VERIFY UNTIL DECIDED: EXACT EARLY STOPPING FOR WEIGHTED VOTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Test-time reasoning systems sample multiple solutions, score them with a verifier, and select the parsed answer with the largest aggregate score. Dense execution verifies every solution even when the remaining scores cannot change the decision. We introduce Verify Until Decided (VUD), a sparse executor that avoids unnecessary verification while preserving the weighted-vote decision. For a fixed candidate pool, VUD uses a tie-aware certificate that is necessary and sufficient for winner invariance under all bounded assignments to the unseen verifier scores. A signed certificate extends this guarantee to candidates not yet generated, while a deferred controller preserves the dense finite-budget decision without increasing candidate generation or verifier calls on any instance. Across three public 8–9B generators and six reasoning benchmarks, covering 2,304 model–question instances, VUD reproduces every dense ThinkPRM decision at $N = 8$ for the same verifier-score realization while using 52.4% fewer verifier calls than a matched dense executor. One full-matrix online pass reduces verifier-only wall time by 18.8–40.3% across three scorer interfaces. With CISC, FUTURECERT saves 15.3% of candidates, and the full two-layer executor saves 9.1–17.5% of generation-plus-scoring wall time across generators. Prospective ThinkPRM runs save up to 48.7% end-to-end wall time at $N = 16$. VUD changes execution, not the scoring rule, terminal budget, or decision for a fixed realization.

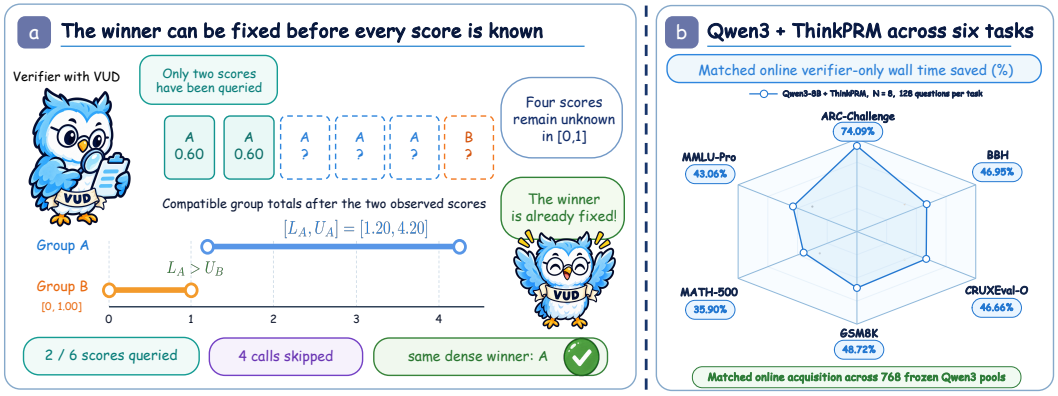


Figure 1: Why VUD can stop early and how the savings transfer. Left: in an illustrative $[0, 1]$ fixed pool, two queried A scores imply $L_A = 1.2 > U_B = 1$; the legal group-total intervals are already separated, so SCORECERT skips four calls and returns the dense winner. Right: matched online verifier-only wall-time savings for Qwen3-8B with ThinkPRM on the six frozen $N = 8$ tasks, each containing 128 questions. Each radius is the within-task elapsed-total ratio $100(1 - T_{\mathrm{VUD}}/T_{\mathrm{dense}})$ under the scorer-matched dense path. The radar uses a common zero-based 0–80% scale. It excludes candidate generation, model loading, and warm-up and represents one online pass rather than repeat-based uncertainty. Appendix Table 13 reports the Qwen micro-aggregate (44.57%) and all generator–scorer aggregates.

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.

1 INTRODUCTION

Sampling and aggregating multiple reasoning paths is a standard form of test-time scaling (Wang et al., 2023; Hariri et al., 2026). A weighted vote generates candidates, groups them by parsed answer, scores them, and returns the largest score sum. SC assigns unit weights (Wang et al., 2023); weighted variants use confidence, latent, or verifier scores (Taubenfeld et al., 2025; Kang et al., 2025; Oh & Lee, 2026; Khalifa et al., 2026). Rather than change this rule, we ask how much of a fixed finite dense computation must be performed to guarantee its identical decision.

Dense pipelines couple *decision semantics*—the weighted vote—to an *execution strategy* that scores every candidate. Yet the selector needs only its winner. In Figure 1, two queried A paths determine that winner under every legal completion of four unseen scores. This decision-preserving sparse execution certifies a computation, not answer correctness.

VUD (*Verify Until Decided*) separates weighted-vote decisions from their execution through two complementary certificates. SCORECERT reduces verification in a fixed candidate pool: it stops querying scores once no unqueried score can change the winner. FUTURECERT extends certification to a growing pool: it stops generation once no possible completion up to the terminal budget can change the winner. Both preserve the corresponding dense decision for the same score realization, arithmetic, and tie rule.

SCORECERT gives a complete, tie-aware interval test for bounded additive scores queried per candidate. FUTURECERT additionally accounts for unseen candidates, new answer groups, and negative scores. Whether generation can stop depends on the score bounds and observed answers. If it cannot, the controller reaches $N_{\max}$ and applies SCORECERT, using no more candidates or score queries than that fixed-pool fallback.

On a frozen 3×6 matrix, SCORECERT preserves all 2,304 dense ThinkPRM decisions while reducing calls from 7,279 to 3,468 (52.4%). It also lowers generation-inclusive wall time in all six prospective Qwen3/Granite $N = 16$ repeats. With CISC, FUTURECERT saves 15.3% of candidates; the joint FUTURECERT+SCORECERT executor saves 9.1–17.5% total measured wall time across generators. These are same-budget execution savings, not accuracy improvements.

Our contributions are threefold:

1. We formulate **exact sparse execution** for finite verifier-weighted voting, separating decision semantics from the computation used to realize them.
2. We derive **two complementary certificates**: SCORECERT gives a complete fixed-pool stopping test, and FUTURECERT handles future-candidate uncertainty, both respecting ties and composing with resource non-increase.
3. Across a generative PRM, CISC, and a discriminative verifier, we demonstrate **SCORECERT verification savings**, and **joint FUTURECERT+SCORECERT generation and wall-time savings**, while preserving each dense target.

Together, these results establish a reusable execution layer for bounded weighted votes.

2 PROBLEM SETUP

Let $\mathcal{X}$ and $\mathcal{Y}$ denote the prompt and candidate spaces, respectively, and write $[N] := \{1, \dots, N\}$. We model generation, parsing, and verification as maps $G : \mathcal{X} \to \mathcal{Y}^N$, $P : \mathcal{Y} \to \mathcal{A} \cup \{\bot\}$, and $V : \mathcal{X} \times \mathcal{Y} \to [\ell, u]$. For a prompt $x \in \mathcal{X}$, the pipeline is

$$\begin{aligned} (y_i)_{i\in[N]} &:= G(x), \\ a_i &:= P(y_i) \in \mathcal{A} \cup \{\bot\}, \quad i \in [N], \qquad I := \{i \in [N] : a_i \neq \bot\}, \\ s_i &:= V(x, y_i) \in [\ell, u], \qquad i \in I. \end{aligned} \tag{1}$$

Here $\mathcal{A}$ is the answer space and $\bot$ denotes an invalid parse; such a candidate receives no score and casts no vote. Thus I is the set of valid-candidate indices, and $s = (s_i)_{i\in I} \in [\ell, u]^{|I|}$ is the corresponding (possibly unqueried) score vector. Assume first that at least one candidate is valid. Let $b_1, \dots, b_G$ be the distinct valid answers in order of first occurrence, and let $C_g = \{i : a_i = b_g\}$

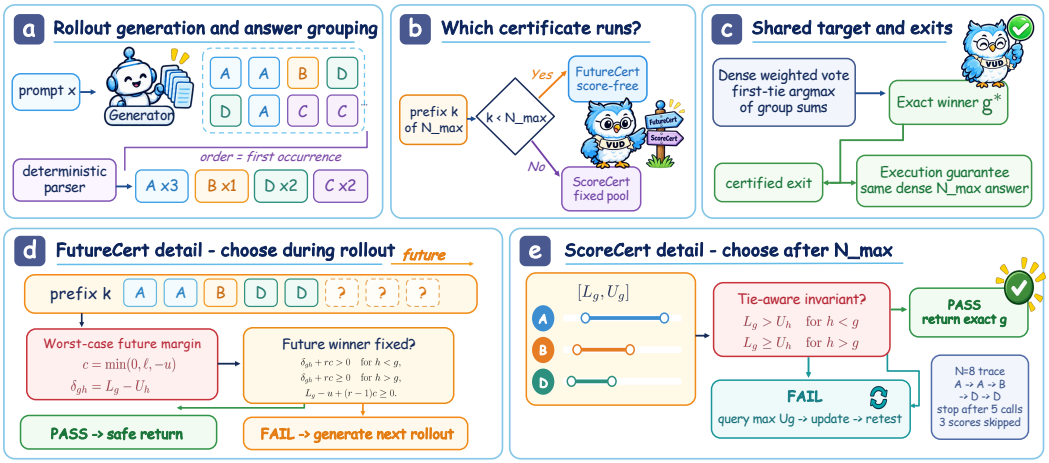


Figure 2: VUD resolves two uncertainty layers. SCORECERT removes score uncertainty in a known candidate pool; FUTURECERT additionally accounts for structural uncertainty from unseen candidates and answer groups, then falls back to fixed-pool certification if needed.

collect the candidate indices supporting answer b_g; hence $(C_1, \ldots, C_G)$ partitions I. Write $[G] := \{1, \ldots, G\}$. The group score is $S_g(s) = \sum_{i \in C_g} s_i$. Dense weighted self-consistency selects the group index

$$M(s) := \max_{h \in [G]} S_h(s),$$
$$g^*(s) := \min\{g \in [G] : S_g(s) = M(s)\}, \quad \widehat{a}(s) := b_{g^*(s)}. \tag{2}$$

In words, dense execution sums verifier scores within each parsed-answer group and returns the earliest group among those with the largest total; $\widehat{a}(s)$ is that group's answer. Thus the minimum in Eq. (2) implements the original first-occurrence tie rule. Ordinary self-consistency is the special case $s_i = 1$: FUTURECERT then reduces to exact finite-horizon majority certification, while SCORECERT is unnecessary because every weight is already known. CISC uses $s_i = \exp(p_i/T)$ for verbal confidence $p_i \in [0, 1]$ and temperature $T > 0$; dividing all weights by $\exp(1/T)$ preserves Eq. (2) and gives the known interval $[\exp(-1/T), 1]$ (Taubenfeld et al., 2025). ThinkPRM probabilities and the released discriminative score use $[0, 1]$.

Notation. Group indices encode tie priority: $h < g$ means that h appeared earlier and wins a tie against g. In a partial transcript, $O_g \subseteq C_g$ and $Q_g = C_g \setminus O_g$ are the queried and unqueried candidates; L_g, U_g are the compatible score extrema, and $\underline{\Delta}_{gh}$ is the worst-case margin of g over h.

An all-invalid pool has $I = \varnothing$ and hence no value under Eq. (2). Our evaluation handles it at the protocol layer with a deterministic null output, zero verifier calls, and an incorrect grade. Such pools remain in the generation and accuracy denominators, but not in the valid-candidate call denominator or the theorem domain. Single-answer pools are within the theorem and certify before any score query.

Our target is fixed: return the same group $g^*(s)$, and hence the same answer $\widehat{a}(s)$, while acquiring as few s_i as possible. This separates the quality of a scoring rule from the computation needed to execute it. A certificate is allowed to exploit observed answer groups, observed scores, score bounds, and the tie convention, but not gold labels or a probabilistic model of unseen scores. Both fixed- and future-pool certificates permit arbitrary finite $\ell \leq u$. The future rule must account for both signs: a future candidate can improve a rival at score u or damage the proposed winner at score ℓ. Section 3.2 gives the resulting closed form.

Our exactness claim has two distinct levels. **Algorithmic exactness** means that, given a fixed vector of verifier scores, dense execution and VUD must use identical arithmetic and tie semantics. We therefore store each score as a decimal string, interpret that string as an exact rational number, and use rational arithmetic for both dense totals and certificate bounds. This prevents floating-point summation order from changing a near-tie decision. **Runtime score reproducibility** is the separate

systems requirement that model runs reproduce the same score vector: temperature zero alone does not guarantee identical outputs under bfloat16 kernels or parallel reductions. Our endpoint audits establish exact agreement conditional on the persisted scores; cross-run agreement additionally requires a stable, versioned verifier service. Appendix I reports adversarial tie tests, historical-trace replay, and the central processing unit (CPU) overhead of rational arithmetic.

3 VERIFY UNTIL DECIDED

We address two forms of uncertainty in sequence. SCORECERT assumes known candidates and answer groups, but unknown scores. FUTURECERT additionally treats future candidates, answers, and scores as unknown. In both cases, "exact" means reproducing the specified dense executor for every completion consistent with current information, not improving its accuracy.

3.1 EXACT CERTIFICATION ON A FIXED POOL

Let $O_g \subseteq C_g$ be the queried indices in group g, and let $Q_g := C_g \setminus O_g$ be the unqueried indices. Define the observed subtotal $R_g := \sum_{i \in O_g} s_i$ and the number of unseen scores $q_g := |Q_g|$. Then

$$S_g(s) = R_g + \sum_{i \in Q_g} s_i \in [L_g, U_g], \tag{3}$$

$$(L_g, U_g) := \big(R_g + q_g \ell,\ R_g + q_g u\big). \tag{4}$$

For distinct groups $g, h \in [G]$, the smallest compatible advantage of g over h is

$$\underline{\Delta}_{gh} := \min_{\substack{s_i \in [\ell, u] \\ i \in Q_g \cup Q_h}} \big[S_g(s) - S_h(s)\big] = L_g - U_h. \tag{5}$$

Theorem 1 (Soundness and completeness)**.** *For a fixed pool with at least one nonempty answer group, the dense winner is identical for every completion of the unqueried scores in $[\ell, u]$ if and only if some group g satisfies*

$$L_g > U_h \quad \forall h < g, \qquad L_g \geq U_h \quad \forall h > g. \tag{6}$$

When Eq. (6) holds, that invariant winner is g.

Appendix A gives the complete proofs. Intuitively, even the lowest legal total for g must defeat the highest legal total of every rival. It is sufficient because Eq. (5) has the required sign for every rival. It is also necessary. If an inequality fails, assign the unseen scores of g their lower endpoint ℓ and those of the relevant rival their upper endpoint u. This legal completion realizes the adverse comparison and prevents g from being the invariant winner. The strict and weak inequalities simply encode the dense tie rule: g must strictly beat every earlier group, whereas a tie with a later group still selects g.

Corollary 1 (Maximal safe stopping under interval information)**.** *For a fixed answer partition, score interval, set of queried scores, and tie rule, any deterministic method required to reproduce the dense winner for every legal completion can stop safely if and only if Eq. (6) holds.*

This is not merely a valid stopping rule: under the information exposed to the executor, Eq. (6) characterizes the entire region in which any exact deterministic executor may terminate. If the certificate fails, Theorem 1 supplies legal completions that agree on every observation but yield different dense winners. A deterministic method cannot distinguish those completions, so stopping must be wrong for at least one of them. This is a statement about whether the method can stop now. It is not a claim that our earlier query choices were optimal, or that our query order reaches a certifiable state with the fewest calls.

SCORECERT: verification over a fixed candidate pool. SCORECERT tests the certificate before requesting any score, so a pool with only one answer group requires zero verifier calls. If the winner is not yet fixed, it selects the group with the largest current upper bound U_g, queries that group's earliest unscored candidate, updates the bounds, and tests again; ties between groups follow group order. Repeating this procedure must terminate because, after all valid candidates are scored, the bounds equal the dense totals. Each bound update costs $O(1)$, and checking all G groups costs

$O(G)$. The largest-upper-bound rule is a practical online heuristic rather than an optimal query policy. Appendix H computes, with hindsight, the smallest terminal certificate for a realized score vector.

The result also permits candidate-specific bounds without changing the proof. If unqueried candidate i has a known interval $[\ell_i, u_i]$, replace Eq. (4) by the sum of observed point scores and the corresponding unobserved coordinate bounds. The pairwise extrema remain simultaneously attainable, so Eq. (6) is still necessary and sufficient.

The analysis above treats each unqueried score as independently ranging over an interval. If additional constraints couple the scores, Appendix B replaces separate intervals with worst-case pairwise score margins over the joint feasible set. For affine utilities and a rational polytope, each rival requires one linear program. We validate this extension on exact finite sets, but do not present it as a systems result because we did not identify a deployed list-conditioned scorer that exposes both the required joint constraints and a cheaper coordinate query interface.

3.2 A COMPLETE CERTIFICATE OVER FUTURE CANDIDATES

Now fix a maximum pool size $N_{\max}$ and arbitrary finite score bounds $\ell \le u$, but suppose only k candidates have been generated. Let $r = N_{\max} - k$. Each future candidate may be invalid, join any visible group, or introduce a new answer; a valid future score lies in $[\ell, u]$. Define the worst one-slot change to a leader–rival margin as

$$c := \min\{0, \ell, -u\}. \tag{7}$$

For the current Cartesian uncertainty, let $\delta_{gh} := L_g - U_h$. Appendix B gives the extension to compact joint uncertainty through exact pairwise margins. Thus r counts the unfilled candidate slots, δ_{gh} is the current worst-case margin against a visible rival, and c is the largest possible one-slot deterioration of that margin.

Theorem 2 (Future-candidate certificate)**.** *For $r > 0$, a visible group g wins the dense $N_{\max}$ vote for every compatible completion of current unknown scores, future answers, and future scores if and only if*

$$\begin{aligned} \delta_{gh} + rc > 0 \quad \forall h < g, \qquad & \delta_{gh} + rc \ge 0 \quad \forall h > g, \\ L_g - u + (r-1)c \ge 0 \qquad & \textit{(new-group condition).} \end{aligned} \tag{8}$$

For $r = 0$, omit the last inequality and recover Theorem 1.

The certificate covers the two ways in which future generation can overturn the current leader. First, an existing rival may catch up. For any visible rival, one future slot can change the leader's margin in three worst-case ways: an invalid candidate or a candidate assigned elsewhere contributes 0; a candidate joining the leader at score ℓ contributes ℓ; and a candidate joining the rival at score u contributes $-u$. Separability across the r future slots gives

$$\min_{z_1,\dots,z_r \in \{0,\ell,-u\}} \sum_{t=1}^{r} z_t = r \min\{0, \ell, -u\} = rc. \tag{9}$$

Second, a previously unseen answer may become a new rival. Creating it consumes one slot at score u; the other $r-1$ slots contribute at worst $(r-1)c$, so the corresponding margin is $L_g - u + (r-1)c$. If every inequality holds, neither route can change the winner. If one fails, the corresponding adverse completion is attainable. This establishes sufficiency and necessity. The explicit zero in c is the invalid/elsewhere contribution; it remains available even when $0 \notin [\ell, u]$, because $[\ell, u]$ bounds valid scores rather than invalid candidates. For $0 \le \ell \le u$, $c = -u$ and Eq. (8) reduces exactly to the original nonnegative ru rule. For $u \le 0$, $c = \ell$ captures the previously missed channel that future negative scores can join and damage the leader.

FUTURECERT: deferred verification during candidate generation. Before $N_{\max}$, FUTURECERT makes no verifier calls and substitutes the score-free bounds $L_g = |C_g|\ell$, $U_g = |C_g|u$, and $\delta_{gh} = L_g - U_h$ into Eq. (8). If the certificate holds, the answer is returned immediately. Otherwise, generation continues to $N_{\max}$ and SCORECERT runs unchanged.

Corollary 2 (Per-instance resource non-increase)**.** *For every candidate/score realization, deferred* VUD *(i) returns the dense-$N_{\max}$ answer, (ii) generates at most $N_{\max}$ candidates, and (iii) makes no more verifier calls than fixed-pool* SCORECERT *on the same maximum pool.*

Table 1: Compatibility with common scoring interfaces. "Exact" means preservation of the specified finite dense vote, not answer correctness. The final column lists concrete methods evaluated in Table 11.

Interface	Known set	Fixed pool SCORECERT	Growing pool FUTURECERT	Concrete method
SC (unit vote)	$[1, 1]$	No queries	Exact majority stop	SC
Confidence-weighted SC	$[e^{-1/T}, 1]$	Exact	Exact; generation stop possible	CISC
Pointwise bounded score	$[0, 1]$	Exact	No score-free early stop if $r > 0$	ThinkPRM; Released WSC
Signed additive score	$[\ell, u]$	Exact	Exact under Eq. (8)	–
Coupled coordinate score	$\mathcal{U}_\tau$	Exact (joint)	Exact if future slots are additive	–
Full-list-only or unbounded	–	Not applicable	Not applicable	Official LatentSC

An early stop makes no verifier calls; otherwise, the controller runs the same fixed-pool fallback. With no queried scores and bounds $[0, u]$ for $u > 0$, this deferred controller cannot certify before $N_{\max}$: $L_g = 0$, so the new-group condition is $-ru < 0$. This restriction concerns score-free generation stopping; with observed scores, the general certificate in Eq. (8) can certify before the budget is exhausted. A positive score floor can enable score-free stopping: SC has $\ell = u = 1$, and CISC has $\ell = \exp(-1/T)$. The signed certificate may also hold in some nonpositive-score regimes. Any bounded additive per-candidate score supports SCORECERT; whether FUTURECERT saves generation is determined by Eq. (8). Artificially adding a floor changes the target selector rather than the executor.

3.3 EXECUTION CONTRACT AND COST ACCOUNTING

At fixed N, the certificate changes execution cost but not the selected answer. Our primary comparison measures verifier calls and wall time against dense execution at the same budget and for the same verifier-score realization. Cross-budget comparisons separately test whether a larger certified pool costs less than a smaller dense pool; they do not share the same target decision.

4 EXPERIMENTS

We evaluate SCORECERT, FUTURECERT, and their combined savings, followed by timing, scaling, and mechanism analysis.

4.1 DESIGN

Models and tasks. We evaluate Granite-3.3-8B-Instruct, Qwen3-8B, and GLM-4-9B-0414 on 128 fixed questions each from the AI2 Reasoning Challenge (ARC-Challenge) (Clark et al., 2018), BIG-Bench Hard (BBH) (Suzgun et al., 2023), the output-prediction split of CRUXEval (CRUXEval-O) (Gu et al., 2024), Grade School Math 8K (GSM8K) (Cobbe et al., 2021), MATH-500 (Hendrycks et al., 2021), and Massive Multitask Language Understanding Pro (MMLU-Pro) (Wang et al., 2024). The equal-cell macro therefore covers $3 \times 6 \times 128 = 2{,}304$ model–questions. All tasks are retained, including negative results. Invalid responses count toward generation cost but receive no verifier call.

Scorers and selectors. Base is greedy decoding and SC votes over the pool. Our dense target groups the released ThinkPRM-14B candidate probabilities (Khalifa et al., 2026). Same-pool comparators are CISC (Taubenfeld et al., 2025), Self-Certainty (Kang et al., 2025), official LatentSC where a matching Qwen checkpoint exists (Oh & Lee, 2026), and the released 1.5B `WangResearchLab/verification-1.5b` verifier of Montgomery et al. (2025) under its released weighted self-consistency (WSC) and pivotal-vote (PV) rules. Every selector receives identical candidates. Dense ThinkPRM-WSC at $N = 4$ is the scaling comparator. The CISC experiments use the weight settings in Appendix E.

Table 2: VUD results at $N_{\max} = 8$. A: SCORECERT calls on 2,304 pools. B: SCORECERT calls and verifier time, pooled over three generators. C: joint FUTURECERT+SCORECERT savings, averaged over three repeats; total time includes generation and CISC scoring.

Panel A: SCORECERT *calls*

Scorer	Gated dense	VUD	Call saved
ThinkPRM	7,279	3,468	52.36%
CISC ($T = 1$)	7,279	1,313	81.96%
Released 1.5B	7,279	3,974	45.40%

Panel B: online SCORECERT

Scorer	Dense/VUD calls	Call save	Dense/VUD wall (s)	Wall save
ThinkPRM	7,279/3,468	52.36%	44,697/26,688	40.29%
CISC	7,279/4,190	42.44%	411.62/258.28	37.25%
Released 1.5B	7,279/5,476	24.77%	111.40/90.43	18.82%

Panel C: CISC, FUTURECERT + SCORECERT

Generator	Candidate save	CISC-call save	Total wall save
GLM-4-9B	14.3%	71.2%	9.1%
Granite-3.3-8B	13.2%	87.9%	10.5%
Qwen3-8B	20.1%	82.4%	17.5%

Execution and accounting. We report answer agreement, verifier calls, tokens, and wall time. Savings are $100(1-C_{\mathrm{VUD}}/C_{\mathrm{dense}})$. A call scores one candidate, not one GPU batch. Dense and VUD both skip scoring all-invalid and single-answer pools (the zero-call gate). Timing excludes model loading and warm-up; verifier-only timing also excludes generation. Tables specify whether savings use pooled costs or means across repeats. CI denotes a bootstrap confidence interval. Appendix E summarizes the experimental settings.

4.2 FIXED-POOL SCORECERT

Finding 1: exact execution removes most verifier work. On the 3×6 matrix, SCORECERT reproduces all 2,304 dense ThinkPRM answers at $N = 8$ while reducing post-gate calls from 7,279 to 3,468 (52.36%). Appendix Table 11 gives accuracy results, and Appendix F reports stability across candidate seeds.

The result transfers across scorers (Table 2A): CISC removes 81.96% of calls, while the released 1.5B verifier removes 45.40%, with zero decision mismatches. A separate experiment times verifier queries on all 2,304 pools (Table 2B). Pooled verifier-time savings are 40.29% for ThinkPRM, 37.25% for CISC, and 18.82% for the released verifier. Tables 13–14 give model and task breakdowns. Fewer calls need not imply faster execution: CISC on GLM/ARC removes 37.21% of calls but increases time by 6.21%.

ThinkPRM saves verifier time in all 18 model–task cells, ranging from 15.66% on Granite/CRUXEval-O to 74.09% on Qwen/ARC; the aggregate gain is not confined to one benchmark.

4.3 GROWING-POOL FUTURECERT

Finding 2: generation and verification savings compose. CISC at $T = 1$ has a positive $[e^{-1}, 1]$ range supporting exact score-free stopping. On the same 2,304 ordered $N_{\max} = 8$ pools, FUTURECERT saves 15.29% of candidates and 11.65% of completion tokens, stops early on 68.14% of instances, and matches dense CISC throughout (Table 6). With unit weights, the same rule preserves dense SC and saves 37.59% of candidates (Table 5). A tighter positive score floor permits earlier certification; CISC temperature controls confirm this dependence (Table 8). ThinkPRM's $[0, 1]$ range does not permit score-free generation stopping.

In the generation-timing experiment, pooled over three generators and three repeats, FUTURECERT saves 15.86% of candidates, 13.50% of completion tokens, and 11.88% of generation wall time; all 1,073,664 completion-endpoint checks pass (Table 16). The full executor adds SCORECERT on pools that reach $N_{\max}$. It saves 71.2–87.9% of CISC calls and 9.1–17.5% of generation+scoring wall time across generators (Table 2C). FUTURECERT saves generation, while SCORECERT saves verification on pools that reach the budget (Table 7).

At $T = 0.5$, FUTURECERT saves no candidates, yet SCORECERT removes 70.85% of calls (Table 8), demonstrating their complementarity. CISC-call savings count omitted confidence queries, not

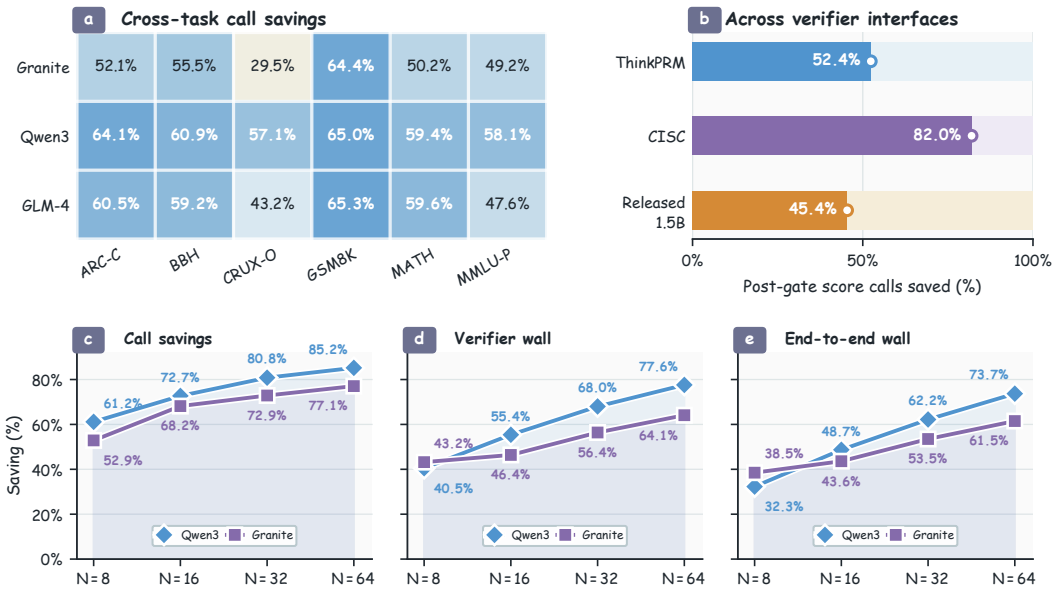


Figure 3: SCORECERT savings. (a) ThinkPRM calls across 18 model–task combinations at $N = 8$. (b) Calls across three scorers, using standard CISC at $T = 1$. (c–e) ThinkPRM call, verifier-time, and end-to-end time savings as N increases. Calls are pooled across repeats; time savings are repeat means (Table 3).

generations, and combine both stages rather than adding their percentages. All nine timing repeats reduce total wall time (3.4–22.9%).

4.4 SCORECERT SYSTEM TIMING AND SCALING

Finding 3: verifier savings survive end to end. Figure 3 reports matched calls, verifier-only wall, and generation-inclusive wall for SCORECERT from prospective Qwen3/Granite runs at $N \in \{8, 16, 32, 64\}$. Within each same-N pair, dense and VUD share measured generation cost; the verifier paths run independently in counterbalanced order.

Same-N savings do not imply that a larger certified pool is cheaper than a smaller dense pool.

Certified $N = 8$ uses 3,468 versus 3,028 gated calls for dense $N = 4$ (+14.53%). In the separate budget comparison, sequential SCORECERT saves 45.56%/44.10% same-N verification wall at $N = 8/16$; matched batch-size-two execution saves 48.06%/49.97%. Yet certified $N = 8$ is 25.20% slower than optimized dense $N = 4$ (95% interval 11.2–38.2% slower), or 27.89% slower with generation included (20.7–35.1%).

Table 3: SCORECERT timing on 48 questions, with three repeats per row. Calls and call savings use pooled totals; time savings are repeat means. E2E includes generation; Range is its minimum–maximum savings. Diff. counts selected-candidate index differences between independent scoring runs.

		Total calls (3 repeats)		Pooled	Repeat mean			
System	N	Dense	VUD	Call saved	Verify saved	E2E saved	Range (%)	Diff.
Qwen/ThinkPRM	8	278	108	61.2%	40.5%	32.3%	13.6–49.9	2
	16	619	169	72.7%	55.4%	48.7%	46.8–50.5	0
	32	1,517	291	80.8%	68.0%	62.2%	56.5–69.0	0
	64	3,723	551	85.2%	77.6%	73.7%	71.5–76.3	0
Granite/ThinkPRM	8	505	238	52.9%	43.2%	38.5%	30.7–48.0	10
	16	1,203	383	68.2%	46.4%	43.6%	34.4–50.3	8
	32	2,555	693	72.9%	56.4%	53.5%	51.2–56.4	5
	64	5,512	1,264	77.1%	64.1%	61.5%	60.2–63.6	2

Table 3 gives repeat totals and timing ranges; Appendix Table 12 gives absolute three-path times.

At $N = 16$, generation-inclusive wall falls 48.7% for Qwen3/ThinkPRM (46.8–50.5%) and 43.6% for Granite/ThinkPRM (34.4–50.3%); all six repeats are positive. Separately, the released verifier saves 28.6% verifier wall (17.6–36.1%) but only 1.5% end to end because generation dominates; its $N = 8$ range includes –2.8%, and both same-score checks have zero mismatch.

4.5 Mechanism and boundaries

Finding 4: concentrated support enables early certification. In 160,000 controlled pools, tighter bounds and concentrated support reduce queries; diffuse $[0, 1]$ pools approach dense execution. Dominant-group share correlates with query fraction ($\rho = -0.66$ to -0.87), with zero mismatches (Appendix G). The observed scaling pools show the same pattern: at $N = 64$, pools with at least 75% of valid candidates in one answer group query only 8.2% on average, versus 42.7% when the largest group is below 50% (Table 17). Larger pools offer more opportunities to omit scores, but the realized gain depends on how their answers are distributed.

5 Related work

Voting and verifier-based selection. Self-consistency aggregates sampled answers (Wang et al., 2023); CISC replaces equal votes with model confidence (Taubenfeld et al., 2025), while discriminative verification offers a lower-cost alternative to generative scoring (Montgomery et al., 2025). These choices determine the target decision or the cost of each score. VUD instead takes a bounded additive selector as given and removes scores that cannot change its winner. Its accuracy therefore equals that of the specified dense selector, rather than improving the selector itself.

Adaptive generation and stopping. Early-stopping self-consistency reduces sampling (Li et al., 2024), and DeepConf filters weak traces using internal confidence (Fu et al., 2025b). Certified Self-Consistency and CITE provide statistical guarantees for the response-distribution mode (Cordero-Encinar & Duncan, 2025; Ota et al., 2026); MARS instead controls disagreement with the full-budget vote probabilistically (Chen et al., 2026). FutureCert preserves a different target: one realized finite-budget weighted vote, for every legal future completion under the declared score bounds. Unlike unweighted voting, this requires accounting for both unseen answers and unknown weights; if generation cannot stop, ScoreCert still reduces terminal verification.

Selective verification and its interface. Adaptive Generate-Rank-Verify searches for a verified positive at low expected cost (Dughmi et al., 2026), while Best-of-Evidence allocates partial evidence to improve fixed-pool selection (Zhang et al., 2026). Pivotal-vote analysis identifies when a single-ballot substitution can affect a panel decision (Shu, 2026). VUD addresses multiple unknown, bounded score coordinates and certifies the unchanged dense weighted vote. Set-conditioned scorers such as Multi-Sequence Verifiers (Kim et al., 2026) raise a separate interface issue: skipping a coordinate is useful only if the service can expose it cheaply and consistently. Appendix C summarizes these target and guarantee distinctions.

6 Limitations and conclusion

VUD assumes bounded additive scores, fixed ties, stable coordinate queries, and a valid group; it preserves decisions, not correctness, and does not optimize acquisition or batching. The online audit has one pass, while end-to-end timing uses 48 questions, one hardware family, and coordinate-queryable verifiers. Within scope, ScoreCert removes 52.4% of ThinkPRM calls beyond a shared gate, while the full CISC executor saves 13.2–20.1% of candidates and 9.1–17.5% of total wall time, preserving every audited dense decision. Dense decision semantics need not imply dense execution.

AI use statement

Generative AI tools assisted manuscript writing and polishing, literature retrieval and discovery, and preparation of figures and verification scripts. The authors are responsible for reviewing and approving all AI-assisted content.

Reproducibility statement

All assumptions and executable numerical semantics appear in Sections 2–3, with complete proofs and extensions in Appendices A, B, and H. Section 4 and Appendix E specify datasets, decoding, verifier execution, hardware, uncertainty estimates, and timing estimands. The anonymous supplement contains immutable candidate pools, per-question score-acquisition traces, protocol manifests, deterministic figure generation, and claim-checking scripts.

References

Pranjal Aggarwal, Aman Madaan, Yiming Yang, and Mausam. Let's sample step by step: Adaptive-consistency for efficient reasoning and coding with LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. URL https://arxiv.org/abs/2305.11860.

Ehsan Aghazadeh, Ahmad Ghasemi, Hedyeh Beyhaghi, and Hossein Pishro-Nik. CGES: Confidence-guided early stopping for efficient and accurate self-consistency. *arXiv preprint arXiv:2511.02603*, 2025. URL https://arxiv.org/abs/2511.02603.

Wenbo Chen, Puheng Li, Mengyang Liu, Weijie Su, and Tianpei Xie. MARS: Margin-adversarial risk-controlled stopping for parallel LLM test-time scaling. *arXiv preprint arXiv:2606.12935*, 2026. URL https://arxiv.org/abs/2606.12935.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018. URL https://arxiv.org/abs/1803.05457.

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

Paula Cordero-Encinar and Andrew B. Duncan. Certified self-consistency: Statistical guarantees and test-time training for reliable reasoning in LLMs. *arXiv preprint arXiv:2510.17472*, 2025. URL https://arxiv.org/abs/2510.17472.

Shaddin Dughmi, Mahdi Haghifam, and Yusuf Hakan Kalayci. Adaptive generate-rank-verify: Inference-time search with costly verification. *arXiv preprint arXiv:2605.17609*, 2026. URL https://arxiv.org/abs/2605.17609.

Yichao Fu, Junda Chen, Siqi Zhu, Zheyu Fu, Zhongdongming Dai, Yonghao Zhuang, Yian Ma, Aurick Qiao, Tajana Rosing, Ion Stoica, and Hao Zhang. Efficiently scaling LLM reasoning with certaindex. In *Advances in Neural Information Processing Systems*, 2025a. URL https://arxiv.org/abs/2412.20993.

Yichao Fu, Xuewei Wang, Yuandong Tian, and Jiawei Zhao. Deep think with confidence. *arXiv preprint arXiv:2508.15260*, 2025b. URL https://arxiv.org/abs/2508.15260.

Alex Gu, Baptiste Roziere, Hugh Leather, Armando Solar-Lezama, Gabriel Synnaeve, and Sida I. Wang. CRUXEval: A benchmark for code reasoning, understanding and execution. *arXiv preprint arXiv:2401.03065*, 2024.

Mohsen Hariri, Weicong Chen, Nahal Shahini, Vikash Singh, Kai Ye, Amirhossein Samandar, Debargha Ganguly, Sreehari Sankar, Yanyan Zhang, Shouren Wang, Jerry Peng, Biyao Zhang, Michael Hinczewski, and Vipin Chaudhary. Test-time scaling in reasoning LLMs: Inference regimes, evaluation, and reproducibility. *arXiv preprint arXiv:2608.04001*, 2026. URL https://arxiv.org/abs/2608.04001.

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 *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021. URL https://arxiv.org/abs/2103.03874.

Jingkai Huang, Will Ma, and Zhengyuan Zhou. Optimal bayesian stopping for efficient inference of consistent LLM answers. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2602.05395.

Ishan Jindal, Sai Prashanth Akuthota, Jayant Taneja, and Sachin Dev Sharma. The path of least resistance: Guiding LLM reasoning trajectories with prefix consensus. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2601.21494.

Zhewei Kang, Xuandong Zhao, and Dawn Song. Scalable best-of-n selection for large language models via self-certainty. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2502.18581.

Muhammad Khalifa, Rishabh Agarwal, Lajanugen Logeswaran, Jaekyeom Kim, Hao Peng, Moontae Lee, Honglak Lee, and Lu Wang. Process reward models that think. *Transactions on Machine Learning Research*, 2026. URL https://openreview.net/forum?id=FPVCb0WMuN.

Yegon Kim, Seungyoo Lee, Chaeyun Jang, Hyungi Lee, and Juho Lee. Parallel test-time scaling with multi-sequence verifiers. *arXiv preprint arXiv:2603.03417*, 2026. URL https://arxiv.org/abs/2603.03417.

Junpei Komiyama, Daisuke Oba, and Masafumi Oyamada. Best-of-∞: Asymptotic performance of test-time LLM ensembling. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2509.21091.

Peng Kuang, Haibo Jin, Xiaoyu Han, Yanli Wang, Xiaopeng Yuan, Ye Yu, Kaidi Xu, and Haohan Wang. KV-PRM: Efficient process reward modeling via KV-cache transfer for multi-agent test-time scaling. *arXiv preprint arXiv:2607.09153*, 2026. URL https://arxiv.org/abs/2607.09153.

Yiwei Li, Peiwen Yuan, Shaoxiong Feng, Boyuan Pan, Xinglin Wang, Bin Sun, Heda Wang, and Kan Li. Escape sky-high cost: Early-stopping self-consistency for multi-step reasoning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=ndR8Ytrzhh.

Kyle Montgomery, Sijun Tan, Yuqi Chen, Siyuan Zhuang, Tianjun Zhang, Raluca Ada Popa, and Chenguang Wang. Budget-aware test-time scaling via discriminative verification. *arXiv preprint arXiv:2510.14913*, 2025. URL https://arxiv.org/abs/2510.14913.

Jungsuk Oh and Jay-Yoon Lee. Latent self-consistency for reliable majority-set selection in short- and long-answer reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 32591–32599, 2026. doi: 10.1609/aaai.v40i38.40536. No. 38.

Hirofumi Ota, Naoto Iwase, Yuki Ichihara, Junpei Komiyama, and Masaaki Imaizumi. CITE: Anytime-valid statistical inference in LLM self-consistency. *arXiv preprint arXiv:2605.05873*, 2026. URL https://arxiv.org/abs/2605.05873.

Vinod Raman, Hilal Asi, and Satyen Kale. AdaBoN: Adaptive best-of-n alignment. *arXiv preprint arXiv:2505.12050*, 2025. URL https://arxiv.org/abs/2505.12050.

Yang Shu. Blind to the pivotal vote: Aggregate independence metrics miss where verification actually helps. *arXiv preprint arXiv:2608.06940*, 2026. URL https://arxiv.org/abs/2608.06940.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. Challenging BIG-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 13003–13051, 2023. doi: 10.18653/v1/2023.findings-acl.824. URL https://aclanthology.org/2023.findings-acl.824/.

Amir Taubenfeld, Tom Sheffer, Eran Ofek, Amir Feder, Ariel Goldstein, Zorik Gekhman, and Gal Yona. Confidence improves self-consistency in LLMs. In *Findings of the Association for Computational Linguistics: ACL 2025*, 2025. URL https://arxiv.org/abs/2502.06233.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. *arXiv preprint arXiv:2406.01574*, 2024.

Cenwei Zhang, Teng Fang, Yuxia Wang, Derek Li, Bryan Dai, and Lei You. Best-of-evidence: Best-of-n selection under partial verification. *arXiv preprint arXiv:2607.20950*, 2026. URL https://arxiv.org/abs/2607.20950.

Tong Zheng, Chengsong Huang, Runpeng Dai, Yun He, Rui Liu, Xin Ni, Huiwen Bao, Kaishen Wang, Hongtu Zhu, Jiaxin Huang, Furong Huang, and Heng Huang. Parallel-probe: Towards efficient parallel thinking via 2d probing. *arXiv preprint arXiv:2602.03845*, 2026. URL https://arxiv.org/abs/2602.03845.

A Proof details

Theorem 1.

Proof. At least one nonempty group ensures that the dense argmax is defined. For sufficiency, every compatible completion obeys $S_g \geq L_g$ and $S_h \leq U_h$. A strict inequality excludes every earlier tie-priority rival; a weak inequality permits a tie only with a later group. For necessity, suppose g is invariant. For each rival h, assign unseen scores of g to ℓ and those of h to u. This completion attains L_g and U_h simultaneously. Invariance forces $L_g > U_h$ for $h < g$ and $L_g \geq U_h$ for $h > g$.

For candidate-specific intervals, define $L_g = \sum_{i \in O_g} s_i + \sum_{i \in C_g \setminus O_g} \ell_i$ and $U_g = \sum_{i \in O_g} s_i + \sum_{i \in C_g \setminus O_g} u_i$. The same corner assignment attains the winner's lower bound and any one rival's upper bound simultaneously, so the proof and the tie-aware condition are unchanged. □

Corollary 1.

Proof. If Eq. (6) holds, Theorem 1 shows that every legal completion has the same dense winner, so returning that winner is safe. Conversely, suppose the certificate fails. Theorem 1 then gives at least two legal completions that agree on the entire observed transcript but have different dense winners. A deterministic method receives identical information on those completions and must return the same output on both; it is therefore wrong on at least one. Hence no exact deterministic method can stop outside the certified region. □

Theorem 2.

Proof. For each visible rival, a future slot has independently attainable margin contribution with minimum $c = \min\{0, \ell, -u\}$: invalid/elsewhere gives 0, joining g at its lower endpoint gives ℓ, and joining the rival at its upper endpoint gives $-u$. Summing gives rc and the tie-aware first two inequalities. A future new group must contain at least one slot; assigning that slot u and optimizing all remaining slots gives $L_g - u + (r-1)c$. These constructions attain every lower bound, so failure of any inequality yields a legal winner-changing completion.

The nonnegative-score formula cannot be used when negative scores are allowed. For example, with two earlier-tied totals $S_g = S_h = -0.1$, interval $[-1, -0.1]$, and two slots, assigning both future candidates to g at -1 changes the winner to h. The signed formula includes this channel because $c = \ell = -1$. □

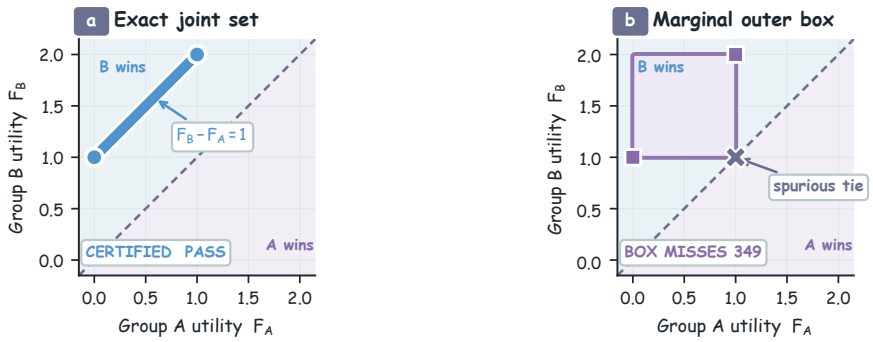


Figure 4: Joint bounds can certify more winners. Left: $F_B - F_A = 1$ throughout the feasible set. Right: independent bounds introduce an impossible tie. On 3,814 finite sets, joint bounds certify all 656 invariant winners; independent bounds miss 349.

Corollary 2.

Proof. If FUTURECERT succeeds after $k < N_{\max}$ candidates, Theorem 2 proves that its answer equals the dense-$N_{\max}$ answer; the controller has generated only k candidates and has made no verifier calls. If the test never succeeds early, the controller generates exactly $N_{\max}$ candidates and invokes the same SCORECERT procedure as the fixed-pool baseline. Theorem 1 proves exactness in this fallback case, and the two branches establish both per-instance resource inequalities. □

B JOINT UNCERTAINTY AND LIST-CONDITIONED BOUNDARY

Theorem 3 (Joint-margin certificate; theory-only extension)**.** *Let a transcript τ define a nonempty compact joint uncertainty set $\mathcal{U}_\tau$, with continuous dense group utilities $F_g(z)$. Choose any feasible reference state, let g be its tie-aware winner, and define $\delta_{gh} = \min_{z \in \mathcal{U}_\tau}[F_g(z) - F_h(z)]$. Then g is invariant over $\mathcal{U}_\tau$ iff $\delta_{gh} > 0$ for every earlier $h < g$ and $\delta_{gh} \geq 0$ for every later $h > g$.*

Proof. Sufficiency follows because every state has $F_g - F_h \geq \delta_{gh}$; strictness only against earlier rivals implements the tie rule. For necessity, compactness and continuity attain each minimum: a nonpositive margin against an earlier rival, or a negative margin against a later rival, supplies a state where g loses. Marginal outer boxes remain sound but can be incomplete: on $\mathcal{U} = \{(0,1),(1,2)\}$ the second group always wins by one, while its marginal lower bound equals the first group's marginal upper bound and fails the required strict test. □

The future-candidate theorem extends to this setting by using the joint δ_{gh} and $L_g = \min_{z \in \mathcal{U}_\tau} F_g(z)$ in Eq. (8). This requires future slots to contribute independent additive scores without changing existing utilities. For $r = 0$, it recovers Theorem 3.

The joint-margin theorem does not by itself make every set-based verifier amenable to sparse queries. Suppose an application programming interface (API) accepts only sublists. Two possible verifiers may both return score 0 for either singleton, yet return $(1,0)$ and $(0,1)$ on the full two-candidate list. All sublist observations are then identical, although the full-list winners differ, so an exact controller cannot skip the full-list call without additional assumptions. The theorem does cover a coordinate oracle evaluated in a fixed full-list context when a valid joint feasible set is known, but any wall-time benefit still depends on the verifier architecture.

This extension is evaluated on synthetic joint sets, not deployed list-conditioned verifiers.

C RELATED-WORK COMPARISON

Table 4 distinguishes the computation each method promises to preserve. "Realized vote" means one particular finite candidate/score computation, not a population mode. Broader adjacent work

spans empirical or risk-controlled stopping (Aggarwal et al., 2023; Fu et al., 2025a; Raman et al., 2025; Jindal et al., 2026), population or infinite-budget targets (Huang et al., 2026; Komiyama et al., 2026), and target-relaxing parallel verification (Zheng et al., 2026); the main text cites the closest representatives to keep its comparison compact.

Table 4: Scope map for the closest efficiency directions.

Method family	**Preserved target**	**Guarantee**	**Reduced stage**
VUD	Finite weighted vote	Exact	Scores; generation[†]
Pivotal Vote	Binary one-ballot panel	Exact	Conditional verifier call
ESC / DeepConf	Endpoint/mode proxy	Empirical/approx.	Generation/filtering
CGES (Aghazadeh et al., 2025)	Confidence-weighted answer	Conditional guarantees	Generation
MARS	Full-budget vote	High probability	Generation
CITE / W-CITE	Population (weighted) mode	Statistical	Generation
Multi-Sequence Verifier	Set-conditioned scores	Empirical	Verification
KV-PRM (Kuang et al., 2026)	KV-conditioned PRM	Theory + empirical	Per-score cost

[†]Generation savings require a score range and realized margin that let the signed certificate fire.

D GENERATION-STOPPING COST CONTROLS

With unit weights, FUTURECERT reduces to exact early stopping for SC. On 2,304 ordered pools at $N = 8$, it generates 4.99 candidates on average, saving 37.59% of candidates and 31.93% of completion tokens with zero disagreement from dense SC (Table 5).

Table 5: FUTURECERT with SC on 2,304 pools per budget. Savings are relative to fixed-budget SC; tokens count completions.

$N_{\max}$	Mean generated	Candidate save	Token save	Early-stop rate	Mismatch
1	1.000	0.00%	0.00%	0.00%	0
4	2.460	38.51%	32.92%	82.47%	0
8	4.993	37.59%	31.93%	89.67%	0

Table 6 evaluates the combined CISC executor. Across 6,912 question–budget settings ($N_{\max} \in \{1, 4, 8\}$), VUD matches dense CISC while reducing candidates and verifier calls.

Table 6: Joint FUTURECERT+SCORECERT results with CISC on 2,304 pools per budget. Savings use pooled costs relative to gated dense CISC.

$N_{\max}$	Mean generated	Candidate save	Token save	Dense/VUD CISC calls	Call save	Mismatch
1	1.00	0.00%	0.00%	0/0	–	0
4	3.37	15.77%	12.81%	3,028/1,062	64.93%	0
8	6.78	15.29%	11.65%	7,279/1,313	81.96%	0

We also measure generation and CISC scoring on 48 questions spanning six tasks, with three dense and deferred runs per generator. Table 7 combines generation, parsing, control, and scoring time. Dense scoring of each deferred run confirms its selected answer.

Original CISC tunes temperature on a held-out set (Taubenfeld et al., 2025); our main cost results use a fixed $T = 1$. Table 8 varies this setting to separate generation savings from call savings. At $T = 0.5$, FUTURECERT never stops generation, yet terminal SCORECERT still removes 70.85% of gated calls. Raising the lower bound permits earlier generation stops and further call savings. All 13,824 temperature–budget cases ($N \in \{4, 8\}$) preserve the same-temperature dense CISC decision. Each temperature is compared against its corresponding dense selector; these savings do not establish the gain at an accuracy-tuned temperature.

Table 7: Joint FUTURECERT+SCORECERT savings with CISC at $N_{\max} = 8$: 48 questions and three repeats. Entries are repeat means; the last column lists total-time savings for each repeat. Total time includes generation and scoring.

Generator	Candidate save	CISC-call save	Mean total save	Raw total-save repeats
GLM-4-9B	14.3%	71.2%	9.1%	11.1%, 3.4%, 12.9%
Granite-3.3-8B	13.2%	87.9%	10.5%	8.4%, 10.2%, 12.8%
Qwen3-8B	20.1%	82.4%	17.5%	22.9%, 11.4%, 18.3%

Table 8: Effect of CISC temperature on 2,304 pools at $N_{\max} = 8$. FUTURECERT saves candidates and tokens; the combined executor saves calls relative to 7,279 gated-dense calls.

T	Score floor	Candidate saved	Token saved	VUD calls	Call saved
0.5	e^{-2}	0.00%	0.00%	2,122	70.85%
1	e^{-1}	15.29%	11.65%	1,313	81.96%
2	$e^{-0.5}$	25.82%	20.91%	926	87.28%

E EXPERIMENTAL SETUP

Experiments used eight NVIDIA L20Y GPUs. Generation uses native chat templates, temperature 0.8, top-$p = 0.95$, a 12,288-token context, and up to 8,192 new tokens, with four questions per batch. Thinking is disabled for Granite and Qwen.

ThinkPRM uses bfloat16 critique generation at temperature zero, a 32,768-token context, and up to 8,172 output tokens. Yes/No probabilities are computed in float32; prefix caching and TF32 are disabled. The released 1.5B verifier uses bfloat16, FlashAttention-2, batch size four, and no KV cache or TF32. Its timing includes tokenization, packing, transfer, scoring, and selection. Checkpoint versions and full configurations are supplied with the code.

CISC uses $\exp((p_i - 1)/T)$ for fixed-score and generation-stopping experiments, with $T = 1$ unless varied explicitly.

Dense and VUD use matched settings and counterbalanced execution order. Generation-timing arms run independently. Decision checks use common scores; independent-run differences are reported in Appendix G. Tables 9–10 summarize the experiment sizes and timing scope.

Table 9: Timing setup. All measurements exclude model loading and warm-up.

Experiment	Setup	Measured time
Budget comparison	3 generators, $N \in \{1, 4, 8, 16\}$, 4 repeats, 4 execution paths	Verifier time; matched vLLM settings and no shared cache
Qwen end-to-end	48 questions, $N \in \{8, 16, 32, 64\}$, 3 repeats; 2 GPUs/path	Generation plus verification; generation cost shared within each dense/VUD pair
Granite replication	Same 48 questions; 3 ThinkPRM repeats (2 GPUs/path); 4 released-1.5B repeats (1 GPU/path)	ThinkPRM end-to-end time; released-verifier scoring and end-to-end time
Full-matrix scoring	3 generators, 6 tasks, 128 questions/task, $N = 8$; 1 pass/path	Verifier time for ThinkPRM, CISC, and released 1.5B, with matched batching
CISC generation and joint execution	48 questions, 3 generators, 3 repeats, $N = 8$	Generation time alone and generation plus scoring

For repeated timing, we report mean savings and the observed minimum–maximum range.

F ACCURACY AND CALL STABILITY

Table 11 reports dense-selector accuracy. VUD produces the same answers with fewer scores. Across three candidate-generation seeds, gated-dense/VUD call totals are 7,279/3,468, 7,420/3,577, and 7,263/3,476. Savings are 52.36%, 51.79%, and 52.14% (mean 52.10%, sample standard deviation 0.28 percentage points).

Table 10: Evaluation sizes. N is the candidate budget per question.

Evaluation	Budget	Size
Decision preservation / work	N8	2,304 model–questions; 18 fixed 128-question cells; 3 candidate seeds
Matched verifier wall	N1/4/8/16	192 reported model–N–repeat–path cells; 4 local repeats per path
Full-matrix verifier wall	N8	2,304 model–questions; one online pass per dense/VUD path
Prospective E2E	N8/16/32/64	One 48-question roster; 3 ThinkPRM repeats per system–N row
Mechanism map	N4/8/16/32	160,000 synthetic pools
Certificate validation	Small states	287,220 fixed states, 3,814 joint sets, and 372,555 signed-future completions

Table 11: Accuracy (%) with 128 questions per task. VUD matches ThinkPRM N8 with fewer calls. Bold marks the best model-level average. Official LatentSC is available only for Qwen.

Generator	Method	ARC	BBH	CRUX	GSM8K	MATH	MMLU	Avg.
Granite	Base	75.78	68.75	35.16	89.84	71.88	42.97	64.06
	SC	79.69	75.00	43.75	93.75	81.25	51.56	70.83
	CISC	79.69	76.56	41.41	93.75	80.47	51.56	70.57
	Self-Certainty	78.91	77.34	42.97	94.53	81.25	52.34	71.22
	Released WSC	78.91	75.78	42.19	92.19	81.25	51.56	70.31
	Dense N4	77.34	78.12	46.09	92.19	79.69	53.91	71.22
	ThinkPRM N8	79.69	79.69	50.78	93.75	78.91	56.25	**73.18**
Qwen	Base	96.09	79.69	75.78	92.19	84.38	68.75	82.81
	SC	96.88	82.03	79.69	94.53	92.19	74.22	86.59
	CISC	96.88	83.59	77.34	94.53	92.19	73.44	86.33
	Self-Certainty	96.88	83.59	79.69	94.53	90.62	73.44	86.46
	Official LatentSC	96.88	82.03	77.34	94.53	91.41	73.44	85.94
	Released WSC	96.88	82.03	78.91	95.31	92.19	74.22	86.59
	Dense N4	96.09	82.03	79.69	94.53	91.41	71.88	85.94
	ThinkPRM N8	96.88	84.38	82.03	94.53	92.19	71.88	**86.98**
GLM	Base	83.59	66.41	49.22	85.94	75.78	69.53	71.74
	SC	92.97	71.88	57.81	91.41	78.91	68.75	76.95
	CISC	93.75	73.44	58.59	90.62	78.12	68.75	77.21
	Self-Certainty	92.97	74.22	61.72	91.41	79.69	69.53	78.26
	Released WSC	92.97	77.34	59.38	92.19	78.12	67.97	77.99
	Dense N4	91.41	72.66	63.28	92.19	79.69	71.88	78.52
	ThinkPRM N8	93.75	75.78	69.53	92.97	80.47	72.66	**80.86**

G ADDITIONAL TIMING AND MECHANISM RESULTS

The budget comparison covers three generators, $N \in \{1, 4, 8, 16\}$, four repeats, and four paths (192 timing cells). Sequential SCORECERT checks after each query; the batch-size-two version checks after each batch. They save 45.56%/44.10% and 48.06%/49.97% verifier time at N8/N16, respectively. Certified N8 remains 25.20% slower than dense N4 (95% CI: 11.2–38.2%), or 27.89% slower including generation (20.7–35.1%). The 24-cell prefix-cache comparison gives a ratio-of-ratios of 0.992 (0.914–1.091).

Table 12: Sequential SCORECERT verifier time in seconds/question, averaged over three models and four repeats. Savings are mean paired reductions relative to gated dense execution.

N	Raw dense	Gated dense	ScoreCert seq.	Paired save vs gated
1	6.06	0.00	0.00	–
4	19.88	8.01	7.37	+6.06%
8	40.74	18.21	10.33	+45.56%
16	82.78	37.55	22.31	+44.10%

The N4 confidence interval includes parity; N8 and N16 show clear same-budget reductions.

A separate released-1.5B experiment uses 3,036 development pools, batch size four, and two scoring runs. Parsing-plus-verification time savings range from 12.30–14.39% at N8 and 25.08–26.13% at N16 across generators.

On the 2,304 formal instances, raw/gated/certified N8 calls are 17,624/7,279/3,468; gated dense N4 uses 3,028 calls. Table 11 reports the complete accuracy matrix.

Table 13: SCORECERT on all 2,304 pools at $N = 8$, with one timed pass per path. Calls and verifier seconds are dense/VUD totals; savings use pooled costs. Generation is excluded.

Scorer	Generator	Dense/VUD calls	Call saved (%)	Dense/VUD wall	Wall saved (%)
ThinkPRM	Qwen	1,678/685	59.18%	10,545.3/5,845.3	44.57%
	Granite	2,916/1,534	47.39%	17,372.6/11,287.1	35.03%
	GLM	2,685/1,249	53.48%	16,778.5/9,555.9	43.05%
ThinkPRM	**All**	**7,279/3,468**	**52.36%**	**44,696.5/26,688.3**	**40.29%**
CISC	Qwen	1,678/973	42.01%	104.10/66.04	36.56%
	Granite	2,916/1,666	42.87%	130.24/81.61	37.34%
	GLM	2,685/1,551	42.23%	177.28/110.63	37.59%
CISC	**All**	**7,279/4,190**	**42.44%**	**411.62/258.28**	**37.25%**
Released 1.5B	Qwen	1,678/1,291	23.06%	28.75/23.51	18.24%
	Granite	2,916/2,155	26.10%	39.06/31.31	19.82%
	GLM	2,685/2,030	24.39%	43.59/35.61	18.31%
Released 1.5B	**All**	**7,279/5,476**	**24.77%**	**111.40/90.43**	**18.82%**

Table 14: SCORECERT at $N = 8$, with 128 questions per task. Entries are **call savings / verifier-time savings (%)**; negative values indicate a slowdown. Pooled rows use total costs. Generation is excluded.

Generator	Dataset	ThinkPRM	CISC	Released 1.5B
Qwen3	ARC-Challenge	64.06 / 74.09	39.06 / 7.47	31.25 / 2.38
	BBH	58.49 / 46.95	38.27 / 34.43	24.53 / 31.39
	CRUXEval-O	58.06 / 46.66	42.93 / 34.61	20.10 / 7.49
	GSM8K	66.67 / 48.72	42.50 / 37.34	23.33 / 28.88
	MATH-500	60.07 / 35.90	43.88 / 36.02	24.10 / 15.82
	MMLU-Pro	57.47 / 43.06	43.44 / 41.06	22.62 / 17.77
	Pooled	59.18 / 44.57	42.01 / 36.56	23.06 / 18.24
Granite	ARC-Challenge	52.07 / 50.78	35.54 / 40.85	27.27 / 42.92
	BBH	55.36 / 46.37	40.85 / 36.07	29.34 / 10.76
	CRUXEval-O	25.98 / 15.66	43.71 / 40.92	15.02 / 20.12
	GSM8K	66.67 / 53.79	42.80 / 39.95	36.36 / 22.49
	MATH-500	51.37 / 39.30	44.92 / 36.26	28.91 / 24.93
	MMLU-Pro	52.17 / 40.57	43.65 / 35.60	28.95 / 16.08
	Pooled	47.39 / 35.03	42.87 / 37.34	26.10 / 19.82
GLM-4	ARC-Challenge	60.47 / 39.32	37.21 / -6.21	32.56 / 44.13
	BBH	59.20 / 48.76	41.10 / 34.60	27.15 / 8.17
	CRUXEval-O	43.58 / 33.81	40.50 / 32.22	19.83 / 24.21
	GSM8K	65.75 / 64.94	44.29 / 41.50	30.59 / 16.92
	MATH-500	57.02 / 47.38	45.55 / 39.47	28.60 / 23.44
	MMLU-Pro	49.89 / 37.69	41.83 / 38.65	18.68 / 7.57
	Pooled	53.48 / 43.05	42.23 / 37.59	24.39 / 18.31

Decision checks on common scores have zero mismatches for all three scorers. CISC and released 1.5B are checked by completing each run's unqueried scores; ThinkPRM is checked on the stored score vectors. Independent dense/VUD runs differ on 122 selected indices for ThinkPRM, 12 for CISC, and 0 for released 1.5B. No instance exceeds dense's call count.

Table 14 shows task-level variation. ThinkPRM time savings range from 15.66% (Granite/CRUXEval-O) to 74.09% (Qwen/ARC). CISC on GLM/ARC reduces calls from 43 to 27 but takes 6.21% longer. These timing runs acquire fresh scores, so their call counts can differ from the fixed-score evaluations in Figure 3(a,b).

The N8 formal matrix isolates broad model and benchmark coverage. A separate label-blind scaling run uses one Qwen3-8B generator and 128 fixed questions from each of GSM8K, MATH-500, and MMLU-Pro. Every row uses a prefix of the same N64 candidate pool. The fraction of calls removed relative to the matched zero-call-gated dense executor increases from 58.0% at N8 to 77.7% at N64 (Table 15).

Table 15: Accuracy (%) and SCORECERT calls on 384 Qwen3-8B questions. All budgets use prefixes of the same N64 pools and the shared zero-call gate.

N	GSM8K	MATH-500	MMLU-Pro	Overall	Gated	VUD	Call saved
8	93.75	90.63	71.88	85.42	896	376	58.0%
16	93.75	92.19	72.66	86.20	1,998	656	67.2%
32	94.53	93.75	71.09	86.46	4,433	1,128	74.6%
64	95.31	94.53	72.66	87.50	9,382	2,093	77.7%

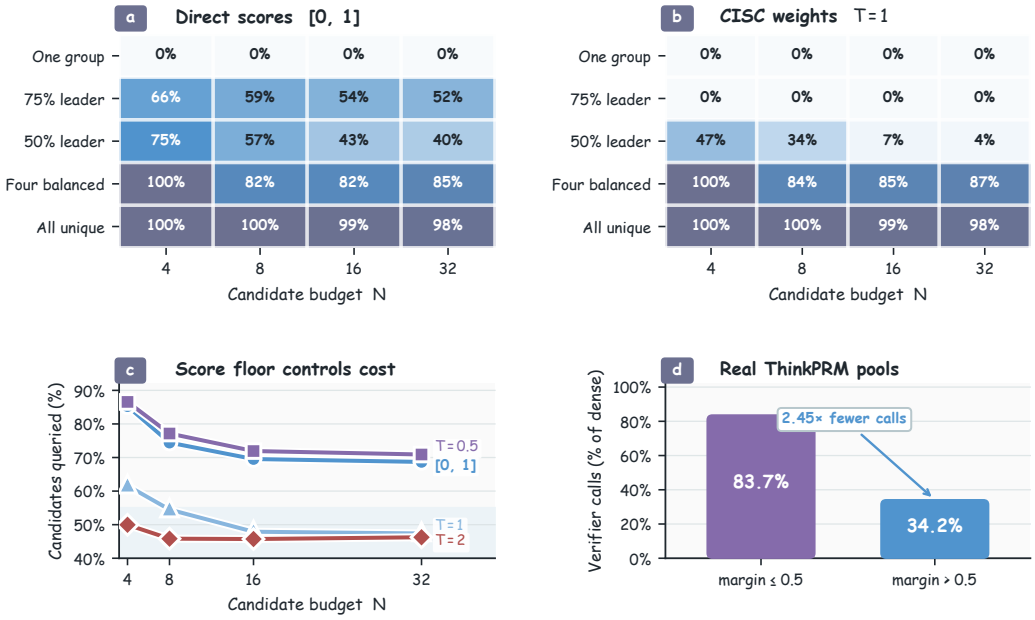


Figure 5: SCORECERT query cost (lower is better). (a–c) Synthetic pools: 2,000 per setting, with zero mismatches. Positive score floors and tighter bounds reduce queries; (c) averages four answer-structure profiles. (d) Calls on 197 multi-answer ThinkPRM pools, grouped into margin strata of 58 and 139 pools. Values are percentages of dense calls, not savings.

Table 16: FUTURECERT generation savings with CISC at $N_{\max} = 8$: 48 questions and three independent runs per path and generator (144 question-runs). Savings use pooled costs and exclude scoring; all completion checks pass.

Generator	Cand. saved	Tokens saved	Gen. wall saved	Stops	Endpoint checks
Granite	13.19%	11.16%	10.13%	106/144	294,912
Qwen	20.05%	17.10%	17.38%	127/144	465,152
GLM	14.32%	12.25%	8.91%	114/144	313,600
All	15.86%	13.50%	11.88%	347/432	1,073,664

Table 17 relates query cost to answer-group concentration. At N64, pools whose largest group contains at least 75% of valid candidates query 8.2% on average; pools below 50% query 42.7%.

Table 17: Queried fractions on multi-answer pools from the 384-question scaling study. Columns group pools by the largest answer group's share; ρ is its Spearman correlation with the queried fraction.

N	Pools	Overall	ρ	< 0.50	0.50–0.74	≥ 0.75
8	114	42.0%	–0.66	53.4%	49.1%	29.0%
16	127	32.8%	–0.80	49.2%	41.4%	17.4%
32	141	25.5%	–0.84	40.2%	35.1%	10.5%
64	149	22.4%	–0.87	42.7%	28.7%	8.2%

H ORACLE CERTIFICATE SIZE

Let w be the realized dense winner. For winner candidate i, define gain $d_i = s_i - \ell$; for candidate i in rival h, define reduction $r_i = u - s_i$. Sort each list descending and let $D_w(k)$ and $R_h(k)$ be prefix sums. For a proposed number k_w of winner queries, the largest possible lower bound is $|C_w|\ell + D_w(k_w)$. The fewest rival queries are the smallest k_h whose reduced upper bound satisfies the appropriate strict or weak tie inequality. Minimizing $k_w + \sum_{h \neq w} k_h$ over k_w gives the following result.

Proposition 1 (Hindsight-minimal terminal certificate)**.** *For a realized fixed-pool score vector with winner w, the minimum number of revealed scores among all terminal transcripts satisfying Eq. (6) is*

$$\min_{0 \le k_w \le |C_w|} \left(k_w + \sum_{h \neq w} k_h^*(k_w) \right), \tag{10}$$

where $k_h^*(k_w)$ is the least k such that $|C_w|\ell + D_w(k_w) > |C_h|u - R_h(k)$ for $h < w$, and the least k satisfying the corresponding weak inequality for $h > w$; set $k_h^*(k_w) = +\infty$ if no such prefix exists.

Proof. For fixed query counts, an exchange argument shows that revealing the largest winner gains and largest rival reductions maximizes certificate slack; any other subset can be swapped element by element without worsening feasibility. Each rival then separates given the chosen winner lower bound, so taking its least feasible prefix is optimal. Enumerating k_w proves the formula. We also validate it against exhaustive subset enumeration on randomized small instances. This is a hindsight lower bound, not an online query policy. □

I NUMERICAL VALIDATION

Numerical precision and overhead. Dense and VUD use exact rational arithmetic on decimal scores. On 39,051 tie-enriched test pairs, this removes all 6,594 consistency failures of the earlier floating-point implementation. On the original 720 ThinkPRM pools and the timing and multi-seed checks, winners, query schedules, and call counts are unchanged; model timing measurements are retained. Median selection time on 3,000 pools rises from 53.18 to 528.57 microseconds, adding about 0.48 ms.

Separately, 540 released-1.5B comparisons across five batch sizes and three orders show a maximum probability change of 0.00390625. This is runtime score variability, not a fixed-score certificate error.

Implementation checks. Certificate tests include 287,220 fixed-pool states, 17,000 randomized pools, and 372,555 future completions across five score intervals, with zero violations. Joint-uncertainty results appear in Appendix B. All eight all-invalid pools return the same null output without scoring. On four near-tied CISC questions, 36,864 candidate-order perturbations produce 27 winner changes, confirming the need for a fixed tie rule.