ONE PRICE, MANY FUTURES: SHARED-PRICE OPTION INDICES FOR POST-PREFIX COMPUTE ALLOCATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Inference-time compute is often allocated before reasoning begins. A shared prefix changes the decision: the realized state reveals which requests may benefit from more work, including cases that recover only at a distant endpoint. All continuation options also share a batch budget, so their predicted values must be comparable. We formulate this problem as state-conditioned selection among complete continuation options. *Shared-Price Option Indices* (SPOI) use the realized prefix state to value each option and compare all requests under one common price. Our analysis shows when state information improves allocation, when myopic stopping misses delayed recovery, and when heterogeneous options can share that price. Two complementary estimators trade off cardinal value preservation against a bounded learning target, with validation selecting between them. In a scale-selection study specified without test-based selection across four backbones, two tasks, and nine matched budgets, SPOI outperforms query-only routing in most settings and matches or improves on independently calibrated option scores. Frozen SPOI schedules also produce physical savings: recurrent active-batch execution on A800 reaches $2.67\times$ speedup with 0.40% mean control overhead, while standard KV-cache execution reaches up to $2.59\times$.

1 INTRODUCTION

Reasoning changes what future computation is worth. At the same early depth, one request may have stabilized while another can recover only after an unfavorable intermediate state. Query-only routing cannot distinguish them, and myopic stopping loses the delayed recovery. Latent recurrent models such as Huginn (Geiping et al., 2025) expose this decision at the realized prefix state.

Most inference-time allocation selects samples, tokens, or depth from input features (Damani et al., 2025; Snell et al., 2025; Zhai et al., 2026; Wan et al., 2026). After a shared prefix, the state contains new evidence and the decision concerns a complete continuation. Because all options draw from one batch budget, they must be compared on a common value-per-compute scale. A threshold on this scale acts as a shadow price. Option-specific percentiles break the comparison because the same score can represent different reservation values at different depths.

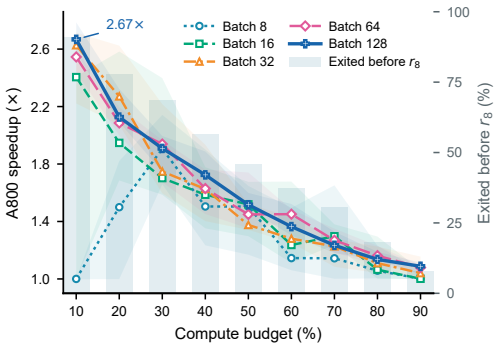

Figure 1: Huginn active-batch execution on A800. Curves compare compacted and uncompressed replay of identical SPOI schedules; bands are hierarchical 95% intervals. Bars show requests exiting before r_8.

Shared-Price Option Indices (SPOI) use the realized prefix state to estimate complete continuation values and compare every request–option pair under one common price. This retains a distant end-

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.

point when the intermediate endpoint is unattractive and directs extra compute to the most valuable available continuations.

Existing methods either allocate depth before the relevant prefix state is observed or make local stopping decisions afterward (Graves, 2016; Dehghani et al., 2019; Banino et al., 2021; Raposo et al., 2024; Bae et al., 2025; Wang et al., 2023; Snell et al., 2025; Damani et al., 2025; Zhai et al., 2026). CLEAR provides global shadow-price allocation (Wan et al., 2026); our setting supplies it with post-prefix option values on a common scale. Early-exit and serving systems determine how selected exits save work through batching and cache management (Schuster et al., 2022; Elhoushi et al., 2024; Del Corro et al., 2023; Dai et al., 2024; Yu et al., 2022; Kwon et al., 2023; Agrawal et al., 2024). Our execution audit fixes the selected schedules and measures the resulting savings.

Scope and contributions. We study three sources of allocation loss: missing state information, myopic continuation horizons, and incomparable option scores. SPOI addresses them with state-conditioned values for complete options and one budgeted assignment rule. Counterfactual endpoints provide offline supervision and a matched-budget Oracle ceiling; at evaluation, SPOI observes the prefix state. Our analysis states when additional state helps, when myopic stopping misses delayed recovery, and when heterogeneous options can share one price. Eight backbone–task settings measure allocation under matched compute budgets, while recurrent and KV-cache audits test whether frozen schedules preserve selected outputs and reduce physical work on real hardware.

2 Post-prefix compute allocation as option selection

The decision has two stages. Every request first receives the same prefix computation. The resulting latent state then determines whether the request should stop or purchase one of several longer continuations. The first stage reveals information; the second allocates a limited pool of additional compute. This separation makes the object of prediction concrete: the controller estimates how much utility each available continuation adds to the state already obtained.

Shared prefix and continuation options. Consider a batch of N requests. Request i has query x_i and reaches state h_i after r_0 latent steps. Because every candidate policy has already paid for this prefix, the remaining choice concerns only future computation. The available terminal depths are

$$\mathcal{D} = \{r_0, d_1, \ldots, d_K\}. \tag{1}$$

Let $U_i(d)$ denote utility at depth d and $C_i(d)$ its measured compute cost. Continuation option o extends the request from r_0 to d_o. Its benefit and price are therefore measured relative to stopping now:

$$G_{io} = U_i(d_o) - U_i(r_0), \qquad \Delta C_{io} = C_i(d_o) - C_i(r_0) > 0. \tag{2}$$

The options are *non-local*. A request may move directly from r_0 to d_2 even when the nearer endpoint d_1 is unattractive. In particular, $U_i(d_1) - U_i(r_0) < 0$ can coexist with $U_i(d_2) - U_i(r_0) > 0$. Treating the first increment as a prerequisite would discard this delayed recovery.

Batch-constrained objective. A policy π maps the observed state to a terminal depth. The batch must maximize expected utility while keeping average compute below B:

$$V(B) = \max_{\pi} \ \mathbb{E}[U_i(\pi(x_i, h_i))] \quad \text{s.t.} \quad \mathbb{E}[C_i(\pi(x_i, h_i))] \le B. \tag{3}$$

The multiplier $\lambda \ge 0$ assigns a price to one unit of compute. Once this price is fixed, each request can compare the utility of a depth with its compute charge independently:

$$d_i^*(\lambda) = \arg\max_{d \in \mathcal{D}} \{U_i(d) - \lambda C_i(d)\}. \tag{4}$$

For a single continuation, this comparison reduces to its utility gain per unit of additional compute, or true reservation price,

$$q_{io} = \frac{G_{io}}{\Delta C_{io}}. \tag{5}$$

Option o is worthwhile in isolation exactly when $q_{io} > \lambda$. The same λ governs every request and every depth, so their predicted reservation prices must be comparable across option families. Independent within-depth ranks do not provide this comparability. The controlled evaluation records every candidate endpoint for each held-out request; these outcomes define a counterfactual Oracle upper bound against which the learned assignment can be measured.

Information, horizon, and coordinate losses. Three losses follow from the available information, action set, and value coordinate.

1. *Information loss:* a query-only policy cannot use information created during the shared prefix.
2. *Horizon loss:* a one-step action set cannot purchase a delayed-recovery continuation whose intermediate outcome is locally unattractive.
3. *Coordinate distortion:* option-wise score maps can reverse the cross-option order required by the common λ in equation 4.

The first two arise from the available information and action space. The third arises from the coordinate on which otherwise valid actions compete.

3 SHARED-PRICE OPTION INDICES

SPOI separates index estimation from budget allocation. It summarizes the state at r_0 and estimates each continuation option's terminal gain per unit cost. Cardinal-SPOI regresses this value directly; Rank-SPOI learns a bounded pooled coordinate. Validation selects the estimator before held-out evaluation. A scalar price then produces the batch assignment, and a residual upgrade rule uses compute left by discrete option costs. The same frozen indices support every budget; only the clearing price changes.

3.1 STATE-CONDITIONED BUNDLE READOUT

The same latent state can support a short continuation and a distant continuation differently. The readout therefore shares its state representation across options but produces a distinct score for each bundle. At boundary r_0, it receives only causal summaries of the current and preceding latent states and outputs

$$s_{io} = s_\theta(h_i, o). \tag{6}$$

The input contains seven causal groups: the last and mean recurrent states, first- and second-order temporal differences, and the pre-coda summary available at the boundary. A shared projection maps group g to $e_{ig} \in \mathbb{R}^p$. Continuation o uses query a_o to weight these groups:

$$\alpha_{iog} = \frac{\exp(a_o^\top e_{ig}/\sqrt{p})}{\sum_{g'} \exp(a_o^\top e_{ig'}/\sqrt{p})}, \qquad v_{io} = \sum_g \alpha_{iog} e_{ig}. \tag{7}$$

A compact vector m_i adds norms, cosine changes, and distributional summaries of state motion. An option-specific head maps $[v_{io}; m_i]$ to s_{io}. The shared projection pools statistical strength, while the option queries can associate different state evidence with d_1 and d_2. Because all features precede the continuation-specific coda, they are available before additional work is assigned. The resulting option index values the full path to its terminal depth and can favor a distant recovery even when the nearer endpoint has little value.

3.2 COMPARABLE OPTION VALUES

A raw controller score has meaning only relative to its training target. Both SPOI estimators use one coordinate across option families. Cardinal-SPOI applies a shared affine standardization to q_{io}, minimizes Huber loss, and maps predictions back to the reservation-price scale:

$$\widehat{G}_{io}^{\mathrm{card}} = \widehat{q}_{io}\, \Delta C_{io}. \tag{8}$$

This coordinate preserves the original unequal-cost surplus geometry but exposes regression to variation in the cardinal target.

Rank-SPOI bounds the target while retaining a common threshold coordinate. If option o used its own empirical CDF F_o, then $F_o(q_{io})$ would record position within that option but lose comparisons between options. Rank-SPOI instead forms one empirical distribution over all request–option pairs,

$$F_{\mathrm{pool}}(t) = \frac{1}{NK} \sum_{i=1}^{N} \sum_{o=1}^{K} \mathbf{1}\{q_{io} \le t\}, \tag{9}$$

and uses the common target

$$z_{io} = F_{\text{pool}}(q_{io}). \tag{10}$$

Every option now passes through the same monotone map. The target z normalizes the range without changing the weak order of reservation prices across depths. Training combines pointwise regression with a listwise term that emphasizes the high-value region most likely to receive compute:

$$\begin{aligned}\mathcal{L}_{\text{SPOI}} = \ & \text{Huber}(s_\theta, z) \\ & + \beta H(\text{softmax}(z/\tau_t), \text{softmax}(s_\theta/\tau_s)) \,.\end{aligned} \tag{11}$$

The Huber term anchors each prediction to the common target. The cross-entropy term aligns the relative mass of the predicted and target upper tails. Calibration applies the same pooling principle to model scores. If $\mathcal{A}$ denotes the fit-set request–option pairs, the empirical map is

$$\widehat{T}_{\text{pool}}(s) = \frac{1}{|\mathcal{A}|} \sum_{(j,o') \in \mathcal{A}} \mathbf{1}\{s_{jo'} \le s\}. \tag{12}$$

The map interprets every option against one fitted score distribution. Let $\widehat{z}_{io} = \widehat{T}_{\text{pool}}(s_{io})$. The allocator converts this calibrated rank coordinate into a cost-weighted pseudo-gain,

$$\widetilde{G}_{io} = \widehat{z}_{io}\, \Delta C_{io}. \tag{13}$$

Thus $\widetilde{G}_{io}/\Delta C_{io} = \widehat{z}_{io}$ is comparable across options. The quantity $\widetilde{G}_{io}$ is a pseudo-gain on the transformed rank scale; recovering the original gain scale would additionally require an inverse-CDF map. The transformed coordinate preserves common-threshold comparisons, although a nonlinear map may change unequal-cost multi-option surplus orderings.

Let $\overline{G}_{io}$ denote the gain supplied by the validation-selected estimator: $\widehat{G}^{\text{card}}_{io}$ for Cardinal-SPOI and $\widetilde{G}_{io}$ for Rank-SPOI. Selection maximizes mean validation utility over the fixed budget grid, after which the chosen estimator is frozen for every test request and budget. Across the eight settings and five seeds, validation selects Cardinal-SPOI 25 times and Rank-SPOI 15 times.

3.3 Global shadow-price allocation

Define $\overline{U}_i(r_0) = 0$ and $\overline{U}_i(d_o) = \overline{G}_{io}$. Each request then chooses the depth with the largest estimated surplus after paying the common compute price:

$$\widehat{d}_i(\lambda) = \arg\max_{d \in \mathcal{D}} \left\{ \overline{U}_i(d) - \lambda C_i(d) \right\}, \tag{14}$$

Stopping at r_0 has zero incremental gain and cost. Raising λ can only make costly depths less attractive, so aggregate selected cost is monotone non-increasing. Bisection therefore finds the smallest price whose assignment respects the budget. Because depths have discrete costs, this assignment can leave residual compute that is too small for some upgrades but sufficient for others.

The residual queue compares every feasible replacement of a request's current depth d_i by a deeper endpoint d'. Its key is the marginal utility obtained per additional unit of compute:

$$\rho_i(d_i, d') = \frac{\overline{U}_i(d') - \overline{U}_i(d_i)}{C_i(d') - C_i(d_i)}. \tag{15}$$

The numerator is the predicted benefit of replacing d_i with d', and the denominator is the extra compute consumed by that replacement. The allocator accepts the largest positive ρ_i that fits, updates the chosen request, and repeats until no positive upgrade fits the remaining budget. Direct $r_0 \to d_K$ candidates stay in the queue, so this discrete correction preserves delayed recovery rather than forcing the allocation through the intermediate depth. All baselines use the same price search, residual rule, and realized budget constraint; SPOI changes only the coordinate supplied to them.

3.4 Validation-selected estimator

The cardinal and rank targets resolve different statistical errors. SPOI selects their common-price estimator on validation data:

1. Train Cardinal-SPOI and Rank-SPOI on the same fit/validation split with matched architecture and initialization.
2. Measure each estimator's mean validation utility over the frozen nine-budget grid.
3. Retain the higher-utility estimator and discard the other head before held-out evaluation.

Selection occurs once per training seed. It never uses test utilities, and inference retains one head with the same latency as either component.

3.5 Frozen-schedule active-batch execution

The systems audit materializes the depth assignments from equation 14 as nested active sets and compacts the replay tensor at each boundary. Requests assigned r_0 exit immediately; only requests assigned at least d_1 enter the next recurrent block, and only deeper assignments remain thereafter:

$$h_i(r_0) \rightarrow \text{indices} \rightarrow \text{price clearing} \rightarrow \text{tensor compaction} \rightarrow \text{continuation/coda}. \tag{16}$$

Writing $A_k = \{i : \widehat{d}_i \geq d_k\}$, the replay gathers the hidden states and cache entries indexed by A_k, evaluates the block $d_{k-1} \rightarrow d_k$ on this compact tensor, and scatters completed outputs back to their original batch positions. The nested sets $A_{k+1} \subseteq A_k$ ensure that each request executes exactly its assigned continuation. Boundary assignment before coda evaluation prevents a full-batch probe from consuming the work saved by early exits.

Gold continuation outcomes construct offline targets only; held-out allocation uses the causal state available at the shared prefix. Appendix Algorithm 1 gives the complete optimization and frozen-replay procedure.

4 Why SPOI has this form

Once the shared prefix is sunk, a state-conditioned policy contains every query-only decision and can exploit states with different continuation values (Proposition 1). Its action set must also retain distant endpoints: when $4 \rightarrow 8$ has non-positive surplus but $4 \rightarrow 16$ recovers, non-local selection avoids the myopic horizon loss quantified by Proposition 2.

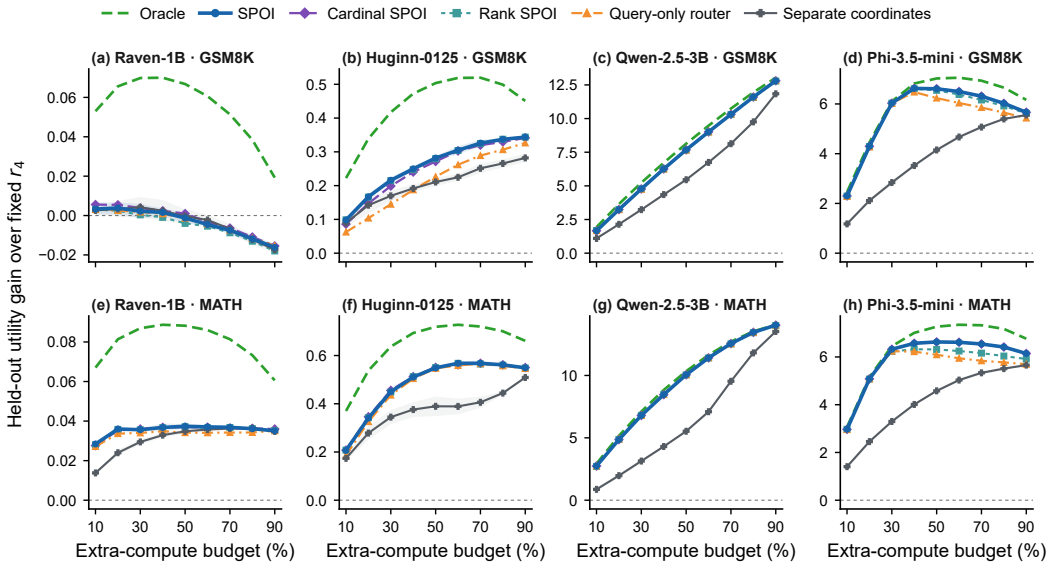

Figure 2: Held-out compute–utility frontiers. Curves average five seeds; shaded regions are 95% intervals for SPOI and Separate coordinates. Cardinal and rank curves expose the two SPOI estimators; the blue curve selects between them on validation. Query-only is a supervised comparator. Every panel obeys the same aggregate TFLOP target.

All continuation options are evaluated under one price. A common threshold reproduces $q > \lambda$ only when option families share the same monotone score map (Theorem 3); option-wise percentiles do not. Cardinal-SPOI retains the original surplus geometry, while Rank-SPOI provides a bounded common map that guarantees threshold compatibility but not every unequal-cost surplus order. Validation resolves this statistical tradeoff before Equation 14 clears the selected coordinate. Appendix A formalizes the distinction.

State conditioning distinguishes continuation prospects, non-local bundles retain delayed recoveries, and validation selects a common price estimator. Together they determine whether a request continues and which endpoint it selects.

Selected depths determine physical work. With fixed-time ratio $\rho = T_0/(NK)$, active fractions a_8, a_{16}, and normalized control and compaction overhead h,

$$\text{Speedup} \approx \frac{\rho + 2}{\rho + a_8 + a_{16} + h}, \qquad \rho = \frac{T_0}{NK}. \tag{17}$$

Small active fractions remove recurrent work at low budgets; as a_8 and a_{16} approach their dense values, speedup contracts toward one. Batch amortization reduces h, while counterfactual tables and selected-exit agreement separate schedule quality from physical realization. Appendix A gives the proofs.

5 Experiments

Decision experiments cover four backbones: Raven-1B (McLeish et al., 2025), Huginn-0125, Qwen2.5-3B-Instruct (Qwen Team, 2024), and Phi-3.5-mini-instruct (Abdin et al., 2024). We evaluate them on GSM8K (Cobbe et al., 2021) and numeric MATH (Hendrycks et al., 2021), using disjoint fit, validation, and once-only test partitions across five seeds. Each held-out request supplies utilities and measured costs at depths $\{4, 8, 16\}$, so every policy selects among identical endpoints after the same shared prefix. Nine budgets span 10%–90% of available incremental compute. Cardinal-SPOI, Rank-SPOI, their validation-selected estimator, Separate coordinates, a supervised query-only router, prompt length, random priority, and Oracle use the allocator in Equation 14 and the same realized aggregate cost ceiling. Only Oracle observes all candidate outcomes. The scale-selection extension followed the original rank-only analysis; its architecture, validation rule, and query-only comparator were fixed before the corresponding test results were opened.

Matched-budget Oracle-gap recovery. Figure 2 compares compute–utility frontiers under the same realized cost ceiling. SPOI exceeds query-only in 55 of 72 backbone–task–budget cells, Separate coordinates in 66, length in 71, and random priority in all 72, with no significant loss to Separate. Figure 3 normalizes these gains by the remaining matched-budget Oracle gap:

$$\text{Recovery} = \frac{U_{\text{SPOI}} - U_{\text{Separate}}}{U_{\text{Oracle}} - U_{\text{Separate}}}. \tag{18}$$

Across 360 seed–budget observations, 58 of 72 aggregated cells improve in every seed, including all 36 Qwen/Phi cells; only one declines throughout. Table 1 summarizes each setting.

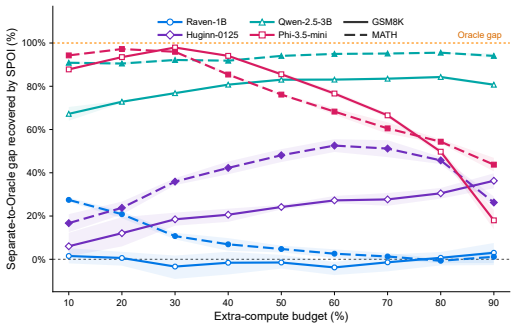

Figure 3: Oracle-gap recovery across eight held-out settings. Curves average five seeds and bands show 95% intervals; Raven–GSM8K remains near zero.

State and delayed recovery. Delayed-recovery trajectories occur when $4 \to 8$ is locally unattractive but $4 \to 16$ recovers. They account for 43/1,024 Raven–GSM8K requests, 69/1,024 Raven–MATH requests, and 396/1,024 Huginn–GSM8K requests. In the recurrent state-information diagnostics, matched state controls exceed their query-only counterparts in 54/57 budget cells and length-only controls in 55/57. The realized state and the non-local action set thus address distinct sources of allocation loss.

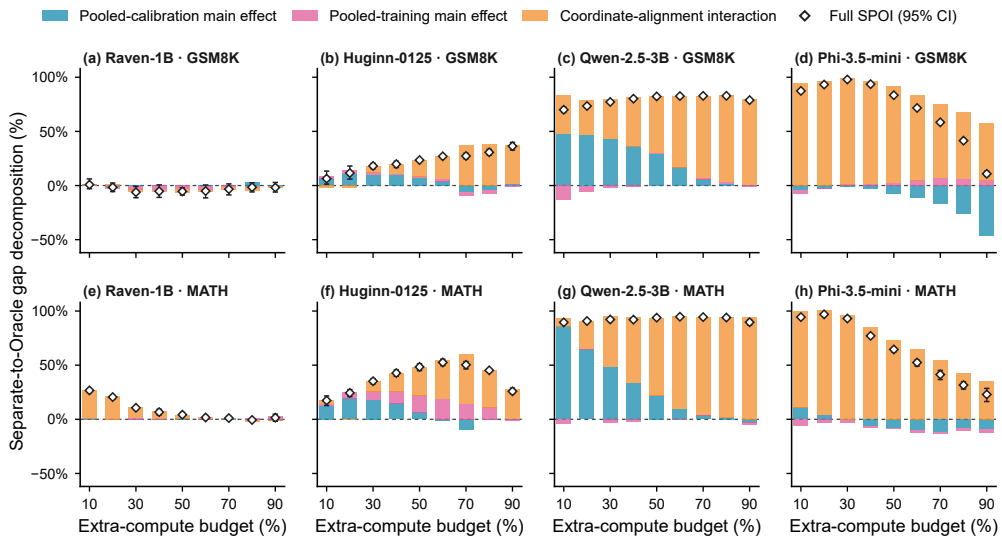

Figure 4: Frozen 2×2 decomposition across eight settings and nine budgets. Bars give the main effects and interaction; diamonds give the full SPOI effect with 95% intervals.

Table 1: Mean Oracle-gap recovery (%) over Separate coordinates.

Backbone	GSM8K	MATH
Raven-1B	−0.5	8.4
Huginn-0125	22.7	38.2
Qwen-2.5-3B	79.2	93.2
Phi-3.5-mini	74.4	75.1

Coordinate alignment and estimator selection. Figure 4 crosses pooled versus option-wise targets with pooled versus option-wise calibration. Their interaction dominates all 18 Phi cells and 12 of 18 Qwen cells, with smaller aligned effects on Huginn and Raven–MATH; Raven–GSM8K instead shows nearly coincident pooled and Separate frontiers. Validation consequently selects Cardinal-SPOI for every Qwen/Phi seed and Rank-SPOI for four of five seeds in each Huginn task, while Raven mixes the two estimators. Across all 72 setting–budget cells, 58 improve over Separate in every seed; paired intervals yield 57 significant gains and no significant loss. Appendix Table 4 reports the per-setting comparison.

Allocation geometry. Theorem 3 predicts that option-wise coordinates matter when they change cross-option choices. Across 72 setting–budget cells, the fraction of Oracle choices changed by option-wise coordinates correlates with recovered gap ($\rho_s = 0.50$; Figure 5). The association remains positive within every backbone (0.41–0.85) and within each task. Coordinate disagreement therefore serves as a practical indicator of recoverable opportunity.

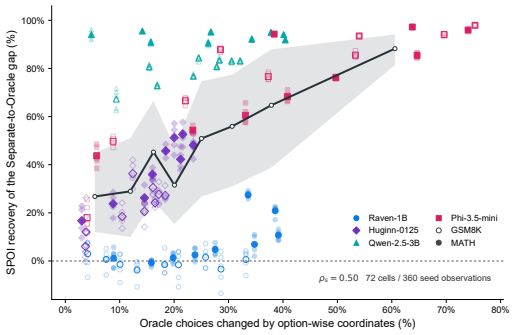

Figure 5: Coordinate error predicts recoverable opportunity. Faint points are 360 seed observations, larger points are 72 cell means, and the black curve with its bootstrap band is an eight-bin empirical trend.

Admission and continuation errors. Figure 6 decomposes this disagreement into the decisions that create it. Total disagreement peaks at an interior budget in every setting: 0.74–0.75 of Phi routes differ near budget 0.3, while Qwen peaks at 0.32–0.40 between budgets 0.4 and 0.6. Admission flips dominate when compute is scarce because the active boundary separates r_4 from any continuation. As the budget rises, continua-

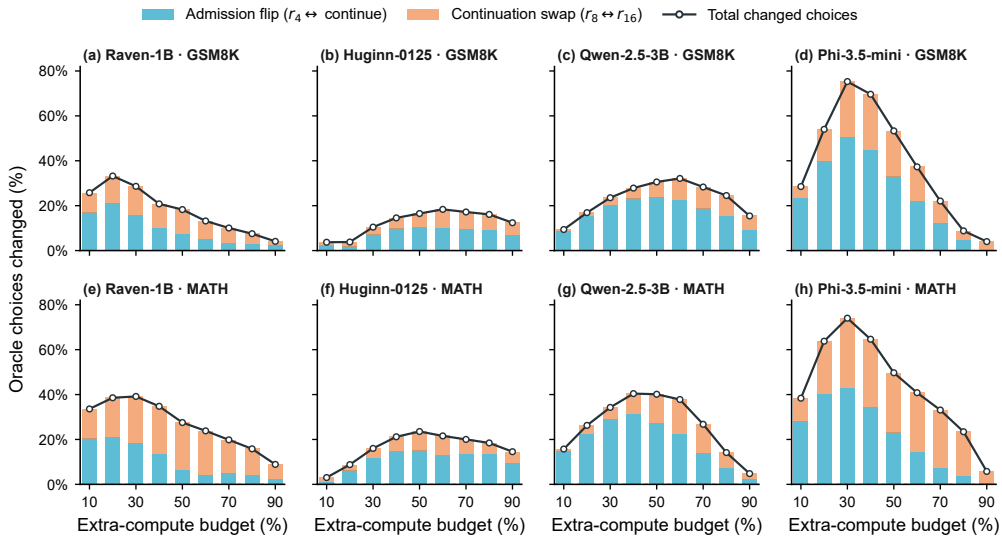

Figure 6: Oracle routing errors induced by option-wise coordinates. Stacked bars separate admission flips from continuation swaps; the black curve is their total.

tion swaps account for a larger share. At budget 0.9, they comprise 0.038 of the 0.040 Phi–GSM8K disagreement and 0.052 of the 0.058 Phi–MATH disagreement. A shared price coordinate therefore changes both whether a request continues and which non-local future it purchases.

Appendix Figure 7 shows the corresponding depth choices. As budget rises, request mass moves from r_4 through r_8 or directly to r_{16} along backbone- and task-specific paths. These transitions show the non-local $4 \to 16$ option hidden by average depth. Appendix B also contains a held-out 136-parameter linear probe and the full state-information results.

Physical realization of frozen schedules. Figure 1 covers recurrent A800 execution at batch sizes 8–128 and nine budgets. Each paired comparison fixes the selected-depth schedule and varies only compacted versus uncompressed execution. All 45 point estimates exceed one, and 41 hierarchical 95% intervals lie strictly above one. Maximum speedup reaches $2.67\times$, with 0.40% mean controller-plus-compaction overhead. As compactable mass falls from 91.0% to 8.0%, speedup approaches one in accordance with Equation 17. Table 2 reports absolute execution measurements: controller-plus-compaction overhead falls from 0.82% at batch 8 to 0.19% at batch 128, where throughput spans 20.1–49.3 requests/s and peak memory remains 15.44 GB. Appendix D gives the corresponding post-hoc fit.

Table 2: Absolute recurrent A800 execution. Ranges cover nine budgets; overhead is relative to same-schedule replay. Each row contains 450 timings per method.

Batch	Latency (ms)	Throughput (req./s)	Overhead	Peak GB
8	133.5–449.1	17.8–59.9	0.82%	7.23
16	367.1–881.1	18.2–43.6	0.47%	7.80
32	661.1–1667.1	19.2–48.4	0.29%	8.94
64	1356.1–3197.8	20.0–47.2	0.24%	10.95
128	2595.8–6362.1	20.1–49.3	0.19%	15.44

The KV-cache audit separates batch amortization from removable layer work. At budget 0.1, Qwen speedup rises from $1.152\times$ at batch 32 to $1.856\times$ at batch 128, while Phi reaches $2.595\times$; near-full-depth schedules approach parity at budget 0.9. Every cell preserves 1.000 selected-exit token agreement, and an exit-diverse RTX 3080 Ti regression likewise records 128/128 agreement across depths 4/8/16. Appendix Table 9 gives the confidence intervals and complete matrix.

6 DISCUSSION AND LIMITATIONS

Table 3 reports fixed-depth native generation before learned routing. The intermediate depth has the highest EM point estimate; extending depth from 24 to 32 increases wall time by 35% while reducing the point estimate from .402 to .379. Low cap rates indicate that truncation does not explain this pattern.

Table 3: Fixed-depth Huginn native generation on a 256-example development split with a 512-token cap. Time is mean wall time per request.

Depth	EM	95% CI	Cap	Time (s)
16	.387	[.329, .448]	3.5%	6.08
24	.402	[.344, .463]	1.6%	8.35
32	.379	[.322, .440]	1.6%	11.27

Free-running generation introduces an additional statistical layer. Earlier routing decisions change emitted tokens and therefore later states, while each realized trajectory exposes only one path-dependent terminal outcome. Learning comparable continuation values in this regime calls for randomized or on-policy route collections, intermediate verifier signals, semantic outcome models, routing-aware distillation, or uncertainty-aware calibration. Given these scores on a shared monotone coordinate, SPOI allocates the resulting options under a common budget.

A free-running route-learning study would learn that interface from path-dependent trajectories, freeze the resulting policy, and pass it through the same allocation and execution stages. Continuous serving introduces another orthogonal systems layer: live arrivals additionally couple route refresh with dynamic batching, KV-cache migration, and queue-dependent latency.

Figures 4 and 6 connect rank-coordinate alignment to the decisions it changes. The interaction in Figure 4 becomes prominent where multiple continuation families compete, while Figure 6 shows a common interior-budget concentration and a budget-dependent shift from admission to continuation choice. The common coordinate thus resolves a sequence of active boundaries rather than applying a uniform correction to all requests. The scale-selection results add a second distinction: even after options share one axis, finite-sample prediction determines whether its cardinal or bounded representation generalizes better.

A post-hoc calibration audit holds the frozen Qwen/Phi scores fixed and replaces their pooled rank coordinate with its pooled inverse quantile (Appendix Table 5). Qwen assignments remain essentially unchanged, whereas Phi exhibits a finite cardinal-calibration effect concentrated in the choice between continuation depths.

Validation choices follow this pattern. Cardinal-SPOI is chosen for every Qwen/Phi seed, whereas four of five seeds select Rank-SPOI in each Huginn task. Pooled ranks regularize target scale, while cardinal prices preserve unequal-cost surplus comparisons.

7 CONCLUSION

One price can allocate compute across many continuation options once their values are comparable. SPOI uses the realized prefix state to estimate complete option values, retains distant endpoints so delayed recovery remains available, and compares heterogeneous options on a common price scale. Our analysis shows when state information helps, when myopic stopping loses delayed recovery, and when heterogeneous options can share one price. Cardinal prices preserve unequal-cost surplus comparisons, while pooled ranks limit target variation; validation selects between them. The value estimator and executor can change while retaining the same budgeted allocator. Denser trajectory supervision and routing-aware serving can extend SPOI to free-running inference.

AI USE STATEMENT

Generative AI tools were used to assist with code implementation, experiment orchestration, mathematical exposition, and manuscript drafting. All generated claims, derivations, code, and numerical results were checked by the authors, and the authors take responsibility for the final content.

Reproducibility statement

The artifact contains trajectory construction, frozen train/validation/test protocols, controller training, the price allocator, strong-baseline extensions, active-batch benchmarks, per-seed tables, and scripts generating every figure from machine-readable data. Appendix A gives the formal proofs, and Appendix C records the data, controller, compute, and timing protocols.

References

Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, et al. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*, 2024. URL https://arxiv.org/abs/2404.14219.

Amey Agrawal, Nitin Kedia, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, Bhargav Gulavani, Alexey Tumanov, and Ramachandran Ramjee. Taming Throughput-Latency tradeoff in LLM inference with Sarathi-Serve. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pp. 117–134. USENIX Association, July 2024. URL https://www.usenix.org/conference/osdi24/presentation/agrawal.

Sangmin Bae, Yujin Kim, Reza Bayat, Sungnyun Kim, Jiyoun Ha, Tal Schuster, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Aaron Courville, and Se-Young Yun. Mixture-of-recursions: Learning dynamic recursive depths for adaptive token-level computation. *arXiv preprint arXiv:2507.10524*, 2025.

Andrea Banino, Jan Balaguer, and Charles Blundell. Pondernet: Learning to ponder. *arXiv preprint arXiv:2107.05407*, 2021.

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

Yinwei Dai, Rui Pan, Anand Iyer, Kai Li, and Ravi Netravali. Apparate: Rethinking early exits to tame latency-throughput tensions in ML serving. In *Proceedings of the ACM SIGOPS 30th Symposium on Operating Systems Principles*, pp. 607–623, 2024. doi: 10.1145/3694715.3695963.

Mehul Damani, Idan Shenfeld, Andi Peng, Andreea Bobu, and Jacob Andreas. Learning how hard to think: Input-adaptive allocation of LM computation. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/ff414825df833edb8b1839e3d5d495e9-Abstract-Conference.html.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. In *International Conference on Learning Representations*, 2019.

Luciano Del Corro, Allie Del Giorno, Sahaj Agarwal, Bin Yu, Ahmed Awadallah, and Subhabrata Mukherjee. SkipDecode: Autoregressive skip decoding with batching and caching for efficient LLM inference. *arXiv preprint arXiv:2307.02628*, 2023.

Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, Ahmed Aly, Beidi Chen, and Carole-Jean Wu. LayerSkip: Enabling early exit inference and self-speculative decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pp. 12622–12642, 2024. doi: 10.18653/v1/2024.acl-long.681.

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-1380.

Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016.

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 Datasets and Benchmarks Track*, 2021. URL https://openreview.net/forum?id=7Bywt2mQsCe.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pp. 611–626, 2023. doi: 10.1145/3600006.3613165.

Sean McLeish, Ang Li, John Kirchenbauer, Dayal Singh Kalra, Brian R. Bartoldson, Bhavya Kailkhura, Avi Schwarzschild, Jonas Geiping, Tom Goldstein, and Micah Goldblum. Teaching pretrained language models to think deeper with retrofitted recurrence. *arXiv preprint arXiv:2511.07384*, 2025. URL https://arxiv.org/abs/2511.07384.

Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024. URL https://arxiv.org/abs/2412.15115.

David Raposo, Sam Ritter, Blake Richards, Timothy Lillicrap, Peter Conway Humphreys, and Adam Santoro. Mixture-of-depths: Dynamically allocating compute in transformer-based language models. *arXiv preprint arXiv:2404.02258*, 2024.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Q. Tran, Yi Tay, and Donald Metzler. Confident adaptive language modeling. In *Advances in Neural Information Processing Systems*, volume 35, pp. 17456–17472, 2022.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=4FWAwZtd2n.

Xu Wan, Speed Zhu, Jianwei Cai, Guang Chen, Ximing Huang, Wiggin Zhou, and Mingyang Sun. The shadow price of reasoning: Economic perspective on optimal budget allocation for LLMs. In *International Conference on Machine Learning*, 2026.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. 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.

Gyeong-In Yu, Joo Seong Jeong, Geon-Woo Kim, Soojeong Kim, and Byung-Gon Chun. Orca: A distributed serving system for Transformer-Based generative models. In *16th USENIX Symposium on Operating Systems Design and Implementation*, pp. 521–538, 2022.

Zhiyuan Zhai, Bingcong Li, Bingnan Xiao, Ming Li, and Xin Wang. Adaptive test-time compute allocation for reasoning LLMs via constrained policy optimization. *arXiv preprint arXiv:2604.14853*, 2026.

A Formal results and proofs

Let the option set $\mathcal{O} = \{0, 1, \ldots, K\}$ be finite. At compute price λ, let $Y_o = G_o - \lambda \Delta C_o$ be option o's net surplus and include stopping as $Y_0 = 0$. Assume every Y_o is integrable. For an information set $\mathcal{F}$, write $m_o^{\mathcal{F}} = \mathbb{E}[Y_o \mid \mathcal{F}]$ and define

$$J(\mathcal{F}) = \mathbb{E}\left[\max_{o \in \mathcal{O}} m_o^{\mathcal{F}}\right]. \tag{19}$$

The maximum is measurable because $\mathcal{O}$ is finite. A fixed tie-breaking rule therefore produces an $\mathcal{F}$-measurable maximizing action.

Proposition 1 (Value of reasoning-state information)**.** *Let $\mathcal{F}_x = \sigma(x)$ and $\mathcal{F}_h = \sigma(x, h)$ with $\mathcal{F}_x \subseteq \mathcal{F}_h$. If the shared prefix cost is sunk, then $J(\mathcal{F}_h) \geq J(\mathcal{F}_x)$. Let A_x be any $\mathcal{F}_x$-measurable maximizer of $m_o^{\mathcal{F}_x}$. The inequality is strict if*

$$\mathbb{P}\Big(\max_o m_o^{\mathcal{F}_h} > m_{A_x}^{\mathcal{F}_h}\Big) > 0. \tag{20}$$

Proof. Because A_x is $\mathcal{F}_x$-measurable and $\mathcal{F}_x \subseteq \mathcal{F}_h$, it is also $\mathcal{F}_h$-measurable. Define the refined-information payoff of this coarse action by

$$m_{A_x}^{\mathcal{F}_h} = \sum_{o \in \mathcal{O}} \mathbf{1}\{A_x = o\} m_o^{\mathcal{F}_h}. \tag{21}$$

For each o, the indicator $\mathbf{1}\{A_x = o\}$ is $\mathcal{F}_x$-measurable. The tower property gives

$$\mathbb{E}\left[ind\{A_x = o\} m_o^{\mathcal{F}_h}\right] = \mathbb{E}[ind\{A_x = o\} Y_o] \tag{22}$$

$$= \mathbb{E}\left[ind\{A_x = o\} m_o^{\mathcal{F}_x}\right]. \tag{23}$$

Summing over the finite action set and using the optimality of A_x yields

$$\mathbb{E}[m_{A_x}^{\mathcal{F}_h}] = \mathbb{E}[m_{A_x}^{\mathcal{F}_x}] \tag{24}$$

$$= \mathbb{E}\left[\max_o m_o^{\mathcal{F}_x}\right] = J(\mathcal{F}_x). \tag{25}$$

The refined policy may choose the best option separately at every refined state, so pointwise

$$\max_o m_o^{\mathcal{F}_h} - m_{A_x}^{\mathcal{F}_h} \geq 0. \tag{26}$$

Taking expectations gives

$$J(\mathcal{F}_h) - J(\mathcal{F}_x) = \mathbb{E}\left[\max_o m_o^{\mathcal{F}_h} - m_{A_x}^{\mathcal{F}_h}\right] \tag{27}$$

$$\geq 0. \tag{28}$$

If the displayed gap is strictly positive on an event of positive probability, its expectation is strictly positive, which proves the final claim. The sunk prefix assumption is essential here: both policies compare the same continuation surpluses after paying the same prefix cost. □

For depths $4, 8, 16$, define $S_8 = G_{4\to 8} - \lambda C_{4\to 8}$, $S_{16} = G_{4\to 16} - \lambda C_{4\to 16}$, and $\mathcal{D}_\lambda = \{S_8 \leq 0, S_{16} > 0\}$.

Proposition 2 (Myopic horizon gap)**.** *Suppose a myopic controller stops whenever $S_8 \leq 0$, while a non-local controller may select $4 \to 16$. With stopping surplus normalized to zero,*

$$J_{\text{nonlocal}}(\lambda) - J_{\text{myopic}}(\lambda) \geq \mathbb{E}[S_{16}\mathbf{1}\{\mathcal{D}_\lambda\}]. \tag{29}$$

Proof. Let V_{M} denote the surplus selected by the myopic controller and let

$$V_{\text{N}} = \max\{0, S_8, S_{16}\} \tag{30}$$

be the surplus selected from the non-local action set. Every action available to the myopic controller is also available to the non-local controller. Hence

$$V_{\text{N}} - V_{\text{M}} \geq 0 \qquad \text{on } \mathcal{D}_\lambda^c. \tag{31}$$

On $\mathcal{D}_\lambda$, the condition $S_8 \leq 0$ forces the myopic controller to stop, so $V_{\text{M}} = 0$. The same event satisfies $S_{16} > 0$, and therefore

$$V_{\text{N}} = \max\{0, S_8, S_{16}\} \geq S_{16}. \tag{32}$$

Combining the two events gives the pointwise bound

$$V_{\text{N}} - V_{\text{M}} = (V_{\text{N}} - V_{\text{M}})\mathbf{1}\{\mathcal{D}_\lambda\} + (V_{\text{N}} - V_{\text{M}})\mathbf{1}\{\mathcal{D}_\lambda^c\} \tag{33}$$

$$\geq S_{16}\mathbf{1}\{\mathcal{D}_\lambda\}. \tag{34}$$

By definition, $J_{\text{nonlocal}}(\lambda) = \mathbb{E}[V_{\text{N}}]$ and $J_{\text{myopic}}(\lambda) = \mathbb{E}[V_{\text{M}}]$. Taking expectations of the pointwise bound therefore proves the stated inequality. In particular, the horizon gap is strict whenever $\mathbb{E}[S_{16}\mathbf{1}\{\mathcal{D}_\lambda\}] > 0$. □

Suppose option family o maps reservation price q to score $T_o(q)$.

Theorem 3 (Common-coordinate compatibility)**.** *Let each T_o be continuous and strictly increasing on a price-relevant interval I_o, and let I be a nonempty open interval contained in $\bigcap_o I_o$. If, for every $\lambda, q \in I$, one threshold $\tau(\lambda)$ satisfies*

$$q > \lambda \quad \Longleftrightarrow \quad T_o(q) > \tau(\lambda) \tag{35}$$

for every o, then $T_o(\lambda) = \tau(\lambda)$ for all o and $\lambda \in I$. Consequently, the score maps coincide on I. Conversely, a common strictly increasing map $T_o = T$ on I satisfies equation 35 with $\tau(\lambda) = T(\lambda)$.

Proof. Fix an option family o and a price $\lambda \in I$. Because I is open, there are sequences $q_n^+ \downarrow \lambda$ and $q_n^- \uparrow \lambda$ with $q_n^+ > \lambda$, $q_n^- < \lambda$, and all terms in I. For every n, equation 35 gives

$$T_o(q_n^+) > \tau(\lambda). \tag{36}$$

Continuity of T_o at λ implies

$$T_o(\lambda) = \lim_{n\to\infty} T_o(q_n^+) \geq \tau(\lambda). \tag{37}$$

For the lower sequence, $q_n^- > \lambda$ is false. The equivalence in equation 35 therefore forces $T_o(q_n^-) > \tau(\lambda)$ to be false, or

$$T_o(q_n^-) \leq \tau(\lambda). \tag{38}$$

Continuity again gives

$$T_o(\lambda) = \lim_{n\to\infty} T_o(q_n^-) \leq \tau(\lambda). \tag{39}$$

The two inequalities imply $T_o(\lambda) = \tau(\lambda)$. Neither $\tau(\lambda)$ nor this conclusion depends on o, so for any two families o, o',

$$T_o(\lambda) = \tau(\lambda) = T_{o'}(\lambda) \qquad \text{for every } \lambda \in I. \tag{40}$$

Thus all score maps agree on the common price interval.

Conversely, suppose $T_o = T$ on I for every family, where T is strictly increasing. For any $q, \lambda \in I$,

$$q > \lambda \quad \Longleftrightarrow \quad T(q) > T(\lambda). \tag{41}$$

Setting $\tau(\lambda) = T(\lambda)$ gives equation 35 simultaneously for every option family. □

Corollary 4 (Option-wise percentile incompatibility)**.** *If $T_o = F_o$ is an option-specific CDF that is continuous and strictly increasing on the common interval I, common-price compatibility holds on I if and only if the CDFs coincide there. If Q_o has CDF F_o, then $F_o(Q_o) \sim \mathrm{Uniform}(0,1)$, so a common score threshold otherwise imposes the same marginal quantile on every option family. The pooled map F_{pool} remains common and nondecreasing and preserves global weak ordering up to empirical ties.*

Proof. Apply Theorem 3 with $T_o = F_o$. Compatibility implies

$$F_o(\lambda) = \tau(\lambda) = F_{o'}(\lambda) \tag{42}$$

for every pair o, o' and every $\lambda \in I$, so the CDFs coincide on I. Conversely, if all CDFs equal one strictly increasing function F on I, then

$$q > \lambda \quad \Longleftrightarrow \quad F(q) > F(\lambda), \tag{43}$$

and the common threshold $\tau(\lambda) = F(\lambda)$ is compatible.

For a random reservation price Q_o with continuous CDF F_o, the probability integral transform yields

$$\mathbb{P}(F_o(Q_o) \leq u) = u, \qquad 0 \leq u \leq 1. \tag{44}$$

Thus a threshold u selects the same upper-tail probability $1-u$ within every option family, regardless of the underlying cardinal prices. If the F_o differ, equal percentile scores can therefore correspond to unequal reservation prices.

Finally, for any two prices $q \leq q'$, monotonicity of the pooled empirical CDF gives

$$F_{\mathrm{pool}}(q) \leq F_{\mathrm{pool}}(q'). \tag{45}$$

The same map is applied to every family, so cross-option order cannot reverse. Empirical flat regions can create ties, which is why the preserved ordering is weak rather than strict. □

Threshold compatibility versus surplus ordering. Theorem 3 concerns the binary condition for whether an isolated option clears a common threshold. A stronger property is required to preserve comparisons between unequal-cost options. For a common transform T, compare the original and transformed surpluses

$$S(q, c; \lambda) = c(q - \lambda), \qquad \widetilde{S}(q, c; \lambda) = c\{T(q) - T(\lambda)\}. \tag{46}$$

Their ordering is identical for every q, $c > 0$, and λ only when T is positive affine on the relevant interval. Sufficiency follows from $T(q) = aq + b$, $a > 0$, which gives $\widetilde{S} = aS$. For necessity, fix λ and two prices $q_1, q_2 > \lambda$. Choose positive costs so that $c_1(q_1 - \lambda) = c_2(q_2 - \lambda)$. Preservation of this tie requires

$$\frac{T(q_1) - T(\lambda)}{q_1 - \lambda} = \frac{T(q_2) - T(\lambda)}{q_2 - \lambda}. \tag{47}$$

Hence every secant from λ has the same slope; varying λ over overlapping subintervals makes that slope constant, so T is affine.

A nonlinear common map can therefore preserve threshold crossings while changing a multi-option choice. With $\lambda = .01$, let $(q_A, c_A) = (.04, 1)$ and $(q_B, c_B) = (.02, 4)$. The original surpluses are .03 and .04, so B is preferred. Under $T(q) = q^2$ and transformed price $T(\lambda)$, the surpluses become .0015 and .0012, so A is preferred. Rank-SPOI intentionally defines its allocator on the shared transformed coordinate and does not claim to reconstruct the original cardinal utility. Cardinal-SPOI avoids this transformation by estimating q directly; validation determines which estimator generalizes better.

B Additional experiments

B.1 Scale selection and strong learned baselines

Cardinal-SPOI and Rank-SPOI share the state readout, fit/validation split, option set, and allocator. Cardinal training applies Huber loss after one pooled affine standardization of q; rank training uses Equation 11. Their selector compares mean validation utility over the nine fixed budgets. Query-only uses a fixed 512-tree ExtraTrees regressor over a 128-dimensional character-TF–IDF projection and 16 structural input features; it learns the pooled rank target and uses the same allocator. The scale selector and query-only comparison were specified after the original rank-only evaluation and before the corresponding held-out results were opened; Table 4 reports this extension separately.

Table 4: Comparison with learned baselines. Wins count budgets, out of nine, where validation-selected SPOI has higher mean utility; ΔU is mean SPOI utility minus the named baseline.

	Query-only		Separate	
Setting	Wins	ΔU	Wins	ΔU
Raven / GSM8K	5	$+0.0001$	4	-0.0005
Raven / MATH	8	$+0.0020$	8	$+0.0046$
Huginn / GSM8K	9	$+0.0461$	9	$+0.0551$
Huginn / MATH	9	$+0.0083$	9	$+0.1103$
Qwen / GSM8K	4	$+0.0005$	9	$+1.6204$
Qwen / MATH	3	-0.0074	9	$+2.9549$
Phi / GSM8K	9	$+0.2332$	9	$+1.7633$
Phi / MATH	8	$+0.3825$	9	$+1.7785$

B.2 Cardinal calibration audit

The learned allocator operates on the transformed coordinate in Equation 12. To measure the associated cardinal distortion without refitting the controller, we retain every frozen Qwen/Phi score and replace $\widehat{z}$ by $F^{-1}_{\text{pool}}(\widehat{z})$, where the inverse empirical quantile is estimated from fit-set reservation prices. Table 5 averages five seeds and nine budgets. Qwen has no $r_8 \leftrightarrow r_{16}$ swaps under this substitution. Phi changes continuation depth more often, and inverse-quantile calibration increases both held-out utility and agreement with the matched-budget Oracle. This diagnostic isolates calibration with the learned scores fixed; Cardinal-SPOI supplies the separate Direct-q evaluation.

Table 5: Post-hoc calibration audit over frozen Qwen/Phi scores. "Any" is the fraction of assignments that differ; "swap" counts only $r_8 \leftrightarrow r_{16}$ changes. ΔOracle is the change in exact choice agreement, in percentage points, and ΔU is inverse-quantile minus rank-coordinate utility.

Setting	Any (%)	Swap (%)	ΔOracle	ΔU
Qwen / GSM8K	0.024	0.000	$+0.004$	$+0.0000$
Qwen / MATH	0.013	0.000	-0.004	-0.0000
Phi / GSM8K	3.049	3.040	$+1.105$	$+0.0338$
Phi / MATH	8.594	8.220	$+2.470$	$+0.0901$

B.3 Allocation phase transitions

Figure 7 exposes decisions underneath aggregate frontiers. As budget increases, mass can move through r_8 or directly from r_4 to r_{16}, with paths differing across backbones and tasks.

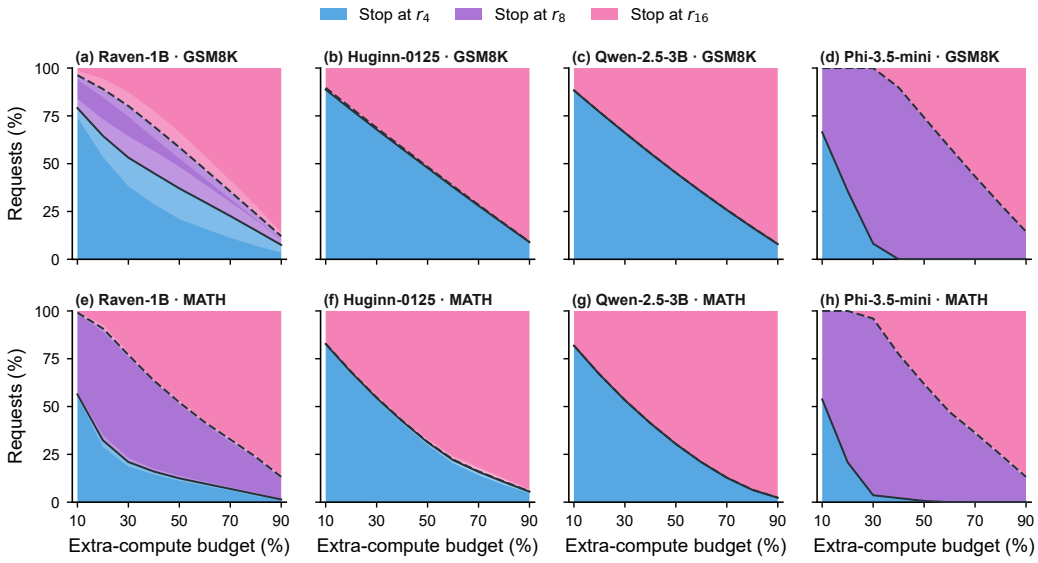

Figure 7: Budget-dependent allocation phase transitions. Areas are fractions stopping at r_4, r_8, and r_{16}; dark boundaries show cumulative r_4 and $r_4{+}r_8$ populations. The audit covers 368,640 held-out request decisions.

B.4 Pooled-rank coordinate diagnostic

Table 6: Pooled-rank versus option-wise rank selection. "Pooled" counts seeds selected by the original paired validation certificate. Gains are relative to Separate; negative cells count selected gains below zero among 45 seed–budget cells.

Setting	Pooled	Mean ΔU	Worst ΔU	Negative
Raven / GSM8K	0/5	0.0000	0.0000	0/45
Raven / MATH	4/5	0.0034	-0.0009	6/45
Huginn / GSM8K	5/5	0.0543	-0.0004	1/45
Huginn / MATH	5/5	0.1101	0.0183	0/45
Qwen / GSM8K	5/5	1.6116	0.4973	0/45
Qwen / MATH	5/5	2.9435	0.4666	0/45
Phi / GSM8K	5/5	1.7068	0.0481	0/45
Phi / MATH	5/5	1.5545	0.1146	0/45

B.5 Controller and allocator overhead

The 114,924-parameter readout occupies 0.44 MB in FP32. The fast allocator matches the discrete reference in every test and reduces allocation latency by $3.0\times$–$72.0\times$.

Table 7: CPU decision-path audit. Total μs/request includes readout and fast allocation. Agreement and violation compare fast and exact allocators.

Batch	Readout ms	Fast ms	Exact ms	Total μs/req.	Agree	Violation
32	2.464	0.984	2.971	107.8	100%	0
128	3.915	1.446	13.943	41.9	100%	0
512	7.786	2.854	89.966	20.8	100%	0
1024	13.179	3.666	263.924	16.4	100%	0

B.6 Small linear decodability control

The nonlinear SPOI controller and the 136-parameter auxiliary probe serve distinct purposes. The probe combines 128 fixed random-projection features, six fixed movement features, and one option bit in a ridge model with 135 coefficients and one intercept; training data determine its regularization.

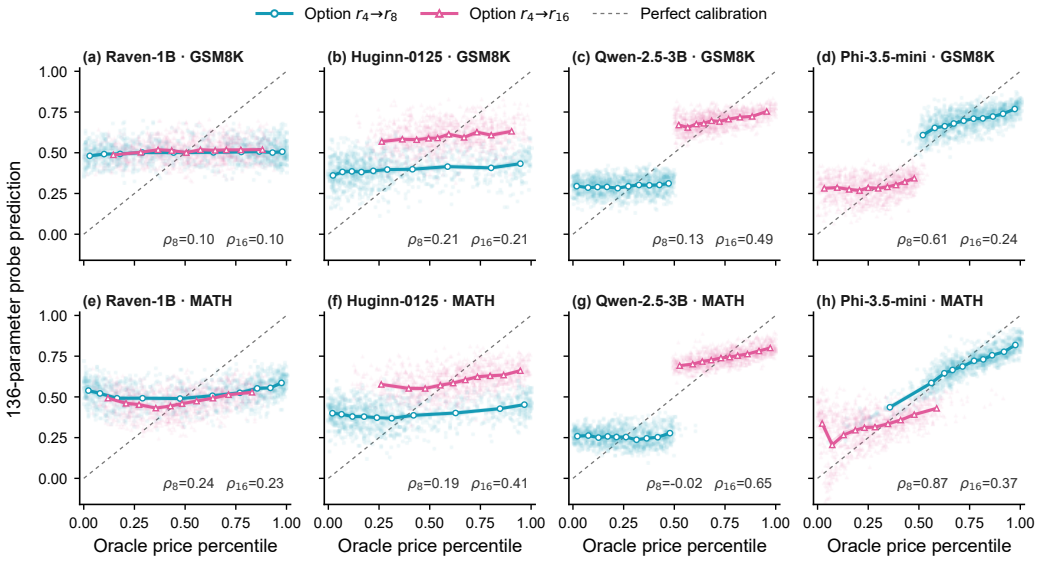

Figure 8: Held-out option-value predictions from an exactly 136-parameter linear probe. Across 16,384 request–option pairs, linear accessibility varies substantially by option and backbone.

C Reproducibility and complete protocols

Data and splits. Experiments use GSM8K (Cobbe et al., 2021) and the numeric subset of MATH (Hendrycks et al., 2021). The recurrent backbones are Raven-1B (McLeish et al., 2025) and Huginn-0125 (Geiping et al., 2025); transformer replications use Qwen2.5-3B-Instruct (Qwen Team, 2024) and Phi-3.5-mini-instruct (Abdin et al., 2024). For recurrent-backbone GSM8K experiments, 2,048 training trajectories are separated from 1,024 once-only test trajectories. MATH processing first retains 8,048 unique problems whose final boxed answer is an unambiguous integer or decimal, then fixes disjoint 2,048-example training and 1,024-example test pools plus a separate formatting demonstration. Fractions, radicals, sets, and symbolic expressions are never approximately credited. Transformer replications use the same frozen option set and a manifest-recorded

Algorithm 1 SPOI: Shared-Price Training and Budgeted Active Inference

Require: Training trajectories $\{(h_i, \{U_i(d), C_i(d)\}_{d\in\mathcal{D}})\}_{i=1}^{M}$; validation set $\mathcal{V}$; serving batch $\mathcal{X}$; average budget B

Ensure: Frozen coordinate rule, depth assignments $\{\hat{d}_i\}$, and compacted execution

Stage 1: Construct a shared option-rank coordinate *(offline)*

for each trajectory i and continuation option $o = 1, \ldots, K$ **do**
  $G_{io} \leftarrow U_i(d_o) - U_i(r_0)$; $\Delta C_{io} \leftarrow C_i(d_o) - C_i(r_0)$
  $q_{io} \leftarrow G_{io}/\Delta C_{io}$
end for
Fit one pooled empirical map F_{pool} over all $\{q_{io}\}_{i,o}$ and set $z_{io} \leftarrow F_{\text{pool}}(q_{io})$
Train Rank-SPOI with $\mathcal{L}_{\text{SPOI}}$ in equation 11 and Cardinal-SPOI on affine-standardized q
Compare mean utility over the fixed validation budget grid
Freeze the higher-utility estimator before held-out evaluation

Stage 2: Evaluate one batch-wide allocation *(held out)*

Execute the shared prefix r_0 for every request $x_i \in \mathcal{X}$ and form its causal state summary $h_i(r_0)$
Predict selected gains $\{\overline{G}_{io}\}_{o=1}^{K}$ with the frozen readout
Find $\lambda_B \geq 0$ by monotone search such that
  $displaystyle\hat{d}_i(\lambda_B) \in \arg\max_{d\in\mathcal{D}}\{\overline{U}_i(d) - \lambda_B C_i(d)\}, \qquad \sum_{i=1}^{N} C_i(\hat{d}_i) \leq NB$
Resolve any discrete budget slack using the largest positive marginal surplus

Stage 3: Realize the allocation *(frozen-schedule replay)*

for $d \in \{r_0, d_1, \ldots, d_K\}$ in increasing order **do**
  Emit requests assigned to d and compact them out of the active tensors
  Execute the next recurrent block only for requests with $\hat{d}_i > d$
end for
return outputs and realized compute $\sum_i C_i(\hat{d}_i)$

disjoint split; each of seeds 2027–2031 makes an 80/20 fit/validation split inside training data. Validation selects the coordinate regime and hyperparameters; test data are used once for reported utility. The router never receives gold answers or future trajectory outcomes at inference.

Trajectory targets and compute accounting. All policies observe the same state after r_4 and choose among terminal depths $\{4, 8, 16\}$. Utilities are computed from the frozen task scorer at each cached depth. Gains subtract the r_4 utility, costs subtract measured r_4 compute, and the shared prefix is sunk for every policy. The nine budget fractions $0.1, \ldots, 0.9$ interpolate between all-r_4 and all-r_{16} incremental cost. Oracle, SPOI, Separate, length, and random policies are cleared by the same allocator and checked against the same realized aggregate budget.

Controller architecture and training. Equation 7 uses 48-dimensional group projections and option queries. Forty-five movement features pass through a $45 \to 48$ GELU layer with dropout 0.15; each $96 \to 48$ option head uses dropout 0.20. On the audited 2,048-hidden-state configuration, the readout has 114,924 trainable parameters. The recurrent LM remains frozen. Five seeds use matched splits and initialization across Cardinal-SPOI and Rank-SPOI; mean validation utility over the fixed budget grid selects the estimator.

A800 execution protocol. The systems run uses one NVIDIA A800 80GB PCIe GPU, PyTorch bfloat16, and sequence length 160. It crosses batch sizes $8, 16, 32, 64, 128$ with all nine budgets, five independently replayed held-out schedules, three execution methods, and ten synchronized timings per schedule, for 675 saved rows, 6,750 individual timings in total, and 2,250 timings per method. The timed region includes the model prelude, recurrent core, required coda readouts, pre-coda dynamics reduction, GPU controller, two-score GPU-to-CPU transfer, shadow-price allocation, and post-r_4/post-r_8 tensor compaction. Hierarchical bootstrap first resamples schedules and then synchronized timings. Depth schedules come from held-out examples but are replayed on one representative fixed-length sequence; online feature materialization is timed, whereas cached features preserve policy identity. The benchmark directly measures prompt-scoring/prefill wall clock for the frozen heterogeneous schedules.

KV-cache execution protocol. The cross-architecture audit uses Qwen-2.5-3B-Instruct at physical exits $\{9, 18, 36\}$ and Phi-3.5-mini-instruct at $\{8, 16, 32\}$, both mapped to the canonical $\{4, 8, 16\}$ option set. It measures 128-token prefill plus eight teacher-forced decode tokens with materialized KV-cache reads and writes. Qwen is tested at batches 32 and 128 and Phi at batch 128, each at budgets $0.1, 0.5, 0.9$, five frozen schedules, and five synchronized timings per method. DenseKV and BucketedKV receive identical tokens and terminal-depth choices; only execution is changed. The paired correctness path reads the selected exit logits and obtains 1.000 selected-token agreement in every cell. Hierarchical bootstrap resamples schedules and then timings. The resulting depth-bucketed fixed-length evaluation pairs semantic and timing measurements.

Per-budget A800 audit. Table 8 reports all nine budgets. Speedup averages the five batch-size point estimates; exit mass and overhead use the same aggregation.

Table 8: Complete A800 budget sweep. Exit is the percentage leaving before r_8; overhead includes controller and compaction relative to the same schedule without online control.

Budget	Mean speedup	Exit before r_8	Mean overhead
0.1	2.249×	91.0%	0.64%
0.2	1.986×	77.9%	0.53%
0.3	1.841×	68.5%	0.44%
0.4	1.614×	56.5%	0.41%
0.5	1.473×	46.0%	0.37%
0.6	1.296×	37.3%	0.31%
0.7	1.234×	30.7%	0.33%
0.8	1.106×	18.2%	0.30%
0.9	1.042×	8.0%	0.27%

D SYSTEMS AND TRANSFER ANALYSES

Work-law fit. The recurrent budget-0.1 measurement sets the fixed-work ratio in Equation 17 to $\rho \approx 3.7214$. The resulting curve has 2.83% descriptive MAPE over the other eight budgets; at budget 0.9 it yields $1.051\times$ against $1.042\times$ observed. The post-hoc anchor makes this a goodness-of-fit analysis of the execution accounting model.

Complete KV-cache matrix. Table 9 reports all nine frozen-schedule cells, including break-even and negative high-budget cells.

Table 9: Frozen-schedule A800 audit on standard KV-cache Transformers. The layer-work ceiling excludes embedding, head, launch, and depth-bucketing overhead. Intervals hierarchically resample five held-out schedules and five timings per method. Selected-exit token agreement is 1.000 in every cell.

Backbone	Batch	B	(a_1, a_2)	Work ceiling	A800 speedup [95% CI]	Dense→bucket (ms)	Peak GB
Phi-3.5-mini	128	0.1	(.336, .000)	2.994×	2.595× **[2.497, 2.684]**	1579.7 → 608.9	20.44 → 9.92
Phi-3.5-mini	128	0.5	(1.00, .258)	1.590×	1.351× **[1.337, 1.379]**	1588.2 → 1175.6	20.44 → 12.22
Phi-3.5-mini	128	0.9	(1.00, .855)	1.078×	0.998× [0.988, 1.010]	1585.9 → 1589.8	20.44 → 18.60
Qwen-2.5-3B	32	0.1	(.125, .125)	2.909×	1.152× **[1.126, 1.175]**	309.8 → 267.6	6.47 → 6.32
Qwen-2.5-3B	32	0.5	(.556, .556)	1.499×	1.015× [0.991, 1.035]	316.2 → 311.1	6.47 → 6.30
Qwen-2.5-3B	32	0.9	(.906, .906)	1.076×	0.891× [0.860, 0.914]	315.2 → 353.6	6.47 → 6.44
Qwen-2.5-3B	128	0.1	(.117, .117)	2.960×	1.856× **[1.799, 1.878]**	749.1 → 403.0	7.87 → 7.27
Qwen-2.5-3B	128	0.5	(.547, .547)	1.515×	1.279× **[1.262, 1.300]**	762.7 → 596.1	7.87 → 7.09
Qwen-2.5-3B	128	0.9	(.922, .922)	1.062×	0.990× [0.981, 1.001]	768.0 → 776.6	7.87 → 7.74

Native-generation transfer. The native-depth development evaluation uses depths 16/24/32 and a 512-token cap. Its non-monotone fixed-depth accuracy establishes depth-dependent native-generation opportunity. Each free-running trajectory supplies one terminal binary outcome rather

than candidate-wise counterfactual supervision, so 256 outcomes provide sparse evidence for a 136-parameter generation posterior. Learning the continuation-value interface directly from free-running trajectories would require larger randomized or on-policy route collections, intermediate verifier signals, and semantic-equivalence tests across route-dependent token histories. This is an upstream value-estimation regime; the shared-price allocation rule is unchanged once comparable continuation values are supplied.