing. Reinforcement learning instead builds efficiency into the reward, trading length against accuracy during training rather than fixing a budget in advance, and it is the setting we adopt. Most RL methods penalize length against a target that is adapted to problem difficulty: ARLCP (Yu et al., 2026) couples a reflection-token penalty with a length penalty, ALP (Xiang et al., 2025) scales a per-token cost by each prompt's online solve rate, Arora & Zanette (2025) tie the penalty to task difficulty under a single knob, and LASER (Liu et al., 2025b) rewards correct traces shorter than an adapted target, with group-relative variants estimating that target from batch statistics (Peng et al., 2026). Others reshape the reward rather than penalize length: CEEH (Luo et al., 2026) gives hard problems more slack through selective entropy regularization, Thinkless (Fang et al., 2025) learns to switch between short and long reasoning by task, and LC-R1 (Cheng et al., 2025) trains an extractor to cut the trace once the answer is reached. A separate line targets redundancy more directly within the reward, penalizing internal and external repetition (Hong et al., 2025), rewarding structurally concise reasoning through process-level anchors (Chang et al., 2026), regularizing toward shorter internal distributions (Wang et al., 2026), or adapting compression from past rollouts (Bian et al., 2026). TERSE instead reads its second signal from the raw bytes of the trace with a parameter-free compressor, requiring no auxiliary model, embedding, or reference answer.

Compression and information-theoretic signals. How much a sequence compresses reflects how much of it is repetition, since a compressor represents recurring content compactly. A reasoning trace that revisits steps it has already settled shrinks far more under compression than one that keeps making new progress. This intuition has a formal counterpart for language models, which are themselves compressors: the code length a model assigns to a sequence equals its negative log-likelihood (Delétang et al., 2024). A compressor need not even be learned to be useful on text: gzip alone drives a text classifier competitive with trained models (Jiang et al., 2023). This is what lets us read redundancy from a trace with an off-the-shelf compressor rather than a model of our own.

In efficient reasoning, the only use of compression we know of operates at decoding time, where LZ-Penalty (Anonymous, 2025) subtracts an LZ code-length term from the logits and suppresses repetition token by token as a trace is generated. We instead make compressibility part of the reward for group-relative RL, so the policy is trained to produce reasoning with less of the repetition that compression exposes.

3 METHODOLOGY

In this section we present Trace-Efficiency Reward via Squeezing rEdundancy (TERSE). A length penalty rewards brevity but cannot see how much of a trace merely repeats. TERSE adds compressibility as a second signal and, within a fixed budget, weights length on easy prompts and compressibility on hard ones. Section 3.1 reviews group-relative RL and the existing length-penalty reward. Section 3.2 then develops TERSE in three parts: the compressibility signal, its residualization against length, and a difficulty-bucketed standardization that stabilizes both signals.

3.1 PRELIMINARIES

Reasoning with large reasoning models. We consider an LLM with parameters θ, denoted π_θ. Given a prompt $q = (q_1, \ldots, q_m)$, the model generates a reasoning trace $\tau = (\tau_1, \ldots, \tau_T)$ autoregressively from the conditional distribution $\pi_\theta(\cdot \mid q)$, which can be decomposed as

$$\pi_\theta(\tau \mid q) = \prod_{t=1}^{T} \pi_\theta(\tau_t \mid q,\, \tau_{<t})\,, \tag{1}$$

where $\tau_{<t}$ denotes the tokens generated before position t and T is the trace length. For a large reasoning model (LRM) (DeepSeek-AI, 2025), it generates the reasoning trace between $\langle$`think`$\rangle$ and $\langle$`/think`$\rangle$ markers, followed by a conclusion p, so its output takes the form

$$q \;\; \langle\texttt{think}\rangle \;\; \tau_1, \ldots, \tau_T \;\; \langle\texttt{/think}\rangle \;\; p. \tag{2}$$

The final prediction is $y = f(p)$, where f extracts the answer from the conclusion, for instance the value inside `\boxed{}`. This final prediction can be verified as correct or not.

Efficient reasoning through length penalties. Longer reasoning improves the accuracy of LRMs, but it raises inference latency, token consumption, and serving cost, which motivates producing shorter traces without losing accuracy. A common approach optimizes the LRM with group-relative RL (Shao et al., 2024; Ahmadian et al., 2024) under an added length-penalty reward. For a prompt q, the policy generates a chain-of-thought trace τ and a conclusion p, from which an extraction function f reads the final answer $f(p)$. We write p^* for the ground-truth answer. Given G independent rollouts $\{\tau^g\}_{g=1}^{G}$, the empirical pass rate is

$$pr(q) = \frac{1}{G} \sum_{g=1}^{G} \mathbf{1}[f(p^g) = f(p^*)]\,, \tag{3}$$

which serves as an online difficulty estimate: a high $pr(q)$ marks an easy prompt and a low one a hard prompt. The length penalty is scaled by this estimate, so easy prompts are penalized more for spending tokens while hard prompts are penalized little, leaving room for the longer reasoning they require. With ℓ the length of τ, a normalization constant N (e.g., the maximum trace length), and a global coefficient $w > 0$, a representative reward is

$$r(q, \tau) = \underbrace{\mathbf{1}[f(p) = f(p^*)]}_{r_{\text{accuracy}}} \;-\; \underbrace{w \, \max\!\big(pr(q),\, G^{-1}\big) \, \frac{\ell}{N}}_{r_{\text{length}}}. \tag{4}$$

Here r_{accuracy} rewards a correct answer, and r_{length} charges a per-token cost weighted by w and by the clipped pass rate, the clip at G^{-1} ensuring that even unsolved prompts incur some penalty. As is standard, the length penalty is applied only to correct rollouts, since penalizing length on incorrect ones distorts the group baseline (Peng et al., 2026).

3.2 TRACE-EFFICIENCY REWARD VIA SQUEEZING REDUNDANCY

Motivation. The reward in Eq. 4 reaches redundancy only through length, which counts how much a model writes but not how much of that writing repeats: two correct traces of equal length can carry very different amounts of redundancy, yet the length term scores them alike. Recent methods add a second signal that measures redundancy more directly, reading it from a reflection lexicon (Yu et al., 2026), an auxiliary model (Hu et al., 2026), model internals (Cai et al., 2026), or the position of the answer in the trace (Cheng et al., 2025). We take a different signal, the gzip compressibility of a trace, which captures repetition from its raw bytes and depends on none of these resources. A second signal, however, cannot simply be added alongside length. The two are not independent: repetition contributes to length, so the signals partially overlap, and summing them charges the shared component twice, which can over-penalize length and suppress reasoning that harder problems need (Xiang et al., 2025; Peng et al., 2026; Liu et al., 2025b). We instead reward only the redundancy that length does not already explain. A further issue arises on hard prompts, where a group yields too few correct rollouts to estimate a stable within-group statistic for either signal. We address this by pooling correct rollouts across prompts of similar difficulty and across training steps. We now develop these three pieces: the compressibility signal, the residualization that removes its overlap with length, and the difficulty-bucketed standardization that supports both.

Compressibility. We quantify this redundancy with a compressor such as gzip (Ziv & Lempel, 1977) or LZMA (Ziv & Lempel, 1978), which encodes repeated content in far fewer bytes while leaving genuinely new reasoning nearly untouched. This is the intuition behind Kolmogorov complexity, the ideal information content of a sequence: that quantity is uncomputable, but any lossless compressor gives a computable upper bound on it (Li & Vitányi, 2019). Let $o = (\tau, p)$ denote the full generated sequence, the reasoning trace τ together with the conclusion p. Writing $|\cdot|$ for length in bytes, the *compressibility* of o is

$$c(o) = \max\left(0,\ 1 - \frac{|\mathrm{gzip}(o)|}{|o|}\right) \in [0, 1], \tag{5}$$

where the outer $\max$ guards against gzip's header overhead on very short sequences. A higher $c(o)$ marks a more compressible, and thus more redundant, sequence. Computing c reads only the bytes of o: it calls no model, needs no reflection lexicon, and locates no answer span, so it is not tied to a domain or model family.

Residualization. Compressibility and length are correlated but not identical: repetition inflates a sequence, so the two partly move together. Before combining them, we standardize each over the correct rollouts,

$$\tilde{\ell}^g = \frac{\ell^g - \mu_\ell}{\sigma_\ell}, \qquad \tilde{c}^g = \frac{c^g - \mu_c}{\sigma_c}, \tag{6}$$

where $\ell^g = |o^g|$ is the length of a correct rollout's output o^g and $c^g = c(o^g)$ its compressibility, and (μ, σ) are the mean and standard deviation over correct rollouts, pooled as described next. The standardized signals still share part of their variation, so adding both to the reward would let length act through both terms. We regress $\tilde{c}$ on $\tilde{\ell}$ and take the residual, which keeps only the part of compressibility that length does not explain

$$e^g = \tilde{c}^g - \rho\,\tilde{\ell}^g, \qquad \rho = \frac{\mathrm{Cov}(\tilde{\ell},\, \tilde{c})}{\mathrm{Var}(\tilde{\ell})}, \tag{7}$$

where the slope ρ is clipped to $[0, 1]$: length and compressibility are positively related in practice, and subtracting a negative slope would add length variation back rather than remove the shared part. Like the standardized signals, the residual is clipped to $[-3, 3]$. It isolates the component of compressibility that length does not account for, so the compressibility term contributes beyond the length penalty rather than restating it.

Difficulty-bucketed standardization. The standardization uses, within each bucket, a mean and standard deviation for each signal, (μ_ℓ, σ_ℓ) for length and (μ_c, σ_c) for compressibility, together with a single slope ρ that regresses compressibility on length. These statistics are computed over correct rollouts, but one prompt seldom has enough of them: a hard prompt may produce only one or two

Algorithm 1 Trace-Efficiency Reward via Squeezing rEdundancy (TERSE)

Input: batch $\mathcal{D}_b$; policy π; group size G; budget W; EMA decay β; bucket statistics $\{\mu_\ell^{(b)}, \sigma_\ell^{(b)}, \mu_c^{(b)}, \sigma_c^{(b)}, \rho^{(b)}\}$
for each prompt $q \in \mathcal{D}_b$ with rollouts $\{o^g\}_{g=1}^G \sim \pi(\cdot \mid q)$ **do**
  $\ell^g \leftarrow |o^g|$ and $c^g \leftarrow c(o^g)$ for each g
  $pr(q) \leftarrow \frac{1}{G}\sum_g \mathbf{1}[f(p^g) = f(p^*)]$ and $d \leftarrow 1 - pr(q)$
end for
Split prompts into easy/medium/hard buckets by the tertiles of pr
for each correct rollout o^g in bucket b **do**
  $\tilde{\ell}^g \leftarrow (\ell^g - \mu_\ell^{(b)})/\sigma_\ell^{(b)}$ and $\tilde{c}^g \leftarrow (c^g - \mu_c^{(b)})/\sigma_c^{(b)}$
  $e^g \leftarrow \tilde{c}^g - \rho^{(b)}\,\tilde{\ell}^g$
  $r^g \leftarrow 1 - W(1-d)\,\tilde{\ell}^g - W d\, e^g$
end for
Update π from $\{r^g\}$ with a group-relative estimator
Update $\{\mu_\ell^{(b)}, \sigma_\ell^{(b)}, \mu_c^{(b)}, \sigma_c^{(b)}, \rho^{(b)}\}$ by EMA (Eq. 8)
Output: updated policy π

correct traces in a group, too few for a meaningful mean or variance. We therefore estimate them per difficulty level. Each step, the batch's prompts are ranked by pass rate $pr(q)$ and split at its tertiles into three buckets, with the highest third by pass rate labeled easy, the lowest third hard, and the rest medium. The bucket statistics are then computed within each bucket b from its correct rollouts. Because difficulty is relative to the current batch, every bucket is populated, and pooling many prompts into a bucket gives more correct traces than any single prompt provides. To keep the statistics stable across the shifting policy, each is tracked as an exponential moving average,

$$s^{(b)} \leftarrow (1-\beta)\, s^{(b)} + \beta\, \hat{s}^{(b)}, \qquad s^{(b)} \in \{\mu_\ell^{(b)}, \sigma_\ell^{(b)}, \mu_c^{(b)}, \sigma_c^{(b)}, \rho^{(b)}\},\ \beta = 0.1, \tag{8}$$

where $\hat{s}^{(b)}$ is the current step's estimate. A step scores its rollouts with the pre-update values before applying Eq. 8. Standardized values are clipped to $[-3, 3]$. These statistics only place the two signals on a comparable scale and leave the advantage unchanged.

The TERSE reward. TERSE shapes only correct rollouts. For a rollout o^g on a prompt of difficulty $d = 1 - pr(q)$,

$$r^g = \begin{cases} 1 - w_\ell(d)\,\tilde{\ell}^g - w_e(d)\, e^g, & o^g \text{ correct}, \\ 0, & o^g \text{ incorrect}, \end{cases} \tag{9}$$

with the two weights sharing a single budget $W \in (0, 1)$,

$$w_\ell(d) = W\,(1-d), \qquad w_e(d) = W\, d. \tag{10}$$

The weights sum to W at every difficulty, so the budget is reallocated between the two signals rather than accumulated: (w_ℓ, w_e) runs from $(W, 0)$ on the easiest prompts to $(0, W)$ on the hardest. On easy prompts the budget goes to length, which already captures most of their waste. On hard prompts it shifts to residual compressibility. This relaxes the length penalty where longer reasoning is often necessary, while still penalizing repetition. Difficulty enters only through the pass rate, and the schedule is fixed by the single coefficient W, without a token vocabulary or any learned component. Algorithm 1 summarizes TERSE within a group-relative RL step.

4 Experiments

4.1 Setup

Dataset and models. We start from DeepSeek-R1-Distill-Qwen at 1.5B and 7B (DeepSeek-AI, 2025), two open reasoning models widely used as backbones in the efficient-reasoning literature, and train on DeepScaleR (Luo et al., 2025b), a set of $\sim$40K math problem–answer pairs disjoint from our evaluation benchmarks.

Table 2: Accuracy (Acc, %) and response length (Len, tokens) on five math benchmarks of increasing difficulty, for DeepSeek-R1-Distill-Qwen base models (DeepSeek-AI, 2025) at 1.5B and 7B. ΔAcc and ΔLen% report the average change in accuracy (in points) and response length (as a percentage) relative to the vanilla base model across the five benchmarks, where a negative ΔLen% denotes shorter responses. Baseline numbers are taken from Yu et al. (2026), and TERSE is evaluated on the same base models and benchmarks. Best accuracy per benchmark is in **bold**.

		GSM8K		MATH-500		AMC 2023		AIME 2024		AIME 2025		Overall (vs. Vanilla)	
	Method	Acc↑	Len↓	Acc↑	Len↓	Acc↑	Len↓	Acc↑	Len↓	Acc↑	Len↓	ΔAcc↑	ΔLen%
1.5B	Vanilla	78.5	1009	80.2	4582	66.7	7742	30.0	12256	21.4	12201	–	–
	NoThinking	70.0	278	69.2	908	49.2	1377	14.4	4266	9.8	3630	−12.8	−81.0
	$SFT_{shortest}$	79.4	928	81.0	4531	67.7	7657	26.0	12181	21.5	12258	−0.3	−2.1
	$DPO_{shortest}$	77.1	989	84.2	4185	69.5	7091	29.6	11721	23.1	11879	1.4	−5.2
	O1-Pruner	78.4	913	82.6	4375	70.5	7046	27.7	11937	20.8	11855	0.6	−5.7
	TLMRE	84.3	530	82.1	1800	72.1	2798	25.8	5386	19.6	4581	1.4	−58.1
	AdaptThink	84.2	488	83.2	1869	67.2	3342	30.8	6864	22.5	6626	2.2	−51.5
	LASER	82.3	850	84.6	2365	75.9	3899	28.8	7629	25.4	7084	4.0	−38.7
	ARLCP	**87.3**	658	84.6	1812	73.3	3037	34.2	5951	**26.5**	5196	5.8	−53.1
	TERSE	86.8	644	**85.6**	1675	**80.9**	2775	**36.9**	5929	25.6	5305	**7.8**	−54.4
7B	Vanilla	87.3	694	91.2	3673	87.5	5980	51.5	10547	35.0	11427	–	–
	NoThinking	84.7	284	80.6	706	61.3	1179	24.0	2671	16.3	2499	−17.2	−74.6
	$SFT_{shortest}$	86.4	606	89.0	3575	85.8	6005	49.2	10575	39.0	11384	−0.7	−3.1
	$DPO_{shortest}$	87.9	624	90.8	3340	88.9	5295	50.4	9899	**39.6**	10692	1.0	−8.6
	O1-Pruner	86.8	625	91.2	3422	87.7	5509	51.5	10034	38.3	10793	0.5	−7.0
	TLMRE	88.8	866	91.2	2419	88.1	3560	52.1	6393	35.6	6584	0.6	−26.3
	AdaptThink	90.3	330	91.0	1862	86.9	3641	56.3	8784	37.7	9622	1.9	−34.7
	LASER	86.7	818	91.0	1871	89.5	2894	56.5	5995	39.4	6410	2.0	−34.0
	ARLCP	89.3	578	91.0	2086	89.7	3260	**56.7**	6795	**39.6**	7560	2.7	−35.0
	TERSE	**91.4**	575	**92.6**	1945	**91.6**	3110	54.4	6478	**39.6**	7297	**3.4**	−37.4

Baselines. We compare TERSE against the vanilla base model and a representative set of efficient-reasoning methods: NoThinking (Ma et al., 2025), the shortest-correct-trace fine-tuning variants $SFT_{shortest}$ and $DPO_{shortest}$, O1-Pruner (Luo et al., 2025a), TLMRE (Yeo et al., 2025), AdaptThink (Zhang et al., 2025), LASER (Liu et al., 2025b), and ARLCP (Yu et al., 2026). We report their numbers under a common evaluation protocol following ARLCP (Yu et al., 2026), on the same base models and benchmarks, so that accuracy and length differences reflect the method rather than the setup.

Evaluation. We evaluate on five math benchmarks of increasing difficulty: GSM8K (Cobbe et al., 2021), MATH-500 (Hendrycks et al., 2021), AMC 2023, AIME 2024, and AIME 2025, reporting pass@1 accuracy (Acc) and mean response length in tokens (Length). To summarize the accuracy–length trade-off, we also report ΔAcc and ΔLength%, the average change in accuracy (in points) and in response length (as a percentage) relative to the vanilla base model, averaged across the five benchmarks. A negative ΔLength% denotes shorter responses. Because AMC 2023, AIME 2024, and AIME 2025 are small (30–40 problems each), we sample 16 responses per problem and report the average. For all models we use a 16K evaluation context and temperature 0.6, the default for the DeepSeek-R1-Distill models.

Implementation details. We build on the veRL framework (Sheng et al., 2024) and optimize with REINFORCE Leave-One-Out (RLOO) (Ahmadian et al., 2024). While GRPO (Shao et al., 2024) is the more common choice for reasoning RL, prior work reports that it can become unstable once a length penalty enters the objective, sometimes collapsing the policy, whereas RLOO stays robust under length-penalized rewards (Yu et al., 2026). Each training step samples 16 rollouts per prompt over batches of 128 prompts, with a learning rate of 2×10^{-6} and a 16K-token generation budget, on 8 H100 GPUs. TERSE adds a single efficiency coefficient to the reward, fixed to $W = 0.15$ across all runs, with EMA decay $\beta = 0.1$ and gzip as the compressor.

4.2 Main results

Table 2 reports accuracy and response length for TERSE and all baselines across the five benchmarks at both scales. TERSE obtains the best averaged accuracy gain and the best averaged length reduction of any method at both model sizes, improving accuracy over the vanilla base model by 7.8 points while cutting length by 54.4% at 1.5B, and by 3.4 points and 37.4% at 7B. The best-

Table 3: Ablation of TERSE on DeepSeek-R1-Distill-Qwen-1.5B. Each variant removes or replaces one part of the reward. ΔAcc and ΔLen% are measured against the vanilla base model.

	GSM8K		MATH-500		AMC 2023		AIME 2024		AIME 2025		Overall	
Variant	Acc ↑	Len ↓	Acc ↑	Len ↓	Acc ↑	Len ↓	Acc ↑	Len ↓	Acc ↑	Len ↓	ΔAcc ↑	ΔLen%
Vanilla base	78.5	1009	80.2	4582	66.7	7742	30.0	12256	21.4	12201	–	–
TERSE	86.8	644	85.6	1675	80.9	2775	36.9	5929	25.6	5305	7.8	−54.4
Reward design												
No length signal	85.1	955	85.8	2683	76.4	4497	35.0	8940	25.6	8519	6.2	−29.2
No redundancy signal	85.2	694	84.6	1844	76.1	3283	34.2	6693	25.0	6133	5.7	−48.7
No residualization	85.9	527	84.2	1539	79.5	2629	36.9	5284	24.4	5178	6.8	−58.9
No difficulty bucketing	87.3	654	85.6	1837	77.7	3144	34.0	6346	25.4	6150	6.6	−50.5
Redundancy measure												
LZMA	86.0	549	85.2	1578	76.7	2863	39.0	5761	27.1	5279	7.4	−56.8
4-gram repetition	86.5	698	86.0	1833	77.8	3006	32.1	6121	28.7	5516	6.9	−51.4

performing baseline, ARLCP, reaches only 5.8 and 2.7 points at 53.1% and 35.0% shorter length. This lead is not an averaging artifact. At 1.5B TERSE is both more accurate and shorter than ARLCP on three of the five benchmarks, most clearly on AMC 2023, where it scores 80.9% against 73.3% in 2775 tokens, 14.2 points above the base model at about a third of its length. At 7B it holds the single best accuracy on GSM8K, MATH-500, and AMC 2023, solving GSM8K at 91.4% in 575 tokens against the base model's 87.3% in 694. On the remaining benchmarks TERSE remains close to the leading method while keeping its responses comparably short. It is the only method whose accuracy never falls below the vanilla base model on any benchmark at either scale, while every baseline drops below it on at least one.

Baselines face a recurring tension between compression and accuracy. NoThinking cuts length by 74.1% but loses 12.8 points of accuracy, while SFT, DPO, and O1-Pruner preserve accuracy but reduce length by only 2 to 6 percent. Even methods that compress by more than half still gain little accuracy: TLMRE and AdaptThink cut length by 58.1% and 51.5% yet improve accuracy by only 1.4 and 2.2 points. TERSE reaches a comparable 54.4% reduction while improving accuracy by 7.8 points, three to five times the gain of these methods at similar compression. TERSE thus obtains a compression comparable to the strongest baselines while achieving a larger accuracy gain, showing that its efficiency does not come at the expense of accuracy.

4.3 Ablation Study

Role of each reward term. We ablate the reward in Eq. 9 on the DeepSeek-R1-Distill-Qwen 1.5B model, removing one component at a time in Table 3. The length and redundancy terms prove complementary. Keeping only the length term still shortens responses by 48.7% but raises accuracy by just 5.7 points, the smallest gain of any variant, whereas keeping only redundancy holds accuracy but shortens responses by only 29.2%, the weakest compression of any variant. Length drives the shortening, and redundancy keeps that shortening from eroding accuracy. Residualization is the least obvious of the three. Removing it, so that redundancy is scored by the raw standardized compressibility, shortens responses even more than the full reward, by 58.9%, yet accuracy rises by only 6.8 points. Because compressibility and length partly overlap, the raw signal reintroduces the length component that the length term already accounts for, so length is effectively weighted twice and the reward over-shortens at a cost in accuracy. Residualization subtracts the part of compressibility that varies with length, so the compressibility term contributes mainly the redundancy that length does not already explain. Removing difficulty bucketing drops accuracy by about three points on AMC 2023 and AIME 2024 and weakens overall compression from 54.4% to 50.5%, lowering the accuracy gain to 6.6 points.

Alternative redundancy measures. The redundancy term is built from gzip, but its effect does not rest on that specific compressor. Here gzip scores a trace by how much a general-purpose compressor can shrink it, LZMA is a stronger compressor of the same family, and a 4-gram repetition rate is a far simpler count of verbatim n-gram repeats with no compression model at all. Despite spanning this range of modeling power, the three behave alike: LZMA raises accuracy by 7.4 points while shortening responses by 56.8%, and the 4-gram rate raises accuracy by 6.9 points at a 51.4% reduction, both above every baseline in Table 2. Their closeness suggests that what the reward needs

Table 4: TERSE on Qwen3-1.7B, outside the DeepSeek-R1-Distill-Qwen family. ΔAcc and ΔLen% are relative to the vanilla base model.

	GSM8K		MATH-500		AMC 2023		AIME 2024		AIME 2025		Overall	
Method	Acc↑	Len↓	Acc↑	Len↓	Acc↑	Len↓	Acc↑	Len↓	Acc↑	Len↓	ΔAcc↑	ΔLen%
Vanilla	90.7	2094	87.0	5096	75.2	8484	38.8	13138	28.5	13287	–	–
ARLCP	90.0	1057	**89.6**	2568	79.7	5320	42.9	9466	32.9	9755	3.0	−38.2
TERSE	90.4	1084	89.4	2580	**79.9**	4956	**44.2**	9364	**34.5**	9650	**3.7**	−39.1

is a content-agnostic measure of redundancy rather than one specific estimator. We use gzip for its ubiquity and simplicity: it is available in every standard library and adds negligible overhead.

4.4 Generalization to Other Models and Domains

Generalization to other model families. The reward depends on nothing beyond a model's output text, so we expect it to carry across model families rather than being tied to the DeepSeek-R1-Distill-Qwen models studied above. We evaluate it on Qwen3-1.7B, from a different family. As Table 4 shows, TERSE again improves accuracy while cutting length, by 3.7 points and 39.1% on average, and it is the most accurate method on AMC 2023, AIME 2024, and AIME 2025. The gains are largest on the hardest benchmarks, where it reaches 44.2 on AIME 2024 and 34.5 on AIME 2025 against ARLCP's 42.9 and 32.9, so the accuracy improvement holds where reasoning is longest and most prone to redundancy. Both averages surpass ARLCP, and neither the reward nor its coefficients were changed for this model.

Table 5: TERSE on MMLU with DeepSeek-R1-Distill-Qwen-7B, an out-of-domain benchmark. ΔAcc and ΔLen% are relative to the vanilla base model.

Method	Acc↑	Len↓	ΔAcc↑	ΔLen%
Vanilla	63.4	1257	–	–
NoThinking	51.2	**128**	−12.2	−89.8
SFT	62.8	1321	−0.6	+0.1
AdaptThink	63.6	856	+0.2	−31.9
TLMRE	63.9	872	+0.5	−30.6
ARLCP	64.1	742	+0.7	−41.0
TERSE	**64.3**	716	+0.9	−43.0

Generalization to other domains. Compressibility is computed from a response's bytes, so the reward carries no assumption about the task being mathematical. We test this on MMLU, a multiple-choice benchmark spanning 57 subjects far from the training distribution. On DeepSeek-R1-Distill-Qwen-7B, TERSE shortens responses by 43.0% while raising accuracy by 0.9 points to 64.3 (Table 5). Among the baselines, NoThinking cuts length by 89.8% but drops accuracy by 12.2 points, SFT barely shortens at all, and AdaptThink and TLMRE compress by about 30% for gains under half a point. TERSE shortens more than ARLCP, the strongest baseline, while reaching a slightly higher accuracy (64.3 against 64.1).

5 Conclusion

In this work, we begin from a simple observation: a length penalty measures how much a model writes, not how much of that writing is redundant, so two correct traces of the same length can waste very different amounts of computation. Building on this, we propose TERSE, a reward-shaping method for group-relative reinforcement learning that scores reasoning efficiency from two signals, a trace's length and its gzip compressibility. Because compressibility is read from the raw bytes of a trace, it needs no reflection lexicon, auxiliary model, or reference answer, and is not tied to a particular domain or model family. TERSE residualizes compressibility against length so the reward charges mainly the redundancy that length does not already explain, and reallocates a single fixed budget between the two signals by problem difficulty. To keep both signals stable when a hard prompt yields few correct rollouts, it pools statistics across prompts of similar difficulty and across steps. Experiments on five mathematical reasoning benchmarks show that TERSE improves the accuracy–length trade-off at both the 1.5B and 7B scales, and the gains carry to a different model family and an out-of-domain benchmark.

USE OF AI TOOLS

We used generative AI tools in preparing this paper. They helped us refine how the experiments and ablations are presented and how the results are described, and we also used them to draft and edit the writing for readability, to format references, and to help survey related work. The research itself was done by the authors: we did not use these tools to generate data, build theoretical models, prove claims, or implement the method. We checked every claim, number, and citation against our own experiments and the original sources, and we take full responsibility for the paper.

REFERENCES

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024.

Anonymous. Lz penalty: An information-theoretic repetition penalty for autoregressive language models. *arXiv preprint arXiv:2504.20131*, 2025.

Daman Arora and Andrea Zanette. Training language models to reason efficiently. *arXiv preprint arXiv:2502.04463*, 2025.

Tingcheng Bian, Yuzhe Zhang, Jing Jin, Jinchang Luo, Mingquan Cheng, Haiwei Wang, Wenyuan Jiang, and Miaohui Wang. Experience-guided reinforcement learning for adaptive chain-of-thought compression. *arXiv preprint arXiv:2605.07501*, 2026.

Ruichu Cai, Haopeng Du, Qingwen Lin, Yutong Chen, Zijian Li, and Boyan Xu. Entra: Entropy-based redundancy avoidance in large language model reasoning. *arXiv preprint arXiv:2601.07123*, 2026.

Kaiyan Chang, Chenwei Zhu, Yingfeng Luo, Yifu Huo, Chenglong Wang, Xiaoqian Liu, Qiaozhi He, Tong Xiao, Zhengtao Yu, and Jingbo Zhu. Penalizing structural redundancy in large reasoning models via anchor-based process rewards. *arXiv preprint arXiv:2602.00760*, 2026.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. Do not think that much for 2+3=? on the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187*, 2024.

Zhengxiang Cheng, Dongping Chen, Mingyang Fu, and Tianyi Zhou. Optimizing length compression in large reasoning models. *arXiv preprint arXiv:2506.14755*, 2025.

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

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Grégoire Delétang, Anian Ruoss, Paul-Ambroise Duquenne, Elliot Catt, Tim Genewein, Christopher Mattern, Jordi Grau-Moya, Li Kevin Wenliang, Matthew Aitchison, Laurent Orseau, Marcus Hutter, and Joel Veness. Language modeling is compression. In *International Conference on Learning Representations (ICLR)*, 2024.

Gongfan Fang, Xinyin Ma, and Xinchao Wang. Thinkless: Llm learns when to think. *arXiv preprint arXiv:2505.13379*, 2025.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiqing Zhao, Shiyu Ma, and Zhenyu Chen. Token-budget-aware llm reasoning. *arXiv preprint arXiv:2412.18547*, 2024.

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. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2021.

Jialiang Hong, Taihang Zhen, Kai Chen, Jiaheng Liu, Junlan Feng, Wenpeng Zhu, Jing Huo, Yang Gao, Depeng Wang, Haitao Wan, Xi Yang, Boyan Wang, Fanyu Meng, and Yuyao Zhang. Reconsidering overthinking: Penalizing internal and external redundancy in cot reasoning. *arXiv preprint arXiv:2508.02178*, 2025.

Bairu Hou, Yang Zhang, Jiabao Ji, Yujian Liu, Kaizhi Qian, Jacob Andreas, and Shiyu Chang. Thinkprune: Pruning long chain-of-thought of llms via reinforcement learning. *arXiv preprint arXiv:2504.01296*, 2025.

Minda Hu, Zexuan Qiu, Zenan Xu, Kun Li, Bo Zhou, and Irwin King. Conmax: Confidence-maximizing compression for efficient chain-of-thought reasoning. *arXiv preprint arXiv:2601.04973*, 2026.

Zhiying Jiang, Matthew Y. R. Yang, Mikhail Tsirlin, Raphael Tang, and Jimmy Lin. "less is more": Parameter-free text classification with gzip. In *Findings of the Association for Computational Linguistics: ACL 2023*, 2023.

Ming Li and Paul Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications*. Texts in Computer Science. Springer, 4 edition, 2019.

Hanbing Liu, Lang Cao, Yuanyi Ren, Mengyu Zhou, Haoyu Dong, Xiaojun Ma, Shi Han, and Dongmei Zhang. Bingo: Boosting efficient reasoning of llms via dynamic and significance-based reinforcement learning. *arXiv preprint arXiv:2506.08125*, 2025a.

Wei Liu, Ruochen Zhou, Yiyun Deng, Yuzhen Huang, Junteng Liu, Yuntian Deng, Yizhe Zhang, and Junxian He. Learn to reason efficiently with adaptive length-based reward shaping. *arXiv preprint arXiv:2505.15612*, 2025b.

Haotian Luo, Li Shen, Haiying He, Yibo Wang, Shiwei Liu, Wei Li, Naiqiang Tan, Xiaochun Cao, and Dacheng Tao. O1-pruner: Length-harmonizing fine-tuning for o1-like reasoning pruning. *arXiv preprint arXiv:2501.12570*, 2025a.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Tianjun Zhang, Li Erran Li, et al. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl, 2025b. Notion Blog.

Qin-Wen Luo, Sheng Ren, Xiang Chen, Rui Liu, Jun Fang, Naiqiang Tan, and Sheng-Jun Huang. Compress the easy, explore the hard: Difficulty-aware entropy regularization for efficient llm reasoning. *arXiv preprint arXiv:2602.22642*, 2026.

Wenjie Ma, Jingxuan He, Charlie Snell, Tyler Griggs, Sewon Min, and Matei Zaharia. Reasoning models can be effective without thinking. *arXiv preprint arXiv:2504.09858*, 2025.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*, 2025.

OpenAI. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Keqin Peng, Yuanxin Ouyang, Xuebo Liu, Zhiliang Tian, Ruijian Han, Yancheng Yuan, and Liang Ding. Think dense, not long: Dynamic decoupled conditional advantage for efficient reasoning. *arXiv preprint arXiv:2602.02099*, 2026.

Ziqing Qiao, Yongheng Deng, Jiali Zeng, Lai Wei, Guanbo Wang, Fandong Meng, Jie Zhou, Ju Ren, and Yaoxue Zhang. ConCISE: Confidence-guided compression in step-by-step efficient reasoning. *arXiv preprint arXiv:2505.04881*, 2025.

Qwen Team. Qwq: Reflect deeply on the boundaries of the unknown, 2025. `https://qwenlm.github.io/blog/qwq-32b-preview/`.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient RLHF framework. *arXiv preprint arXiv:2409.19256*, 2024.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Na Zou, Hanjie Chen, and Xia Hu. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419*, 2025.

Abdul Waheed, Chancharik Mitra, Laurie Z. Wang, Deva Ramanan, and Bhiksha Raj. Less is more tokens: Efficient math reasoning via difficulty-aware chain-of-thought distillation. *arXiv preprint arXiv:2509.05226*, 2025.

Chen Wang, Hexuan Deng, Yining Zhang, Yuchen Zhang, Jionghao Bai, Zhaochun Li, Ge Lan, and Yue Wang. Implicit compression regularization: Concise reasoning via internal shorter distributions in rl post-training. *arXiv preprint arXiv:2605.07316*, 2026.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 2022.

Heming Xia, Yongqi Li, Chak Tou Leong, Wenjie Wang, and Wenjie Li. Tokenskip: Controllable chain-of-thought compression in llms. *arXiv preprint arXiv:2502.12067*, 2025.

Violet Xiang, Chase Blagden, Rafael Rafailov, Nathan Lile, Sang Truong, Chelsea Finn, and Nick Haber. Just enough thinking: Efficient reasoning with adaptive length penalties reinforcement learning. *arXiv preprint arXiv:2506.05256*, 2025.

Edward Yeo, Yuxuan Tong, Morry Niu, Graham Neubig, and Xiang Yue. Demystifying long chain-of-thought reasoning in llms. *arXiv preprint arXiv:2502.03373*, 2025.

Zewei Yu, Lirong Gao, Bo Zheng, Junbo Zhao, Sheng Guo, and Haobo Wang. Stop unnecessary reflection: Training lrms for efficient reasoning with adaptive reflection and length coordinated penalty. *arXiv preprint arXiv:2602.12113*, 2026.

Linan Yue, Yichao Du, Yizhi Wang, Weibo Gao, Fangzhou Yao, Li Wang, Ye Liu, Ziyu Xu, Qi Liu, Shimin Di, and Min-Ling Zhang. Don't overthink it: A survey of efficient R1-style large reasoning models. *arXiv preprint arXiv:2508.02120*, 2025.

Jiajie Zhang, Nianyi Lin, Lei Hou, Ling Feng, and Juanzi Li. Adaptthink: Reasoning models can learn when to think. *arXiv preprint arXiv:2505.13417*, 2025.

Jacob Ziv and Abraham Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23(3):337–343, 1977.

Jacob Ziv and Abraham Lempel. Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, 24(5):530–536, 1978.