WHEN DOES MORE INFORMATION PAY OFF? EFFECTIVE COST IN LLM INFERENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

More context is not always cheaper. Extra information can reduce how much an LLM needs to generate, but a longer prompt also makes each cached decoding step more expensive. We study when the saved computation outweighs the cost of carrying additional context. We define effective cost as the smallest physical budget needed to reach a common target, such as a desired response behavior or task score. Under regularity conditions, one representation provides at least as much attainable utility as another across budgets and decision objectives exactly when it reaches every response behavior attainable from the other at no greater effective cost. We then derive when a larger context is worthwhile and develop rules for jointly choosing the input context and the amount of generation, including the cost of constructing or replacing the cached state. In controlled routing, useful extra information lowers the cost of reaching a common numerical success threshold despite a longer prompt. On 576 held-out MuSiQue questions with supplied intermediate answers, jointly choosing how many paragraphs to retain and how long to generate improves mean answer F1 by 0.040 over choosing one evidence scope per budget, averaged across three prespecified profiled GPU-time budgets. Finally, rebuilding an active KV cache from a shorter prompt yields decoding savings that exceed the measured rebuilding cost.

1 INTRODUCTION

Large language models (LLMs) can benefit from evidence supplied in their input, or prompt, but processing and using that evidence also consumes computation. Under a fixed time or compute budget, input selection and response generation must therefore be considered together. Retrieval-augmented generation (RAG), for example, supplies external documents (Lewis et al., 2020), while test-time compute methods allocate additional computation to producing or assessing an answer (Snell et al., 2025). These choices interact because the supplied information can change both the cost and usefulness of subsequent computation. For instance, Fang et al. (2026) show that additional computation to process retrieved text can improve answer accuracy. When is additional information worth its execution cost, and how much computation should follow?

The cost of supplied information extends beyond prompt processing into the subsequent computation, which we call continuation. An LLM generates one token, a small unit of text, at a time. In dense cached decoding, each step accesses stored states for all preceding tokens in a key-value (KV) cache. Caching avoids prompt recomputation, but longer prompts leave more cached states to access throughout generation (Pope et al., 2023; Kwon et al., 2023). We call the size of this retained input state its footprint, measured here by the number of prompt tokens in the KV cache. Additional information can nevertheless reduce the amount of generation needed to complete a task. For example, supplying a useful intermediate result can allow the model to omit its derivation from the response. A longer prompt can thus make each generation step more costly while reducing required work, the least continuation work needed to reach a target.

Adaptive retrieval and context compression seek to improve accuracy and efficiency by choosing or shortening evidence (Jeong et al., 2024; Pan et al., 2024). To compare the value of such inputs, formal frameworks characterize the decisions they support or their predictive value within a restricted predictor family (Blackwell, 1953; Xu et al., 2020). Frameworks for costly access and computation also account for the resources needed to obtain or use information (Park et al., 2026a; Halpern & Pass,

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.

2010). Applying these perspectives to cached inference still requires an explicit link between required work and the physical cost of executing it at the retained footprint. We therefore ask when reductions in required work compensate for a larger footprint at a common target, and how this tradeoff should guide the choice of input and continuation.

To answer this question, we introduce effective cost for representations, which specify how task information is supplied and retained. Effective cost is the least budget needed to reach a target using the allowed continuation procedures. We analyze how required work and footprint determine this cost, yielding rules for comparing representations and jointly selecting a representation and continuation. Our contributions are:

- **Comparing representations through effective cost.** Under stated regularity conditions, we characterize comparisons of attainable expected utility across budgets and decision objectives through the effective costs of target response behaviors.
- **Quantifying the tradeoff between footprint and work.** We derive the work savings needed to offset a larger footprint and an explicit break-even boundary for dense cached decoding.
- **Selecting representations and continuations.** We derive rules for minimizing cost at a utility threshold or maximizing utility within a budget, accounting for acquisition and transition costs.
- **Evaluation in three inference settings.** We evaluate these elements through three complementary experiments. First, we use controlled routing to test whether useful information reduces target cost despite a larger footprint. Second, we demonstrate gains from jointly selecting evidence scope and generation checkpoint in MuSiQue question answering. Third, we consider changes to an active representation and test whether continuation savings offset the cost of rebuilding its cache.

2 Effective Cost: Comparing Representations at a Common Target

Two representations can attain the same expected score while succeeding on different instances. To compare their value across decision objectives, we first characterize which response behaviors are affordable. We then specialize to a fixed task, where attaining a utility threshold is sufficient.

2.1 Behaviors and budgeted capability sets

Let Θ be a finite set of task states and $\mathcal{A}$ a finite set of task-level responses, called actions. For question answering, a state $\theta \in \Theta$ specifies the question and the facts relevant to answering it, while an action $a \in \mathcal{A}$ is a possible response. A representation channel $E : \Theta \rightsquigarrow \mathcal{Z}$ supplies an observation $z \in \mathcal{Z}$, such as a prompt containing the question and selected evidence. The symbol $\rightsquigarrow$ denotes a possibly randomized map. For example, randomized evidence selection can produce different prompts for the same task state. A response rule $D : \mathcal{Z} \rightsquigarrow \mathcal{A}$ maps the observation to a response and may also be randomized, as when an LLM samples output tokens. Deterministic prompt construction and response generation are included as special cases. The behavior $Q = ED$ assigns probability $Q_{\theta a}$ to response a in state θ, averaging over randomness in E and D. Behaviors share a common state and action space, allowing comparison across representations with different observation spaces.

A channel alone does not specify execution cost. A representation therefore pairs the channel with a specified implementation for retaining and accessing its observations during continuation. For cached decoding, this retained state includes the key-value states produced by processing the prompt z. The specified implementation is held fixed in all costs and capability sets indexed by E. Different algorithms can implement the same response rule at different costs. We therefore fix the allowed continuation algorithms, called the procedure family, as either a general class or a finite set of implementations.

Let $\mathcal{D}_E^b(\mathcal{A})$ contain the response rules that these algorithms can implement on observations from E within a physical budget $b \geq 0$, for a specified resource such as execution time, operation count, or energy. The capability set collects the resulting attainable behaviors:

$$\widehat{\mathcal{Q}}_E^b(\mathcal{A}) = \mathrm{cl}\,\mathrm{co}\{ED : D \in \mathcal{D}_E^b(\mathcal{A})\}. \tag{1}$$

The convex hull co allows random selection among individually affordable rules, with the selection overhead included or negligible. The closure cl includes limiting behaviors, which may be arbitrarily approximable without a single finite implementation.

We evaluate these behaviors using a prior p that assigns probabilities to task states and a bounded utility $u(\theta, a)$ that scores response a in state θ. Their expected utility under behavior Q is $\langle u, Q\rangle_p = \sum_{\theta,a} p_\theta u(\theta, a) Q_{\theta a}$. The budgeted value is the greatest feasible expected utility:

$$V_E^b(p, u; \mathcal{A}) = \max_{Q \in \widehat{\mathcal{Q}}_E^b(\mathcal{A})} \langle u, Q\rangle_p. \tag{2}$$

We suppress fixed objective and action-space arguments when clear. We assume that the capability sets are nonempty and compact, ensuring that the maximum exists. They are also nested, so a larger budget preserves every previously feasible behavior and cannot decrease budgeted value. Finally, budget closure requires a behavior feasible at every $b + \delta$, $\delta > 0$, to remain feasible at b. This condition includes the boundary budget at which a target first becomes affordable. Appendix A states the assumptions formally and gives the proofs.

2.2 Reversing the question: the cost of a target

Budgeted value maximizes expected utility at a fixed budget. To compare representations at a common target behavior Q, we instead minimize the required budget, defining its effective cost as

$$\bar{\kappa}_E(Q; \mathcal{A}) = \inf\{b \geq 0 : Q \in \widehat{\mathcal{Q}}_E^b(\mathcal{A})\}, \tag{3}$$

with $+\infty$ for an unreachable target. Nesting and budget closure give $Q \in \widehat{\mathcal{Q}}_E^b \iff \bar{\kappa}_E(Q) \leq b$: if the infimum is at most b, the target is feasible at every slightly larger budget, and closure includes the boundary. Since effective costs recover the capability sets at every budget, they also determine budgeted value for every decision objective.

Theorem 2.1 (Effective-cost criterion)**.** *Under assumptions A.1 and A.2, for finite $\mathcal{A}$, the following are equivalent: (i) $V_{E_1}^b(p, u; \mathcal{A}) \geq V_{E_2}^b(p, u; \mathcal{A})$ for every budget, prior, and bounded utility, and (ii) $\bar{\kappa}_{E_1}(Q; \mathcal{A}) \leq \bar{\kappa}_{E_2}(Q; \mathcal{A})$ for every behavior Q attainable from E_2 at a finite budget.*

If every target is no more expensive under E_1, every behavior affordable under E_2 is also affordable under E_1. Conversely, a linear score can assign any behavior outside a closed convex capability set higher expected utility than every behavior inside it. This supplies a decision objective violating value dominance and proves the converse.

A task may require a minimum score without prescribing every response probability. We therefore define the target-cost frontier for a fixed prior and utility as the least budget needed to reach τ:

$$\kappa_E^{p,u}(\tau; \mathcal{A}) = \inf\{b \geq 0 : V_E^b(p, u; \mathcal{A}) \geq \tau\}. \tag{4}$$

For binary success, $\tau = 0.75$ requires expected success at least 0.75, without prescribing which instances must succeed. The same boundary conditions give $V_E^b \geq \tau \iff \kappa_E^{p,u}(\tau) \leq b$.

Corollary 2.1 (Fixed-task target costs)**.** *Under the same assumptions, $V_{E_1}^b \geq V_{E_2}^b$ at every budget for a fixed $(p, u, \mathcal{A})$ if and only if $\kappa_{E_1}^{p,u}(\tau; \mathcal{A}) \leq \kappa_{E_2}^{p,u}(\tau; \mathcal{A})$ for every utility threshold τ.*

The effective-cost criterion compares the cost of attaining the same response behavior. For a fixed task, the target-cost frontier permits any behavior meeting the common utility threshold.

3 From Required Work to Representation Selection

To explain why representations can have different effective costs for the same target, we separate required work from its physical execution cost. We first analyze continuations that retain the chosen input representation while generated output may grow. For cached decoding, continuation accounting begins after prompt processing, when the KV cache is available. We derive the work savings needed to offset a larger footprint and a break-even boundary for dense cached decoding.

We then examine attainable utility after a transition changes an active representation. Selection also includes acquisition, which constructs and prepares the initial representation for continuation. We combine continuation cost with any acquisition or transition costs incurred after a common decision point. The resulting selection rules minimize total cost at a utility threshold or maximize budgeted value within a total budget.

3.1 THE TRADEOFF BETWEEN FOOTPRINT AND WORK

The footprint $s(E)$ is the fixed size of the input state retained during continuation, such as the number of prompt tokens represented in the KV cache. Let $\mathcal{P}(\mathcal{A})$ be a common class of continuation programs P producing responses in $\mathcal{A}$. Executing P on observations from E induces the response rule $D_{E,P}$. Its continuation work $w(P) \geq 0$ measures computation on a common scale, such as scheduled decoding steps. Required work is the least allowance r that makes Q attainable:

$$\rho_E(Q) = \inf\{r \geq 0 : Q \in \operatorname{cl}\operatorname{co}\{ED_{E,P} : P \in \mathcal{P}(\mathcal{A}),\ w(P) \leq r\}\}. \tag{5}$$

Required work can differ across representations because supplied information can replace computation that a continuation would otherwise perform.

To convert work into physical cost under the specified implementation, we use $C_E^{\mathrm{cont}}(P) = \Phi(s(E), w(P))$. For each footprint, $\Phi(s, \cdot)$ is continuous, strictly increasing, unbounded, and zero at zero. It is also nondecreasing in footprint. Thus additional work costs more, and retaining more state cannot reduce the cost of a fixed amount of work within this model. Zero work incurs no continuation charge because acquisition is accounted for separately. For a utility threshold, write $\rho_E^{p,u}(\tau; \mathcal{A})$ for the required work: the least work allowance whose maximum expected utility reaches τ. For either target, converting its required work through Φ gives its physical cost:

$$\bar{\kappa}_E(Q; \mathcal{A}) = \Phi(s(E), \rho_E(Q)), \qquad \kappa_E^{p,u}(\tau; \mathcal{A}) = \Phi(s(E), \rho_E^{p,u}(\tau; \mathcal{A})). \tag{6}$$

Strict increase ensures that a work allowance and its corresponding physical budget admit the same programs, hence the same capability set. Continuity preserves this correspondence at the infima defining required work and effective cost. Unboundedness gives infinite cost for infinite required work, with $\Phi(s, +\infty) = +\infty$. Appendix B gives the proof.

To quantify the work reduction needed to offset a larger footprint, define the work allowance

$$T_\Phi(s_1, s_2; r) = \sup\{r' \geq 0 : \Phi(s_1, r') \leq \Phi(s_2, r)\}. \tag{7}$$

The allowance is the maximum work at footprint s_1 that costs no more than r units of work at s_2. Continuity ensures that this maximum is included, and monotonicity in footprint gives $T_\Phi(s_1, s_2; r) \leq r$ when $s_1 \geq s_2$. A target is therefore no more expensive under E_1 exactly when its required work fits this allowance. For example, if cost is linear in work and each unit costs twice as much at the larger footprint, required work must be at least halved. The next result applies the same comparison to every attainable behavior.

Theorem 3.1 (Work allowance criterion)**.** *Under assumptions A.1, A.2 and B.1, the dominance in Theorem 2.1 holds exactly when $\rho_{E_1}(Q) \leq T_\Phi(s(E_1), s(E_2); \rho_{E_2}(Q))$ for every behavior attainable from E_2 at a finite budget.*

The allowance criterion explains why greater informativeness alone need not lower effective cost. A Blackwell relation $E_2 = E_1 G$ means that a stochastic map G reproduces E_2's observation distribution from observations under E_1 in every state, but gives no execution-work bound. The semantic compiler closure assumption additionally requires every continuation under E_2 to be reproducible under E_1 with no more work. As shown in Appendix B, this assumption implies $\rho_{E_1}(Q) \leq \rho_{E_2}(Q)$. A larger footprint can require a strict work reduction, so this bound alone does not establish cost dominance.

3.2 A BREAK-EVEN BOUNDARY FOR CACHED DECODING

To express the work allowance as a limit on additional context, we specialize the cost model to dense cached decoding. Each step accesses the cached states of the input and preceding output tokens. Let $N \geq 1$ be the input token count and K the number of continuation steps. At step $t = 0, \ldots, K-1$, the accessible sequence contains $N + t$ tokens. We model the cost of that step as $\alpha(N + t) + \beta$, where $\alpha > 0$ is the access cost per cached token and $\beta \geq 0$ accounts for work independent of sequence length. Summing over the K steps gives

$$\Phi(N, K) = \alpha NK + \frac{\alpha}{2} K(K-1) + \beta K, \qquad \alpha > 0, \quad \beta \geq 0. \tag{8}$$

The NK term charges repeated access to the retained prefix. The quadratic term charges the growing suffix. This idealized model captures a dense attention interaction (Vaswani et al., 2017; Pope et al.,

2023). Empirical selection instead uses independently profiled costs. Appendix D verifies that this model satisfies the assumptions on Φ for $N \geq 1$ and $K \geq 0$.

Suppose compact E_2 needs K_2 steps, while E_1 adds $M \geq 0$ input tokens and needs $0 < K_1 \leq K_2 < \infty$ for the same target behavior or utility threshold. The additional input costs $\alpha M K_1$. Eliminating $K_2 - K_1$ steps saves both their prefix access and their other execution costs.

Corollary 3.1 (Break-even additional input)**.** *Under Eq.* (8)*, E_1 has no greater continuation cost exactly when*

$$M \leq M^\star, \qquad M^\star = \frac{K_2 - K_1}{K_1}\left[N + \frac{K_1 + K_2 - 1}{2} + \frac{\beta}{\alpha}\right]. \tag{9}$$

The cost gap is $\alpha K_1(M - M^\star)$, so its sign changes exactly at this boundary. The ratio $(K_2 - K_1)/K_1$ compares steps saved with steps that must carry the additional input. The β/α term captures the prefix-independent cost saved by eliminating steps: a larger β lets the same work reduction justify more additional input. Without a work reduction, $M^\star = 0$. If $K_1 = 0$, continuation is free under the model and no division is needed.

3.3 Changing an active representation

The preceding cost comparison retains the chosen input representation throughout continuation. Changing that state can lower subsequent execution cost, so the comparison must include the transition cost. Recovering observations through G alone does not change the retained state, whereas rebuilding a smaller KV cache does. Let R map observations from E_1 to the reference channel E_2's observation space, inducing $\widetilde{E}_2 = E_1 R$. We call this transformation restoration and assume that producing its observations and establishing their continuation state costs at most d. Matching observations alone does not guarantee the reference execution cost. We therefore require each response rule executable under E_2 to remain executable after restoration at no greater continuation cost. We measure restoration error by total variation d_{TV}, the largest difference in probability assigned to any event. This measure bounds the utility loss after any shared response rule when scores lie in $[0, 1]$. Writing $(E)_\theta$ for the observation distribution in state θ, we assume $\sup_\theta d_{\mathrm{TV}}((\widetilde{E}_2)_\theta, (E_2)_\theta) \leq \varepsilon$ to obtain a guarantee for every prior.

Let $V_{E_1}^{B,\mathrm{tr}}$ denote budgeted value when the procedure family allows either retaining E_1 or restoring and then continuing. The retain-only procedure family has budgeted value $V_{E_1}^{B}$.

Theorem 3.2 (Approximate representation restoration)**.** *Under assumption D.1, for $u \in [0, 1]$ and every prior, budget b, and finite action space,*

$$V_{E_1}^{b+d,\mathrm{tr}}(p, u; \mathcal{A}) \geq V_{E_2}^{b}(p, u; \mathcal{A}) - \varepsilon. \tag{10}$$

Applying R and a reference continuation whose utility approaches $V_{E_2}^b$ costs at most $d + b$. Total variation contracts under the shared response rule and bounds expected utility loss by ε in every state for $u \in [0, 1]$. Averaging over any prior preserves this bound, and taking the reference utility to its optimum gives the result. Exact restoration has $\varepsilon = 0$.

3.4 Choosing a representation and continuation

Let x record the request and current serving state, including any active representation. A finite candidate set $\mathcal{J}(x)$ offers representations E_j at acquisition or transition costs $d_x(j) \geq 0$, incurred before their continuation. Before prompt processing, this charge includes that processing and any remaining candidate construction. Retaining an active representation has zero transition cost. All costs must use the same accounting boundary, unit, and serving configuration (Agrawal et al., 2024; Zhong et al., 2024). To reach a utility threshold τ, savings in continuation cost must be weighed against acquisition or transition cost. We therefore minimize their sum:

$$C_x(j; \tau) = d_x(j) + \kappa_{E_j}^{p,u}(\tau; \mathcal{A}), \qquad j_x^{\mathrm{target}}(\tau) \in \operatorname*{argmin}_{j \in \mathcal{J}(x)} C_x(j; \tau). \tag{11}$$

If every target cost is infinite, report the target as infeasible. For a given total budget, obtaining j leaves $b - d_x(j)$ for continuation:

$$j_x^{\text{budget}}(b) \in \operatorname*{argmax}_{\substack{j \in \mathcal{J}(x) \\ d_x(j) \le b}} V_{E_j}^{b-d_x(j)}(p, u; \mathcal{A}). \tag{12}$$

When zero-utility abstention is available, include it as a zero-cost candidate. Otherwise an empty affordable set is infeasible. Exact rebuilding improves the retain-only target cost when $d + \kappa_{E_2}(\tau) < \kappa_{E_1}(\tau)$: the saved continuation cost must exceed the transition cost. For approximate restoration, $d + \kappa_{E_2}(\tau + \varepsilon)$ is a sufficient budget when the reference target is attainable and $\tau + \varepsilon \le 1$. The increased reference target compensates for restoration loss, preserving the desired utility τ.

A finite empirical implementation. To implement the fixed-budget choice, we evaluate one generation procedure on a finite grid of checkpoints $\mathcal{K}$, where K is the scheduled generation count. In synchronized runs, finished responses retain their scores while padding keeps batch slots active through the scheduled checkpoint. Let $\widehat{U}_j(K)$ be mean utility estimated on development tasks and $\widehat{C}_x(j, K)$ the total cost estimated from separate execution profiles. Maximizing over affordable pairs estimates the best budgeted value across candidates and checkpoints:

$$(\widehat{j}_x(b), \widehat{K}_x(b)) \in \operatorname*{argmax}_{\substack{j \in \mathcal{J}(x),\ K \in \mathcal{K} \\ \widehat{C}_x(j,K) \le b}} \widehat{U}_j(K). \tag{13}$$

The rule uses fixed tie-breaking and zero-utility abstention. Although utility may fall at later checkpoints, maximizing over affordable pairs gives a nondecreasing estimated budgeted value.

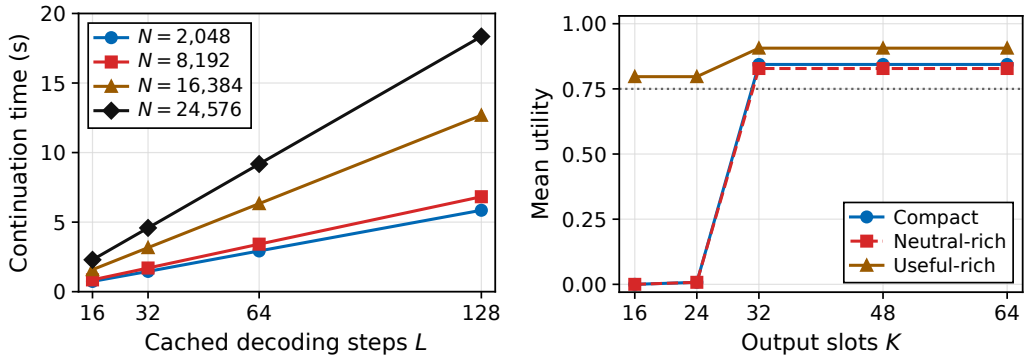
4 Experiments

We present three sets of experiments. First, controlled routing examines the tradeoff between footprint and required work. By varying prompt length and shortcut availability separately, we test when additional information lowers the cost of reaching a common success threshold. We then test joint selection of evidence scope and generation checkpoint in MuSiQue question answering (QA), where more evidence may improve answer quality but leave less budget for generation. Finally, cache rebuilding asks whether replacing an already active cache saves enough continuation cost to cover the transition cost. Appendix E.1 describes development, runtime profiling, and held-out evaluation.

4.1 When Does a Larger Context Reduce Target Cost?

This experiment separates the execution cost of retaining more input from the work needed to reach the target. We use three prompt conditions to vary footprint and access to useful information independently. Each of 128 held-out tasks supplies a route of six edges and asks for its destination and a path certificate, a valid sequence of nodes connecting the start to that destination. Compact supplies the route records in 2,048 tokens. Neutral-rich adds inert material to reach 8,192 tokens. Useful-rich has the same larger footprint but also reveals a certified one-edge shortcut between the start and destination. Thus compact versus neutral-rich varies footprint without adding route information, while the two rich conditions vary shortcut availability at the same footprint.

A response receives utility one if it identifies the correct destination and supplies a complete path using only links in that condition's records, and zero otherwise. Mean utility is the success rate under these condition-specific certificate checks. A generation checkpoint scores K output slots, including the first token produced during prompt processing (prefill). Padding keeps finished responses in their batch slots through the scheduled checkpoint. To compare costs at a common numerical success threshold, we estimate the first checkpoint in $K \in \{16, 24, 32, 48, 64\}$ reaching mean utility $\tau = 0.75$. The experiment uses Qwen2.5-1.5B-Instruct on an RTX 4090 in synchronized batches of four. Independent profiling measures batch continuation latency after prefill for contexts up to 24,576 tokens and cached decoding step counts $L \in \{16, 32, 64, 128\}$. The frozen cost proxy assigns checkpoint K the profile cost at $L = K$, one more cached step than the $K - 1$ steps in the output schedule.



(a) Continuation time by footprint

(b) Mean utility by output checkpoint

Figure 1: The two components of target cost in controlled routing. Panel (a) shows continuation times by context length and cached-step count L. Panel (b) shows mean utility at output checkpoint K, with target $\tau = 0.75$ dotted.

In Figure 1, increasing the context from 2,048 to 24,576 tokens adds 1.554 seconds at $L = 16$ but 12.490 seconds at $L = 128$: a larger footprint makes the continuation work more expensive. All eight profiling sessions have a positive interaction between context length and decoding steps (Appendix E.2). Compact and neutral-rich first attain the utility threshold at $K = 32$, while useful-rich attains it at $K = 16$.

At the first checkpoints reaching the target, useful-rich has a 12.1% lower cost proxy than compact using fitted costs and 41.7% using measured profile cells. The fit preserves the ordering but understates the saving. Appendix E.2 reports individual costs and fit errors. Appendix E.4 compares budgets and shortcut strengths, including selection-boundary sensitivity to the cost approximation. Thus, our target-cost comparison identifies when useful information makes a larger footprint the less costly choice under the profiling proxy.

4.2 Can Joint Allocation of Evidence and Generation Improve QA?

We next ask whether a question's input cost can guide the allocation of evidence and generation within a fixed budget. An evidence scope is the number of ranked paragraphs retained in the prompt. Retaining more paragraphs can improve coverage, whereas retaining fewer can make a later generation checkpoint affordable. Because paragraph lengths vary, the same scope does not leave the same generation budget for every question. We therefore compare joint selection of scope and checkpoint for each request with a policy that assigns one scope to all requests at a given budget.

MuSiQue questions require two to four linked reasoning steps, or hops (Trivedi et al., 2022). For each question, we supply a verified chain of intermediate subquestions and answers from the dataset annotations, excluding the final answer. BM25, a lexical relevance ranker, orders the candidate paragraphs using the question and that chain. All candidate prompts share this chain and differ only in whether they retain the top 2, top 4, top 8, or all ranked paragraphs. We use answer F1 as utility, measuring token overlap with the reference answer through the harmonic mean of precision and recall. Accounting starts after the shared chain and ranking are supplied, charging only candidate-specific prompt processing and scheduled continuation.

Qwen2.5-7B-Instruct runs on an RTX 4090 at batch size one with $K \in \{2, 4, 6, 8, 16, 32\}$ output slots. The first token is produced during prefill. A disjoint 256-question development cohort supplies the mean F1 for each scope j and checkpoint K, recorded in $\widehat{U}_j(K)$. Independent profiling estimates the cost of each pair at the candidate's actual prompt length. For a new request, the joint selector first identifies the pairs that fit the budget and then chooses the one with the highest development F1, as specified by Eq. (13). Requests thus share utility estimates but can receive different choices because their prompt lengths make different pairs affordable.

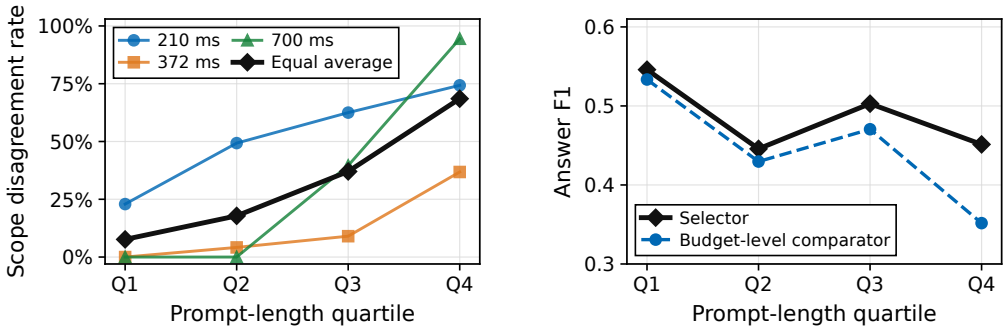
The budget-level comparator uses one development-selected scope per budget, while fixed-scope comparators retain one scope across budgets. Both execute each request's largest affordable checkpoint. The budget-level comparison tests the benefit of choosing evidence scope separately for each request. Policies abstain with zero utility when no checkpoint is affordable. We focus on budgets

where the development-selected scope changes to test whether these shared choices remain suitable for requests with different prompt lengths. Before generation on 576 held-out questions, we fix 210, 372, and 700 ms, where the comparator switches to top 4, top 8, and all paragraphs, respectively. These budgets constrain independently profiled GPU time, not per-request wall-clock deadlines.

Table 1: Answer F1 at prespecified scope-change budgets. Budget-level uses a development-selected scope. Fixed-scope policies use the indicated scope. Differences use unrounded F1.

Budget (ms)	Selector	Budget-level	Top 2	Top 4	Top 8	All	Sel. − Budget
210	0.387	0.330	0.347	0.330	0.158	0.000	+0.058
372	0.510	0.486	0.392	0.496	0.486	0.022	+0.025
700	0.562	0.524	0.394	0.520	0.570	0.524	+0.038
Equal average	0.486	0.446	0.378	0.449	0.404	0.182	+0.040

Against the budget-level comparator, the mean gain across the three equally weighted budgets is $+0.040$ F1 in Table 1. To test whether checkpoint optimization alone explains the gain, we retrospectively optimize the comparator's checkpoint using the same development utility table. None of its 1,728 choices changes. The policies also use identical checkpoints on all 1,162 decisions where their scopes agree. The gain thus comes from scope selection and its affordable checkpoints.



(a) Scope disagreement rate

(b) Answer F1

Figure 2: Request-level adaptation across quartiles of all-paragraph prompt length. Panel (a) shows per-budget rates and their equal average. Panel (b) averages F1 equally over the three budgets.

To identify which requests benefit from adaptation, Figure 2 groups questions into quartiles of all-paragraph prompt length, using the same groups for every policy. The scope disagreement rate is the fraction assigned different scopes by the selector and comparator at a given budget. From the shortest to longest quartile, the average scope disagreement rate rises from 7.6% to 68.5% and the F1 advantage from 0.012 to 0.100. Longer requests benefit most from retaining fewer paragraphs to reach a later checkpoint. The benefit varies with budget: at 700 ms, fixed top 8 scores 0.570 F1 versus the selector's 0.562. Across 27 prespecified nearby budgets, 23 point estimates favor the selector, with four small negative estimates around the largest scope-change budget. Appendix E.5 details the evaluation protocol, cost validation, and broader-budget comparisons.

With LLMLingua-2 compressed prompts added as candidates, the mean F1 difference from the budget-level comparator is -0.001 when compression is charged online and $+0.004$ when it is precomputed (Appendix E.7). A smaller representation therefore need not provide a better affordable answer once acquisition cost and utility are considered. In the primary QA comparison, our joint selector improves mean F1 under the same budgets by adapting scope and checkpoint to each request.

4.3 Offsetting the Cost of Rebuilding an Active Cache

The preceding experiments choose an input before continuation begins. We now consider an already active 8,192-token rich cache and ask whether replacing it can reduce the remaining cost. Rebuilding incurs a one-time transition cost, while a smaller cache can reduce the cost of each subsequent generation step. We evaluate 128 additional routing tasks with paths of 32 edges, grouped into eight blocks of 16. Persistent-rich retains the active cache, while rebuilt-compact discards it and processes the exact 2,048-token prompt containing the query and route records. Both use Qwen2.5-1.5B-Instruct,

synchronized batches of four, and $K \in \{96, 112, 128\}$. Here K counts scheduled output slots, including the first token produced during prefill. The transition cost includes selection, tokenization, cache reset, and reprocessing. Both choices exclude the already incurred initial acquisition cost. The latency proxy adds wall-clock transition time to GPU continuation time.

We first fix the checkpoint to test whether continuation savings cover the transition cost. At matched work $K = 112$, 832.5 ms of continuation savings exceed the 696.5-ms transition cost by 136.0 ms. All 128 rebuilt prompts and complete greedy outputs match the compact reference. This verifies reconstruction and response agreement on these tasks, consistent with exact restoration in Eq. (10). We then ask whether these savings make a later checkpoint affordable within a fixed total budget. At the development-selected budget of 5,825 ms, a retrospective replay selects the largest checkpoint fitting each block's measured mean cost. After rebuilding, six of eight blocks reach $K = 112$ and two reach $K = 96$. Retaining the rich cache permits only $K = 96$, and mean utility increases from 0.008 to 0.625 after rebuilding. The later checkpoint accommodates certificates requiring up to 107 tokens. Appendix E.6 details checkpoint and transition costs. In this replay, our post-acquisition selection rule improves utility within the same total budget after accounting for the cache transition cost.

5 Related Work

Information value with costly computation. Blackwell comparison characterizes decision value under unrestricted downstream processing (Blackwell, 1953). Usable and decodable information are defined relative to restricted prediction rules (Xu et al., 2020; Dubois et al., 2020). Costly computation and resource-rational models account for computational resources (Halpern & Pass, 2010; Lieder & Griffiths, 2020). Expanding the available information can raise processing costs (Park et al., 2026b). Park et al. (2026a) relate the cost of accessing observation channels to value across Bayesian decision problems. We apply this perspective to behavioral targets, separating required work from execution cost and including acquisition and transition costs. Approximate restoration also connects to statistical deficiency and computational simulability, which compare information sources by how well one can reproduce another (van Rooyen & Williamson, 2014; Akdemir, 2025).

Constructing and reducing context. RAG supplies external evidence (Lewis et al., 2020; Ram et al., 2023), while adaptive retrieval balances inference quality and computational overhead (Asai et al., 2024; Jeong et al., 2024). Prompt compression shortens input (Jiang et al., 2023; Pan et al., 2024; Xu et al., 2024), while KV-cache eviction or compression reduces retained state (Zhang et al., 2023; Li et al., 2024). Acquisition and transition costs reduce the budget left for continuation.

Execution cost and allocation of inference. Systems analyses explain how sequence length, attention implementation, and memory management affect execution cost (Pope et al., 2023; Dao et al., 2022; Kwon et al., 2023). Test-time scaling allocates computation according to the prompt (Snell et al., 2025), while retrieved information can change the return to additional inference (Fang et al., 2026). Our break-even conditions compare information and computation choices by their cost of reaching a common target, allowing the required work to differ.

6 Conclusion

We introduced effective cost to compare representations by the least budget needed to reach a common target. Changing the retained input can affect both required work and the physical cost of executing that work. Our selection rules account for these effects together with acquisition and transition costs when choosing a representation and continuation. Controlled routing showed that useful information can lower continuation cost at a common success threshold despite a larger footprint. In MuSiQue QA, jointly selecting evidence scope and generation checkpoint improved mean answer F1 over the budget-level comparator under the same profiled budgets. Cache rebuilding showed that continuation savings can cover transition cost and, in retrospective replay, enable higher utility within the same total budget. Together, these findings show the benefit of comparing representations through both required work and physical execution cost. Effective cost and the resulting selection rules thus provide a common basis for obtaining lower target cost or higher utility within a fixed budget.

AI USE STATEMENT

Generative AI tools assisted with manuscript organization and editing, mathematical consistency checks, code inspection, figure preparation, and reanalysis of archived experimental outputs. These tools are not an independent source of empirical evidence or a substitute for scientific validation. Responsibility for the study and its final claims remains with the authors.

REPRODUCIBILITY STATEMENT

Appendices A–D state the assumptions and proofs. Appendix E specifies task construction, measurement boundaries, development and held-out cohorts, and evaluation procedures. The accompanying package contains frozen task definitions, raw session outputs, analysis code, and figure inputs. Its results index maps reported findings to those artifacts. The additional numerical audit reuses archived outputs and does not rerun model inference.

REFERENCES

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, 2024.

Deniz Akdemir. Approximate computation via Le Cam simulability, 2025. URL https://arxiv.org/abs/2512.24860. arXiv:2512.24860.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *International Conference on Learning Representations*, 2024.

David Blackwell. Equivalent comparisons of experiments. *The Annals of Mathematical Statistics*, 24(2):265–272, 1953. doi: 10.1214/aoms/1177729032.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems*, volume 35, pp. 16344–16359, 2022. doi: 10.52202/068431-1189.

Yann Dubois, Douwe Kiela, David J. Schwab, and Ramakrishna Vedantam. Learning optimal representations with the decodable information bottleneck. In *Advances in Neural Information Processing Systems*, volume 33, pp. 18674–18690, 2020.

Alex Fang, Thomas Voice, Ruoming Pang, Ludwig Schmidt, and Tom Gunter. Reusing pre-training data at test time is a compute multiplier. In *International Conference on Learning Representations*, 2026.

Joseph Y. Halpern and Rafael Pass. I don't want to think about it now: Decision theory with costly computation. In *Proceedings of the Twelfth International Conference on the Principles of Knowledge Representation and Reasoning*, pp. 182–190, 2010. URL https://cdn.aaai.org/ocs/1353/1353-7408-1-PB.pdf.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C. Park. Adaptive-RAG: Learning to adapt retrieval-augmented large language models through question complexity. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 7036–7050, 2024. doi: 10.18653/v1/2024.naacl-long.389.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. LLMLingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 13358–13376, 2023. doi: 10.18653/v1/2023.emnlp-main.825.

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.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pp. 9459–9474, 2020.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. SnapKV: LLM knows what you are looking for before generation. In *Advances in Neural Information Processing Systems*, volume 37, pp. 22947–22970, 2024. doi: 10.52202/079017-0722.

Falk Lieder and Thomas L. Griffiths. Resource-rational analysis: Understanding human cognition as the optimal use of limited computational resources. *Behavioral and Brain Sciences*, 43:e1, 2020. doi: 10.1017/S0140525X1900061X.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Dongmei Zhang. LLMLingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 963–981, 2024. doi: 10.18653/v1/2024.findings-acl.57.

Jonghyun Park, Sudipta Sarangi, and Hector Tzavellas. Free disposal with costly access, 2026a. URL `https://ssrn.com/abstract=7263479`. SSRN working paper.

Jonghyun Park, Sudipta Sarangi, and Hector Tzavellas. Information overload under rational learning, 2026b. URL `https://ssrn.com/abstract=6592239`. SSRN working paper.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Anselm Levskaya, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. In *Proceedings of Machine Learning and Systems*, volume 5, pp. 606–624, 2023.

Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. In-context retrieval-augmented language models. *Transactions of the Association for Computational Linguistics*, 11:1316–1331, 2023. doi: 10.1162/tacl_a_00605.

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.

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

Brendan van Rooyen and Robert C. Williamson. Le Cam meets LeCun: Deficiency and generic feature learning, 2014. URL `https://arxiv.org/abs/1402.4884`. arXiv:1402.4884.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. RECOMP: Improving retrieval-augmented LMs with context compression and selective augmentation. In *International Conference on Learning Representations*, 2024.

Yilun Xu, Shengjia Zhao, Jiaming Song, Russell Stewart, and Stefano Ermon. A theory of usable information under computational constraints. In *International Conference on Learning Representations*, 2020.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H2O: Heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems*, volume 36, pp. 34661–34710, 2023. doi: 10.52202/075280-1506.

Yinmin Zhong, Shengyu Liu, Junda Chen, Jianbo Hu, Yibo Zhu, Xuanzhe Liu, Xin Jin, and Hao Zhang. DistServe: Disaggregating prefill and decoding for goodput-optimized large language model serving. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pp. 193–210, 2024.

A WHY TARGET COSTS CHARACTERIZE BUDGETED VALUE

The argument has two parts. Linear decision objectives identify inclusion of closed convex capability sets. A closed budget boundary then identifies those sets with the behaviors whose effective cost is at most that budget.

Assumption A.1 (Nested compact capability sets)**.** *For each representation E, finite nonempty action space $\mathcal{A}$, and $b \geq 0$, $\widehat{\mathcal{Q}}_E^b(\mathcal{A})$ is nonempty, compact, and convex. For $b_1 \leq b_2$, $\widehat{\mathcal{Q}}_E^{b_1}(\mathcal{A}) \subseteq \widehat{\mathcal{Q}}_E^{b_2}(\mathcal{A})$.*

Closedness and convexity follow from Eq. (1). Finite Θ and $\mathcal{A}$ make a closed subset of the channel simplex compact. Nonemptiness can be implemented by a zero-cost baseline action such as abstention, when that action belongs to the decision problem. If all admissible procedures have strictly positive execution costs, the zero-budget capability set is empty, so this assumption is not satisfied. Randomization here selects among individually affordable rules, with its overhead already included or negligible. It does not replace a hard budget by an average-cost constraint.

Assumption A.2 (Budget-closed feasibility)**.** *For every E, $\mathcal{A}$, and $b \geq 0$,*

$$\widehat{\mathcal{Q}}_E^b(\mathcal{A}) = \bigcap_{\delta > 0} \widehat{\mathcal{Q}}_E^{b+\delta}(\mathcal{A}).$$

Taking a closure at each fixed budget does not by itself imply this assumption. A target could be feasible at every budget above c but not at c, making its effective cost c without being feasible there. Closed capability sets alone do not rule this out. Budget closure ensures that finite effective cost also identifies a feasible budget.

Theorem A.1 (Value dominance and capability inclusion)**.** *Under assumption A.1, fix b and finite $\mathcal{A}$. Then $V_{E_1}^b(p, u; \mathcal{A}) \geq V_{E_2}^b(p, u; \mathcal{A})$ for every prior and bounded utility if and only if $\widehat{\mathcal{Q}}_{E_2}^b(\mathcal{A}) \subseteq \widehat{\mathcal{Q}}_{E_1}^b(\mathcal{A})$.*

Proof. Inclusion makes every feasible behavior under E_2 feasible under E_1, proving the forward value comparison. Conversely, suppose that $Q^\star \in \widehat{\mathcal{Q}}_{E_2}^b$ is outside $\widehat{\mathcal{Q}}_{E_1}^b$. In this finite-dimensional space, strict separation of a point from a compact convex set supplies a score matrix H. Writing $\langle H, Q \rangle = \sum_{\theta,a} H_{\theta a} Q_{\theta a}$, it satisfies

$$\langle H, Q^\star \rangle > \max_{Q \in \widehat{\mathcal{Q}}_{E_1}^b} \langle H, Q \rangle.$$

Let $m = |\Theta|$, take $p_\theta = 1/m$, and define $u(\theta, a) = mH_{\theta a}$. This utility is bounded because its domain is finite, and its expected utility is the separating score. Consequently $V_{E_2}^b \geq \langle H, Q^\star \rangle > V_{E_1}^b$, contradicting the claimed comparison over all objectives. □

Lemma A.1 (Exact cost sublevels)**.** *Under assumptions A.1 and A.2, for finite b,*

$$Q \in \widehat{\mathcal{Q}}_E^b(\mathcal{A}) \quad \Longleftrightarrow \quad \bar{\kappa}_E(Q; \mathcal{A}) \leq b.$$

Proof. Membership bounds the infimum from above. Conversely, if $c = \bar{\kappa}_E(Q; \mathcal{A}) \leq b$, then $c < \infty$. For each $\delta > 0$, the infimum definition supplies a feasible budget below $c + \delta$. Nesting puts Q in $\widehat{\mathcal{Q}}_E^{b+\delta}$, and budget closure puts it in $\widehat{\mathcal{Q}}_E^b$. □

Proof of Theorem 2.1. Suppose target costs are ordered and $Q \in \widehat{\mathcal{Q}}_{E_2}^b$. Its E_2 cost is finite and at most b, so its E_1 cost is also at most b. By Lemma A.1, $Q \in \widehat{\mathcal{Q}}_{E_1}^b$. Theorem A.1 gives value dominance. Conversely, value dominance gives capability inclusion at every budget. If Q has finite E_2 cost c, it is feasible under E_2, and hence under E_1, at c. Therefore $\bar{\kappa}_{E_1}(Q) \leq c$.

If $\bar{\kappa}_{E_1}(Q) > \bar{\kappa}_{E_2}(Q)$, choose a finite budget between the two costs. The target is feasible only in the second capability set. The same separation argument supplies a decision objective with strictly higher E_2 value. This strict consequence requires no Blackwell ordering of the representations. □

Proof of Corollary 2.1. Fix an objective and a target τ. Its feasible budget set $A_E(\tau) = \{b : V_E^b \geq \tau\}$ is upward closed. If its infimum c is finite, take $b_n \downarrow c$ with $b_n \in A_E(\tau)$. The sets

$$F_n = \{Q \in \widehat{\mathcal{Q}}_E^{b_n} : \langle u, Q\rangle_p \geq \tau\}$$

are nonempty nested compact sets. Their intersection is nonempty, and budget closure places every element of that intersection in $\widehat{\mathcal{Q}}_E^c$. Thus $c \in A_E(\tau)$ and $V_E^b \geq \tau \iff \kappa_E^{p,u}(\tau) \leq b$. Value dominance gives inclusion of the utility-target budget sets and hence the reversed cost order. Conversely, at each b choose $\tau = V_{E_2}^b$. The cost order gives $\kappa_{E_1}^{p,u}(\tau) \leq \kappa_{E_2}^{p,u}(\tau) \leq b$, hence $V_{E_1}^b \geq \tau$. □

B From Required Work to Physical Cost

The programs in this appendix retain the selected input representation during continuation. The continuation cost function can account for the growing generated suffix, as the quadratic term in Eq. (8) does, while replacing the initial representation requires the transition-enabled family. Write $\mathcal{W}_E^r = \mathrm{cl\,co}\{ED_{E,P} : P \in \mathcal{P}(\mathcal{A}),\ w(P) \leq r\}$ for the capability set at work allowance r, using the randomization and limiting-behavior conventions of Eq. (1).

Assumption B.1 (Execution model based on footprint and work)**.** *For a common program language with nonnegative finite continuation work,*

$$\widehat{\mathcal{Q}}_E^b(\mathcal{A}) = \mathrm{cl\,co}\{ED_{E,P} : P \in \mathcal{P}(\mathcal{A}),\ \Phi(s(E), w(P)) \leq b\}.$$

For every reachable footprint s, $\Phi(s, \cdot) : [0, \infty) \to [0, \infty)$ is continuous and strictly increasing, satisfies $\Phi(s, 0) = 0$, and tends to $+\infty$ as work tends to $+\infty$. For fixed work it is nondecreasing in footprint. Set $\Phi(s, +\infty) = +\infty$.

The unboundedness condition is substantive. A bounded increasing continuation cost function, such as $1 - e^{-r}$, could admit arbitrarily large work at a finite physical budget. Closure could then make a target attainable at that budget even though it belongs to no finite work sublevel. Continuity and strict monotonicity alone would not justify the extended-real factorization.

Lemma B.1 (Effective cost from required work)**.** *Under assumption B.1, effective cost factors as*

$$\bar{\kappa}_E(Q; \mathcal{A}) = \Phi(s(E), \rho_E(Q)),$$

including infinite values.

Proof. Fix E and let g_E be the inverse of $\Phi(s(E), \cdot)$. The assumptions make this a continuous increasing bijection of $[0, \infty)$ onto itself. Strict increase gives $w(P) \leq r \iff \Phi(s(E), w(P)) \leq \Phi(s(E), r)$. Thus the same raw behaviors appear before convexification and closure:

$$\widehat{\mathcal{Q}}_E^{\Phi(s(E),r)} = \mathcal{W}_E^r, \qquad \widehat{\mathcal{Q}}_E^b = \mathcal{W}_E^{g_E(b)}.$$

If $S_Q = \{r : Q \in \mathcal{W}_E^r\}$ is nonempty, it is upward closed. Writing $c = \inf S_Q < \infty$, monotonicity gives $\inf_{r \in S_Q} \Phi(s(E), r) \geq \Phi(s(E), c)$, while a sequence in S_Q decreasing to c gives the reverse inequality by continuity. The exact sublevel correspondence identifies the left-hand side with effective cost. If S_Q is empty, the bijection shows that Q belongs to no finite physical-budget set either. Both sides are infinite. If budget closure also holds, the same sublevel identity implies work-budget closure and attainment of every finite required-work threshold. □

For utility thresholds define

$$W_E^r(p, u; \mathcal{A}) = \max_{Q \in \mathcal{W}_E^r} \langle u, Q\rangle_p, \tag{14}$$

which is nondecreasing because increasing r retains every earlier program, and

$$\rho_E^{p,u}(\tau; \mathcal{A}) = \inf\{r \geq 0 : W_E^r(p, u; \mathcal{A}) \geq \tau\}. \tag{15}$$

The maximum is well-defined under assumptions A.1 and B.1, using the sublevel correspondence above.

Corollary B.1 (Work allowance for utility thresholds)**.** *Under assumptions A.1, A.2 and B.1,*

$$\kappa_E^{p,u}(\tau;\mathcal{A}) = \Phi(s(E), \rho_E^{p,u}(\tau;\mathcal{A})). \tag{16}$$

For a fixed objective, E_1 dominates E_2 at every physical budget if and only if

$$\rho_{E_1}^{p,u}(\tau;\mathcal{A}) \le T_\Phi(s(E_1), s(E_2); \rho_{E_2}^{p,u}(\tau;\mathcal{A})) \quad \textit{for every } \tau. \tag{17}$$

Here $T_\Phi(s_1, s_2; +\infty) = +\infty$.

Proof. The exact sublevels give $V_E^b = W_E^{g_E(b)}$. Applying the same increasing continuous change of budget to the first level attaining utility τ proves Eq. (16). Empty attainable sets correspond on both sides. By Corollary 2.1, fixed-task dominance is equivalent to an ordering of these costs. For each finite reference target cost, apply the inverse continuation cost function at $s(E_1)$ to obtain Eq. (17). If the reference cost is infinite, both the cost-order constraint and its allowance version are vacuous. □

Proof of Theorem 3.1. The allowance is the unique nonnegative solution of $\Phi(s_1, T_\Phi(s_1, s_2; r)) = \Phi(s_2, r)$ for finite r. Consequently,

$$\bar\kappa_{E_1}(Q) \le \bar\kappa_{E_2}(Q) \quad \Longleftrightarrow \quad \rho_{E_1}(Q) \le T_\Phi(s(E_1), s(E_2); \rho_{E_2}(Q)) \tag{18}$$

whenever the reference cost is finite. This follows by substituting Lemma B.1 and inverting the strictly increasing continuation cost function. Apply Theorem 2.1 across all such targets. A Blackwell relation is not needed for this equivalence. It becomes relevant when interpreting why the required work might decrease. □

B.1 Semantic recovery is an additional execution assumption

Blackwell dominance $E_1 \succeq_{\mathrm{B}} E_2$ means that some channel G gives $E_2 = E_1 G$ (Blackwell, 1953). This is a statement about observations, not a bound on the cost or work of implementing G.

Assumption B.2 (Semantic compiler closure)**.** *If $E_2 = E_1 G$, every program P has a compiled counterpart $\mathsf{C}_G(P)$ satisfying*

$$E_1 D_{E_1, \mathsf{C}_G(P)} = E_2 D_{E_2, P}, \qquad w(\mathsf{C}_G(P)) \le w(P).$$

The compiled program retains the physical representation of E_1.

The compiled program must reproduce the entire behavior under E_2, including any randomness, without increasing the declared work. This is an execution assumption in addition to the channel relation $E_2 = E_1 G$. The program retains the input state associated with E_1, so eviction or rebuilding remains a separately charged transition.

Proposition B.1 (Semantic simulation)**.** *Under assumption B.2, $E_1 \succeq_{\mathrm{B}} E_2$ implies $\mathcal{W}_{E_2}^r \subseteq \mathcal{W}_{E_1}^r$ for every $r \ge 0$. Hence $\rho_{E_1}(Q) \le \rho_{E_2}(Q)$ for every target behavior attainable from E_2 at finite work.*

Proof. Each raw behavior in the second representation's work sublevel has an identical compiled behavior in the first representation's work sublevel. Convexification and closure preserve inclusion. Taking the least work allowances containing a target gives the required-work inequality. □

B.2 Local changes and the growing footprint penalty

The work allowance compares finite changes between representations. For a smoothly varying family, a local version identifies the work reduction needed to compensate for a small increase in footprint.

Proposition B.2 (Marginal tradeoff between footprint and work)**.** *For a fixed target Q, let $s(t) = s(E_t)$ and $\rho(t) = \rho_{E_t}(Q)$. At a point t_0, suppose both functions are finite and differentiable, and Φ is continuously differentiable on an open neighborhood of $(s(t_0), \rho(t_0))$, with $\Phi_r > 0$ there. Let Φ_s and Φ_r denote partial derivatives with respect to footprint and work. Writing $c(t) = \Phi(s(t), \rho(t))$, at t_0,*

$$c'(t_0) \le 0 \quad \Longleftrightarrow \quad -\rho'(t_0) \ge \frac{\Phi_s(s(t_0), \rho(t_0))}{\Phi_r(s(t_0), \rho(t_0))} s'(t_0).$$

Proof. The assumed differentiability permits the chain rule, $c' = \Phi_s s' + \Phi_r \rho'$. Divide by the strictly positive Φ_r and rearrange. □

Differentiability of a representation path alone does not imply differentiability of its required work for a fixed target. At a program switch, checkpoint jump, or other nonsmooth point, this proposition does not apply. The finite allowance comparison remains the appropriate test.

We next ask when a larger footprint becomes more costly as continuation work grows. The following condition ensures that the cost gap between two fixed footprints cannot decrease with work.

Proposition B.3 (Increasing footprint penalty with work)**.** *Fix $s_1 > s_2$ and an interval $I \subset (0, \infty)$. If Φ is twice continuously differentiable on a neighborhood of $[s_2, s_1] \times I$ and $\Phi_{sr} \geq 0$ there, then $B_\Phi(r; s_1, s_2) = \Phi(s_1, r) - \Phi(s_2, r)$ is nondecreasing on I. It is strictly increasing if $\int_{s_2}^{s_1} \Phi_{sr}(s, r)\, ds > 0$ for every $r \in I$.*

Proof. Differentiate and use the fundamental theorem of calculus: $B'_\Phi(r) = \Phi_r(s_1, r) - \Phi_r(s_2, r) = \int_{s_2}^{s_1} \Phi_{sr}(s, r)\, ds$. □

C Why Equivalent Information Can Have Different Costs

Write $E_1 \sim_{\mathrm{B}} E_2$ when the representations are Blackwell-equivalent: each observation channel can be recovered from the other by a stochastic map. Such representations preserve the same information but may retain physical states of different sizes. The results below isolate the cost of that difference.

Proposition C.1 (Different costs for equivalent information)**.** *Under assumptions A.1, A.2 and B.1, suppose $E_1 \sim_{\mathrm{B}} E_2$ and a target Q satisfies $\rho_{E_1}(Q) = \rho_{E_2}(Q) = r < \infty$ but $\Phi(s(E_1), r) > \Phi(s(E_2), r)$. Then some finite continuation budget and bounded decision objective give strictly higher budgeted value under E_2.*

Proof. Factorization makes the two effective costs strictly ordered. The strict separation consequence of Theorem 2.1 supplies the objective. The equivalence assumption gives the information-theoretic interpretation. The cost implication itself also holds for any equal-work pair. □

Assumption C.1 (Neutral-extension coverage)**.** *For every pair of reachable footprints $s_1 > s_2$ and every $r > 0$, there exist Blackwell-equivalent representations with $s(E_i) = s_i$ and a target Q with $\rho_{E_1}(Q) = \rho_{E_2}(Q) = r$.*

This coverage assumption ensures that an equal-work comparison is available at every footprint and positive work level. Without it, Φ could vary at pairs of footprint and work that no target realizes, so equal budgeted values would not force footprint independence. The following characterization uses coverage for this reverse implication.

Theorem C.1 (Footprint-independent execution)**.** *Under assumptions A.1, A.2, B.1, B.2 and C.1, the following are equivalent on the reachable domain of footprint and work: (i) every Blackwell-equivalent pair has identical value at every budget for every finite decision problem, and (ii) $\Phi(s, r) = \psi(r)$ is independent of footprint.*

Proof. If the continuation cost function is footprint-independent, semantic simulation in both directions gives identical closed work sublevels, hence identical budgeted capability sets and budgeted values. Conversely, footprint dependence, monotonicity, and $\Phi(s, 0) = 0$ supply $s_1 > s_2$ and $r > 0$ with strictly different continuation costs. Neutral-extension coverage gives a Blackwell-equivalent equal-work pair at these values. Proposition C.1 supplies a budget and objective with unequal value, contradicting (i). □

D Cached Decoding, Transitions, and Finite Candidate Frontiers

D.1 Dense-decoding domain and break-even calculation

For integer $K \geq 0$, the stepwise sum is

$$\sum_{t=0}^{K-1} [\alpha(N+t)+\beta] = \alpha NK + \tfrac{\alpha}{2}K(K-1) + \beta K.$$

The empty sum is zero. On the continuous extension $N \geq 1$, $K \geq 0$, $\Phi_K = \alpha(N+K-\frac{1}{2})+\beta > 0$, $\Phi_N = \alpha K \geq 0$, and $\Phi_{NK} = \alpha > 0$. Moreover $\Phi(N,K) \to \infty$ as $K \to \infty$. Thus this extension satisfies the conditions on the continuation cost function in assumption B.1. On the integer domain, the corresponding increments are $\alpha(N+K)+\beta$ per added step and αK per added prefix token. The restriction on N matters: extending the same polynomial to all $N, K \geq 0$ would not preserve nonnegativity or work monotonicity near the origin.

Proof of Corollary 3.1. Subtract the cost under E_2 from that under E_1 and collect the saved steps:

$$\Phi(N+M,K_1) - \Phi(N,K_2) = \alpha M K_1 - (K_2-K_1)[\alpha N + \tfrac{\alpha}{2}(K_1+K_2-1)+\beta].$$

For $K_1 > 0$, setting the difference to zero and dividing by αK_1 gives Eq. (9). Equivalently,

$$\Phi(N+M,K_1) - \Phi(N,K_2) = \alpha K_1 (M - M^\star). \tag{19}$$

Since $\alpha K_1 > 0$, the sign is precisely the sign of $M - M^\star$. When $K_1 = 0$, the continuation cost under E_1 is zero and its comparison follows directly, without using $M^\star$. □

This model begins after the initial representation has been formed. A profile that includes prefill and its first predicted token uses a different accounting boundary. Its cost must be calibrated at that boundary rather than silently identified with K cached forwards.

D.2 Serial restoration in a transition-enabled procedure family

Assumption D.1 (Serial execution of a representation change)**.** *A transformation R maps observations from E_1 to the reference channel E_2's observation space, inducing $\widetilde{E}_2 = E_1 R$. Producing these observations and preparing their continuation state has total cost at most d. For every response rule executable from E_2 within b, the same rule is executable after R within total cost $d+b$. Such composed rules, with their admissible randomizations and limits, belong to the transition-enabled procedure family defining $V_{E_1}^{d+b,\mathrm{tr}}$. The retain-only procedure family is also included. In addition,*

$$\sup_{\theta \in \Theta} d_{\mathrm{TV}}((\widetilde{E}_2)_\theta, (E_2)_\theta) \leq \varepsilon. \tag{20}$$

The execution clause and the error bound are separate assumptions. Similar observation distributions alone do not establish that a continuation fits the same physical budget after a cache transformation. Neither assumption identifies transition-enabled cost with a single continuation cost function at a fixed footprint.

Proof of Theorem 3.2. The maximizing behavior may be a limit rather than the output of a single executable rule. For any $\delta > 0$, continuity and linearity of expected utility allow a raw feasible response rule or its admissible randomization D with expected utility at least $V_{E_2}^b - \delta$ under E_2. Execute R followed by D. This composed procedure is feasible within $d+b$ in the transition-enabled family. Applying the same response rule contracts total variation, so each state's action distributions $(E_1 RD)_\theta$ and $(E_2 D)_\theta$ differ in total variation by at most ε. For a utility in $[0,1]$, the difference of its expectations is at most ε. Averaging over p preserves this bound. Consequently

$$V_{E_1}^{d+b,\mathrm{tr}} \geq \langle u, E_1 RD \rangle_p \geq V_{E_2}^b - \delta - \varepsilon.$$

Let $\delta \downarrow 0$. □

For exact restoration, the transition-enabled target cost is at most $d + \kappa^{p,u}_{E_2}(\tau)$, the cost of restoration followed by continuation. The rebuilding comparison in Section 3.4 instead uses the distinct retain-only cost $\kappa^{p,u}_{E_1}(\tau)$, which can exceed that total cost. For approximate restoration with attainable reference target $\tau + \varepsilon$, compactness and budget closure give reference budgeted value at least $\tau + \varepsilon$ at its target cost. The theorem then gives budgeted value at least τ after paying d.

D.3 Candidate optimality and empirical work

For a finite candidate set, obtaining E_j costs $d_x(j)$ and is followed by its retain-only continuation. Serial accounting makes the total target cost exactly $d_x(j) + \kappa^{p,u}_{E_j}(\tau)$. Taking the least of these costs proves Eq. (11). At total budget b, an affordable candidate gives budgeted value $V^{b-d_x(j)}_{E_j}$. Maximizing over candidates proves Eq. (12). External randomization between individually affordable candidates cannot improve their maximum expected utility, so choosing a best candidate also optimizes over such mixtures.

If the decision problem permits zero-utility abstention, include it as a zero-cost candidate. Otherwise an empty affordable set means there is no feasible action under the specified model. When every candidate has infinite cost for a target, no optimizer with finite target cost exists. The operational output is infeasibility rather than an arbitrary tied choice. Deterministic tie-breaking can be applied whenever finite optima are attained.

To compare measured performance at a common utility target, let $U_E(K)$ be the mean utility of a particular generation procedure at checkpoint K in a finite grid $\mathcal{K}$. Its empirical required work is the earliest tested checkpoint that reaches threshold τ:

$$\rho^{\mathrm{emp}}_E(\tau) = \inf\{K \in \mathcal{K} : U_E(K) \geq \tau\}, \tag{21}$$

with the smallest qualifying checkpoint selected, or $+\infty$ if none qualifies. The minimized family here consists of the tested checkpoints of that procedure. A later output prefix can reduce utility, so $U_E(K)$ need not be monotone. With earlier stopping and zero-utility abstention, the best utility at work allowance r within this grid is the nondecreasing envelope $\max(\{0\} \cup \{U_E(K) : K \in \mathcal{K},\ K \leq r\})$. The included zero also defines the envelope when no checkpoint is affordable. For positive targets, the first work allowance at which this envelope reaches τ equals $\rho^{\mathrm{emp}}_E(\tau)$. For $\tau \leq 0$, the abstention-enabled family reaches the target at zero work. The empirical rule in Eq. (13) explicitly compares all affordable pairs and does not assume that mean utility increases with the generation checkpoint. Errors in the profiled cost function can change affordability, and errors in the utility table can change the chosen pair. The theoretical candidate optimality therefore does not constitute a guarantee for the plug-in policy or a hard-deadline guarantee for profiled time.

E Experimental Protocols and Additional Results

E.1 Experimental Design and Separation of Calibration and Evaluation

The experiments estimate two quantities separately: the scheduled work needed for a target and the physical cost of executing that work. Selection combines them with acquisition or transition costs through the target-cost and fixed-budget rules in Eqs. (11) and (13). Generation checkpoints are the prespecified values of scheduled generation count K at which utility is evaluated. Each study below defines its counting and accounting conventions. Table 2 locates the protocols and additional results.

The accounting boundary follows the decision being evaluated. Controlled routing starts after prefill and charges continuation only. MuSiQue selection starts after the shared ranking and chain construction, but includes candidate-specific prompt processing and continuation. Cache rebuilding starts with the rich cache already active and adds the full measured transition cost to subsequent continuation. The compression experiment separately varies whether the measured compressor cost is paid online. Competing policies within each comparison use the same cost accounting and serving configuration.

For selection experiments, calibration and held-out evaluation are separated, with physical cost profiling performed independently of task outcomes. The controlled task generator, utility, generation checkpoints, target, and runtime schedules are fixed before outcome analysis, and file hashes record

Table 2: How the appendix experiments support the main comparisons.

Appendix	Measurement or comparison	Role in the main evidence
E.2	Cost across context length and decoding work	Provides the execution costs in Figure 1a
E.3	Work needed for a shared numerical success threshold	Supplies Figure 1b and routing protocols
E.4	Selection across budgets, footprints, and shortcut strengths	Extends Section 4.1 from cost comparison to selection
E.5	Evidence-scope selection and supporting comparisons	Tests and qualifies pre-acquisition selection in Section 4.2
E.6	Rebuilding cost and affordable continuation	Tests post-acquisition selection in Section 4.3
E.7	Compressed candidates with and without a compression cost	Examines acquisition costs for additional candidates
E.8	Evidence and chain contrasts across model sizes	Checks whether context effects persist across models

their versions. For the grid that varies footprint and shortcut strength, a single manifest also records the prompts, model artifacts, analysis code, and fitted cost function. For MuSiQue, the development utility table and independently profiled cost function determine the three scope-change budgets and the scope and checkpoint selected for each request. These selections and the retrieval rankings are recorded before generation on 576 held-out questions. Separate 256-question cohorts cover a broader budget range and the expanded compression candidate set. The rebuild procedure and its accounting boundary are fixed before its outcomes are produced. Dataset indices, prompt templates, candidate transformations, and model identities are likewise fixed before the corresponding runs. Appendix E.8 compares context conditions on development data. It is not another held-out selector evaluation.

E.2 Calibrating Continuation Cost across Context Lengths

This experiment measures the execution cost in the model of footprint and work. It tests whether a larger active context raises the marginal cost of another decoding step, corresponding to $\Phi_{NK} > 0$ in Eq. (8). The fitted function predicts continuation cost from context length and scheduled decoding steps for the routing experiments below.

We profile context lengths $N \in \{2{,}048, 8{,}192, 16{,}384, 24{,}576\}$ and cached decoding steps $L \in \{16, 32, 64, 128\}$. The primary configuration uses an RTX 4090, the fixed Qwen2.5-1.5B-Instruct model snapshot, PyTorch scaled dot-product attention (SDPA), bfloat16 arithmetic, and batches of four with a fixed decoding schedule. Each session contains two warm-up blocks and six measurement blocks over the complete 4×4 grid. A prespecified balanced schedule changes order across blocks and is validated before execution. A cell is one pair of context length and cached decoding step count. A valid session has exclusive GPU use, complete cell coverage, passing reference timing checks at the start and end, and complete raw records for the schedule, CUDA events, wall time, active processes, clocks, power, temperature, and memory. The session is the independent unit. Repetitions within a session estimate its cell medians. For each session s, let $T_s^{\mathrm{cell}}(N, L)$ denote its observed cell median. We fit

$$T_s^{\mathrm{cell}}(N, L) = a_s + g_s N + b_s L + q_s \frac{L(L-1)}{2} + \alpha_s NL + e_{s,N,L}. \tag{22}$$

The fitted prediction $\widetilde{T}_s(N, L)$ is the right-hand side without the residual $e_{s,N,L}$. Here the subscript s indexes profiling sessions, rather than representation footprint. The coefficient a_s is the intercept, g_s captures dependence on initial context length, and b_s captures the component proportional to the number of decoding steps. The coefficient q_s captures the additional cost of attending to previously generated tokens as decoding proceeds, and $e_{s,N,L}$ is the residual. The interaction coefficient α_s measures the additional cost per decoding step contributed by each token of initial context. Unlike the analytic model, this empirical fit estimates the coefficients of $L(L-1)/2$ and NL separately. Equation (22) is fit separately in each of eight sessions, and the mean fitted coefficients define the routing cost function. The primary summary reports the eight α_s estimates and their mean. The sign count and leave-one-session-out range describe stability. We also compare the extra cost of the larger

context at long and short continuations directly from observed cell medians:

$$D_s = [T_s^{\text{cell}}(24576, 128) - T_s^{\text{cell}}(2048, 128)] - [T_s^{\text{cell}}(24576, 16) - T_s^{\text{cell}}(2048, 16)] \quad (23)$$

A positive D_s means that the larger context adds more cost at the longer of these two checkpoints. This extreme-cell increasing-differences contrast and the corresponding relative slowdown are secondary summaries. A pooled regression, batch-one measurements, and decomposition into attention-adjacent and remaining runtime components provide supplementary checks.

All eight session-level interaction estimates are positive. Their mean is 4.527 microseconds per context token per decoding step. The mean extreme-cell increasing-differences contrast is 10.935 seconds, and the relative slowdown from $N = 2{,}048$ to $N = 24{,}576$ at $L = 128$ is 213.4%, relative to the smaller context. These measurements support the higher execution cost of work at larger footprints. Combining this fitted cost function with the target-reaching checkpoints in Appendix E.3 yields the target-cost comparison in the main text.

Sensitivity to cost-model approximation. To check whether approximation error changes the target-cost ordering, Table 3 compares fitted costs with means of the eight session-level cell medians at the target-reaching checkpoints. This retrospective reanalysis uses the archived calibration observations. The original policies and task outcomes remain fixed.

Table 3: Routing target-cost proxies using profiles at $L = K$ cached decoding steps. Relative error is fitted prediction divided by cell mean, minus one.

Representation	N	K	Fitted (ms)	Cell mean (ms)	Error
Compact	2,048	32	1,172.6	1,457.4	-19.5%
Neutral-rich	8,192	32	2,062.2	1,700.7	$+21.3\%$
Useful-rich	8,192	16	1,030.5	849.9	$+21.2\%$

Useful-rich has lower target cost than compact under both calculations: 12.1% under the fitted cost function and 41.7% under the observed-cell summary. The three-condition ordering is stable, while the reduction depends on the cost calibration. The cell errors in the table describe the fitted cost function's predictive accuracy. Routing budgets and least-cost choices use this fitted cost function. MuSiQue uses a separately calibrated, piecewise-linear cost function.

E.3 Measuring Required Work in Controlled Routing

We estimate the empirical required work in Eq. (21) from utility across generation checkpoints for three routing conditions, independently of the cost calibration in Appendix E.2. Compact versus neutral-rich varies footprint without adding route information. Neutral-rich versus useful-rich holds length fixed and changes access to a certified shortcut. Their combination supplies the two measurements used in Figure 1.

Controlled routing uses the same RTX 4090, immutable Qwen2.5-1.5B-Instruct artifact, PyTorch SDPA, bfloat16 arithmetic, and fixed-slot batch size four used for runtime calibration. The frozen generator creates 128 routing tasks in eight disjoint blocks of 16. Each task instance contains a route of six edges and a shortcut record connecting its start directly to its destination. The six ordinary route edges are displayed in path order, with no route-edge distractor. The compact prompt contains the query and ordinary route records at an exact active length of 2,048 tokens. Neutral-rich retains those records and adds semantically inert archive material to reach 8,192 tokens. Useful-rich has the same 8,192-token length and reveals the certified shortcut. Deleting the added archive material and any supplied shortcut record recovers exactly the query and route records in the compact prompt.

Each query asks which destination is reached from the specified start and requires a path supporting that answer. For example, the first held-out case has the six-edge route from `QAS` to `UPM` below. Its complete ordinary-route response is

```
PATH: QAS -> ZXT -> KKV -> LFN -> LJD -> MLJ -> UPM
FINAL: UPM
```

The useful-rich prompt additionally supplies `CERTIFIED SHORTCUT: QAS -> UPM`, for which the following certificate is also valid.

```
PATH: QAS -> UPM
FINAL: UPM
```

The parser requires both `PATH:` and `FINAL:` fields. Utility is one when the path starts at the queried node, ends at the correct destination named in `FINAL:`, and uses only links supplied in that condition. Otherwise, utility is zero. The shortcut is accepted only in useful-rich. Each task in each condition follows a fixed-slot schedule through 64 output slots, with outputs scored at $K \in \{16, 24, 32, 48, 64\}$. After a sequence emits the end-of-sequence token, its scored output is frozen while its occupied batch slot continues with padding through the scheduled checkpoint. Thus, K counts output slots without attributing post-termination padding to the response. The first output token comes from prefill, so checkpoint K requires $K-1$ cached decoding steps. The reported aggregate required work is the first checkpoint at which mean utility across all 128 tasks reaches $\tau = 0.75$. A checkpoint can therefore fail because the path and final answer have not yet been completed, even when the available records identify the route.

The frozen cost mapping assigns output checkpoint K the fitted cost at $L = K$ cached decoding steps, one additional step beyond the scored checkpoint. This profiling proxy determines the reported routing costs and selections, including the selector grid in Appendix E.4. The original cost mapping was fixed before held-out task generation. Three budgets are fixed at midpoints between adjacent reachable costs. At 1,101.6 ms, compact can access $K = 24$ and rich conditions $K = 16$. At 1,359.5 ms, compact can access $K = 32$ and rich conditions $K = 16$. At 2,204.3 ms, compact can access $K = 48$ and rich conditions $K = 32$. The two primary contrasts are useful-rich minus compact at the low budget and neutral-rich minus compact at the middle budget.

The aggregate required-work estimates at $\tau = 0.75$ are $K = 32$ for compact and neutral-rich and $K = 16$ for useful-rich. At the 1,101.6-ms budget, useful-rich exceeds compact utility by 0.789, with a positive contrast in all eight blocks. At the 1,359.5-ms budget, neutral-rich minus compact utility is -0.844, with a negative contrast in all eight blocks. The equal required-work estimates for compact and neutral-rich show that added footprint alone does not reduce required work in this task. The fixed-budget contrasts then show how the profiling proxy changes which checkpoint each condition can afford. With condition-specific certificate sets, routing illustrates the tradeoff between footprint and work rather than testing fixed-utility dominance.

A retrospective sensitivity check instead evaluates the same fitted function at $L = K - 1$. The useful-rich saving over compact changes from 12.1% to 15.0%, while the three-condition cost ordering, all nine aggregate grid choices, and the checkpoint choices at the three representative routing budgets remain unchanged. This check extrapolates the useful-rich cost to $L = 15$, below the profiling grid minimum of 16, and does not provide a direct timing measurement at $K - 1$.

E.4 Testing Selection across Budgets and Shortcut Strengths

We evaluate two decisions using separately measured costs and utilities: maximizing success at a fixed budget and minimizing cost at a fixed numerical success threshold. An independent cohort varies footprint and shortcut strength to test when the least-cost choice changes.

Selecting under a fixed budget. We use the selection forms of Eqs. (12) and (13) with condition-specific routing success rates. The physical cost grid induces a finite set of budget intervals over which the affordable generation checkpoints do not change. For each held-out routing block and interval, each candidate's best affordable success rate is estimated from the other seven blocks by maximizing mean utility over its affordable checkpoints. The candidate with the largest estimate is selected, with candidate ties resolved by a fixed footprint-first order. When estimated utilities tie within a candidate, the calibration record stores the smaller checkpoint. Held-out scoring takes the maximum utility over affordable checkpoints. On every evaluated trajectory in the frozen grid, utility is nondecreasing with K, so this score is also attained by executing the largest affordable checkpoint. A zero-work option has zero utility. If no positive checkpoint is affordable, the representation label follows the tie-break rule. The three representative budgets in Table 4 were selected from the independent runtime cohort and yield 24 held-out decisions.

Table 4: Representative points from the controlled held-out budget range. C, N, and U denote compact, neutral-rich, and useful-rich.

Budget (ms)	Affordable K (C/N/U)	Selected	Selector	Fixed C	Fixed N	Fixed U
1,101.6	24/16/16	Useful-rich	0.797	0.008	0.000	0.797
1,359.5	32/16/16	Compact	0.844	0.844	0.000	0.797
2,204.3	48/32/32	Useful-rich	0.906	0.844	0.828	0.906
Mean	–		0.849	0.565	0.276	0.833

Across the complete checkpoint surface, all eight calibration folds select compact below 1,030.5 ms, useful-rich from 1,030.5 to 1,172.6 ms, compact from 1,172.6 to 2,062.2 ms, and useful-rich above 2,062.2 ms within the frozen $K \leq 64$ grid. The selected representation changes when an additional checkpoint becomes affordable for one candidate. Between such thresholds, the affordable pairs and the selected representation remain unchanged.

All eight folds select useful-rich at 1,101.6 ms, compact at 1,359.5 ms, and useful-rich at 2,204.3 ms. Mean held-out utility across the three equally weighted budgets is 0.849 for the selector. The corresponding means for fixed compact, neutral-rich, and useful-rich choices are 0.565, 0.276, and 0.833. The selected candidate belongs to the held-out blockwise maximizing set in 20 of the 24 decisions. Its mean utility shortfall from the blockwise maximum (regret) is 0.016. The mean difference between the selector and fixed useful-rich, paired by block, is 0.016. The selected representation changes with the budget, while its average advantage over fixed useful-rich is small in this comparison.

Selecting at a fixed utility threshold. We compare the cost of reaching the same numerical success threshold under each condition's certificate check, using the cost-comparison form of Eq. (11). Using calibration tasks disjoint from the reported cohorts, the controlled-routing required-work estimates at $\tau = 0.75$ are $K = 32$, 32, and 16 for compact, neutral-rich, and useful-rich, respectively. Their fitted continuation cost proxies are 1,172.6, 2,062.2, and 1,030.5 ms. The rule therefore selects useful-rich.

Varying footprint and shortcut strength. To test how the least-cost choice changes with footprint and shortcut strength, we evaluate a selector grid on a separate held-out cohort of 128 routing tasks in eight blocks of 16. Every scenario presents a common 2,048-token compact representation and one rich alternative. Rich footprint takes values 8,192, 16,384, and 24,576, while its information supplies no shortcut, a partial shortcut that jumps to a supplied five-edge route suffix, or a direct shortcut. The compact route has 18 edges. Prompts at each rich footprint are exactly length matched, so footprint and shortcut content vary independently. The generation checkpoint grid is $K \in \{16, 24, 32, 48, 64\}$ and the target is $\tau = 0.75$ under the valid-route criterion. Responses contain a single path whose final node supplies the answer, with no separate `FINAL:` field. The parser permits an omitted `PATH:` prefix and scores the path through its first occurrence of the destination. Later tokens do not undo attainment. The path must start at the query code and use only links supplied in that condition. Tasks and prompts are disjoint from development and the other controlled cohorts.

For each held-out block and scenario, the other seven blocks estimate required work for the compact and rich representations. The independently fitted cost function converts each finite work estimate to an estimated physical cost, and the smaller cost determines the representation choice. Exact cost ties favor the smaller footprint. A held-out oracle identifies the least-cost candidate reaching the target, using held-out utility and the same fitted cost function. It is computed only after the selected representation and checkpoint are fixed. The always-compact and within-scenario always-rich policies use their own seven-block required-work estimates, preserving the same information available to the selector. The nine binary decisions compare compact with the rich state available in that scenario. The no-shortcut column is therefore the neutral-rich comparison, while the partial and direct columns represent two strengths of useful-rich information.

The seven-block required-work estimates are stable across folds, with $K = 64$ for compact and no-shortcut rich, $K = 24$ for partial shortcut, and $K = 16$ for direct shortcut. Their estimated costs yield the selection pattern in Table 5. Each of the nine scenarios produces one selection decision for each of eight held-out blocks, giving 72 decisions. The primary agreement indicator is one only if the selected checkpoint reaches the target and its representation has the lowest fitted cost among

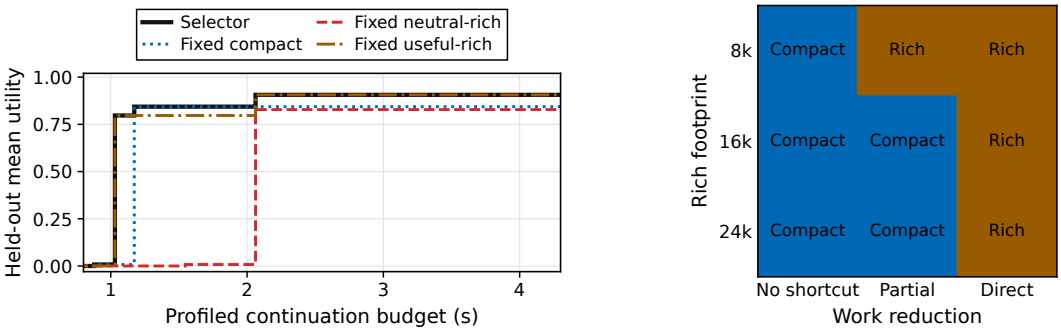
Table 5: Cross-fitted selector choice in the grid of footprints and shortcut strengths. Each entry is stable across the eight held-out folds.

Rich footprint	No shortcut	Partial shortcut	Direct shortcut
8,192 tokens	Compact	Rich	Rich
16,384 tokens	Compact	Compact	Rich
24,576 tokens	Compact	Compact	Rich

those reaching the target on that held-out block. Its mean is 0.986 (71/72). Always-compact and within-scenario always-rich achieve agreement 0.569 and 0.431. The paired agreement advantages are 0.417 over always-compact and 0.556 over within-scenario always-rich. Target attainment is 0.986 for the selector, 1.000 for always-compact, and 0.986 for within-scenario always-rich. The distinction from agreement reflects the target-cost objective, which prefers the least expensive feasible representation rather than the representation with the highest mean utility above threshold.

Among target-attaining decisions, the block-averaged mean excess cost over the held-out oracle is 0.0 ms for the selector, 311.7 ms for always-compact, and 1,526.3 ms for within-scenario always-rich. The single selector miss occurs in one held-out block of the 8,192-token direct-shortcut cell. Calibration selects rich at $K = 16$, but its held-out utility of 0.688 falls below the target, while compact reaches the target at $K = 64$. Thus, the grid supports the predicted change in the least-cost choice as work savings and footprint vary. The single miss also shows that a calibrated required-work estimate does not guarantee target attainment on every held-out block.

Figure 3a summarizes the budget-conditioned utility envelope, and Figure 3b shows the fixed-target choices across footprint and work reduction.



(a) Utility envelope across physical budgets

(b) Fixed-target representation choice

Figure 3: Controlled selection across operating regimes.

Exploratory sensitivity of the selection boundary. Keeping the observed required-work estimates fixed, we replace fitted costs with the eight-session cell means at $L = 16$ and $L = 64$, and linearly interpolate between $L = 16$ and $L = 32$ for the unprofiled $L = 24$ cost. Eight of the nine scenario choices in Table 5 remain unchanged. The exception is the partial shortcut at 16,384 tokens: the fitted comparison favors compact (2,346.4 versus 2,435.8 ms), whereas cell interpolation favors rich (2,928.3 versus 2,374.0 ms). This exploratory archive analysis locates a boundary sensitive to cost calibration. The $L = 24$ cost is interpolated rather than directly measured. The 71/72 agreement reported above uses the original fitted cost function for both the selector and its oracle. No new policy was evaluated on held-out tasks.

E.5 MuSiQue Evidence Selection and Supporting Results

This appendix evaluates selection before candidate processing using Eq. (13). A shared development utility table estimates the mean utility of each pair of scope and checkpoint, while each request's prompt length determines its affordable pairs. The primary 576-question experiment tests this adaptation at prespecified scope-change budgets. A separate 256-question experiment compares the selector with fixed scopes over a broader budget range.

Candidate construction and utility estimation. The selector's utility lookup uses 256 questions from the MuSiQue answerable development split (Trivedi et al., 2022), arranged in eight 32-question blocks balanced across two-, three-, and four-hop questions. A fixed BM25 ranker, using the question and decomposition chain as lexical queries, orders the supplied paragraphs without support labels or model outcomes. Every candidate prompt contains the same verified intermediate subquestions and answers from the dataset annotations, excluding the final answer, together with the top 2, top 4, top 8, or all ranked paragraphs. On development questions, these four scopes contain every annotated support for 17.6%, 54.3%, 82.8%, and 100% of tasks, respectively. These coverage calculations use support labels. Ranking and selection do not.

The primary model is Qwen2.5-7B-Instruct with SDPA, bfloat16, and batch size one. Its immutable snapshot identifier is `a09a35458c702b33eeacc393d103063234e8bc28`. For each question and candidate prompt, a single greedy generation run with 32 scheduled output slots supplies the response prefixes evaluated at $K \in \{2, 4, 6, 8, 16, 32\}$. Predictions and accepted reference-answer variants (gold aliases) are lowercased, with punctuation mapped to spaces and articles removed. The scored answer is the `ANSWER:` field, or the first nonempty line when that field is absent or empty. Exact match records normalized equality to any alias. Answer F1 is the maximum token-overlap F1 over aliases. Table 6 reports the development endpoints at $K = 32$.

Table 6: MuSiQue development performance at $K = 32$ from the fixed utility table.

Evidence scope	Mean prompt tokens	Answer F1	Exact match
Top 2	416.4	0.434	0.320
Top 4	685.1	0.567	0.441
Top 8	1,212.0	0.614	0.508
All paragraphs	2,779.4	0.648	0.523

Calibrating prompt-processing and continuation cost. A separate cost calibration uses three valid profiling sessions at eight prompt lengths and the same generation checkpoints. It measures the complete scheduled call, including prompt processing and continuation, at batch size one. At generation checkpoint K, prefill produces the first output token and is followed by $K - 1$ cached decoding steps. Padding steps preserve the schedule after end-of-sequence. Piecewise-linear interpolation in prompt length gives $\widehat{C}_x(j, K)$ for each candidate's actual token count. Work is evaluated only at measured checkpoints. At two measured cells, raw timing fluctuations violate the nondecreasing cost order. Enforcing this order changes any measured cost by at most 0.91%. The development utility table and this independent cost function determine the frozen selection rule before the held-out cohort is generated. The shared decomposition chain and paragraph ranking precede this accounting boundary. Their construction is excluded.

Evaluation at prespecified scope-change budgets. This evaluation tests request-specific selection of scope and checkpoint at budgets where the development-selected evidence scope changes.

The primary natural-question cohort contains 576 held-out questions from the MuSiQue answerable training corpus. Sixteen blocks of 36 questions each contain three tasks from every combination of hop count (two, three, or four) and all-paragraph footprint quartile. Task identifiers are disjoint from the development and other evaluation cohorts. MuSiQue questions can share underlying single-hop subquestions. Connecting questions that share such a subquestion defines 410 dependence clusters, with a largest cluster of 18 questions. These clusters are fixed before analysis.

Natural paragraph length produces a substantial footprint range while evidence coverage increases with scope. The top-2, top-4, top-8, and all-paragraph representations average 415, 694, 1,236, and 2,752 prompt tokens, and contain all annotated supports in 17.9%, 49.7%, 77.6%, and 100% of questions. The joint selector does not use these support labels.

The three evaluation budgets are selected from the development cohort before generating responses on the held-out questions. For each integer budget, the budget-level comparator assigns one evidence scope to all requests and uses the largest affordable checkpoint for each request. The comparator's scope changes at 210, 372, and 700 ms, where the selected scope changes to top 4, top 8, and all paragraphs, respectively, and at least half of development requests can reach checkpoint $K = 4$ under the new scope. The joint selector uses the same budget, utility table, and cost function, but may

change the scope according to the request's actual token footprint. A policy abstains with zero utility if no checkpoint is affordable.

Table 7: Answer F1 at prespecified scope-change budgets. The oracle maximizes held-out F1 over all pairs of scope and checkpoint affordable under the same profiled cost function and serves only as an upper reference.

Budget (ms)	Selector	Budget-level	Top 2	Top 4	Top 8	All	Oracle
210	0.387	0.330	0.347	0.330	0.158	0.000	0.482
372	0.510	0.486	0.392	0.496	0.486	0.022	0.652
700	0.562	0.524	0.394	0.520	0.570	0.524	0.719
Equal average	0.486	0.446	0.378	0.449	0.404	0.182	0.618

Table 7 reports the full fixed-policy comparison. The prespecified endpoint averages the paired F1 difference between the selector and the budget-level comparator over the three budgets for each question, then equally over the twelve balanced sampling cells. The resulting mean difference is $+0.0401$ F1. Exact match and attainment of F1 at least 0.65 follow the same direction. Their equal-budget averages are 0.344 and 0.455 for the selector, compared with 0.292 and 0.418 for the budget-level comparator.

Retrospective check of checkpoint comparability. The joint selector maximizes estimated utility over affordable checkpoints, whereas the frozen comparator takes the largest affordable checkpoint. To determine whether this distinction explains the primary gain, we reanalyze the archived cohort while holding the comparator's budget-level scope fixed and selecting its checkpoint with the same development utility table and cost function as the joint selector. Ties favor lower estimated cost, as in the joint rule. The optimized comparator chooses exactly the original checkpoint on all 1,728 decisions across questions and budgets, leaving the $+0.0401$ contrast unchanged. All 1,162 decisions where the joint and original comparator select the same scope also have the same checkpoint and F1. This retrospective check was added after the original experiment. For this cohort, taking the largest affordable checkpoint and optimizing it with development utility therefore give the same comparator.

Table 8: Evidence-scope differences from the budget-level comparator and their effects on footprint and work. Differences are averaged over requests assigned different scopes by the two policies.

Budget (ms)	Main change	Different scope	Δ prompt tokens	ΔK	F1 difference
210	Top 4 $\rightarrow$ Top 2	301 (52.3%)	-331	$+2.00$	$+0.110$
372	Top 8 $\rightarrow$ Top 4	72 (12.5%)	-657	$+3.89$	$+0.198$
700	All $\rightarrow$ Top 8	193 (33.5%)	$-1,886$	$+4.51$	$+0.114$

Why selection changes evidence scope. Table 8 shows that the main scope changes reduce prompt length and increase the affordable checkpoint, with positive mean F1 differences on the affected requests. The joint selector's counts by scope and checkpoint are top 2 at $K = 2$ (2 questions), top 2 at $K = 4$ (299), top 4 at $K = 2$ (14), and top 4 at $K = 4$ (261) for 210 ms. At 372 ms, they are top 4 at $K = 4$ (9), top 4 at $K = 6$ (63), top 8 at $K = 4$ (315), and top 8 at $K = 6$ (189). At 700 ms, they are top 8 at $K = 8$ (193), all paragraphs at $K = 6$ (257), and all paragraphs at $K = 8$ (126). Averaged across budgets, the scope disagreement rate increases through the four footprint quartiles as 7.6%, 17.8%, 37.0%, and 68.5%. The corresponding F1 differences are $+0.012$, $+0.016$, $+0.032$, and $+0.100$.

The only negative cell across budgets and footprint quartiles is the third quartile at 372 ms, where the mean difference is -0.0118. The two policies coincide on 131 of 144 questions. On the remaining 13, the selector uses top 4 at $K = 6$ instead of top 8 at $K = 2$. The mean difference within these changed requests is -0.131. The two changed two-hop questions improve, while the loss is concentrated among the five changed four-hop questions. This pattern shows variation in answer F1 across requests assigned the same evidence scope under the shared utility table.

Cost validation and nearby budgets. All 16 blocks satisfy the exclusive-GPU, monitoring, exact-checkpoint, and data-integrity criteria. Across 13,824 retained timing measurements, the profiled and raw total GPU times have correlation 0.9990, mean absolute percentage error 2.63%, and mean error

1.79 ms. All prompt lengths lie within the measured range used for interpolation. The profiled GPU cost defines affordability. Individual execution times provide the calibration check and may exceed the nominal budget.

To test sensitivity to the precise evaluation budgets, we analyze 27 policies frozen before target outcomes, covering $\pm 10\%$ around each scope-change budget. The selector difference is positive at 23 of the 27 budgets. All nine point estimates around each of the 210 and 372 ms scope-change budgets are positive. Four small negative estimates occur around the 700 ms scope-change budget, locating the budget region where the advantage reverses.

Evaluation over a broader budget range. To complement the comparison at scope-change budgets, this cohort evaluates the frozen selector against fixed evidence scopes over a broader budget range.

A separate held-out cohort of 256 questions comes from the answerable training split. It is disjoint from development in both task and decomposition-step identifiers, and from the scope-change cohort in task identifiers. Each of eight valid blocks contains 32 questions balanced across hop counts and four complete scope trajectories per question. The retrieval ranking, utility table, five budgets, cost function, and choices of the selector and fixed policies were recorded before answer generation. On these questions, the top-2, top-4, top-8, and all-paragraph scopes contain every annotated support for 19.5%, 53.1%, 78.1%, and 100% of tasks. The labels do not enter ranking or selection. Table 9 reports the corresponding endpoint performance.

Table 9: Held-out candidate performance at $K = 32$.

Evidence scope	Mean prompt tokens	Answer F1	Exact match
Top 2	422.0	0.404	0.309
Top 4	715.5	0.556	0.430
Top 8	1,238.8	0.576	0.453
All paragraphs	2,757.9	0.619	0.477

Table 10 records the selected pairs of scope and checkpoint, and Table 11 compares their answer F1 with fixed scopes across budgets.

Table 10: Frozen choices of scope and checkpoint for 256 held-out questions. An entry $n@K$ means that n questions use the indicated evidence scope and checkpoint. Mean K excludes abstentions.

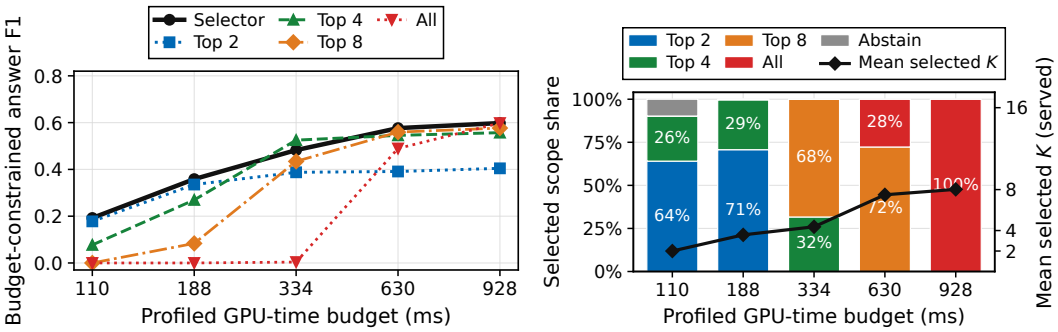
Budget (ms)	Abstain	Top 2	Top 4	Top 8	All	Mean K
109.9	25	164@2	67@2	—	—	2.00
188.0	0	7@2, 174@4	46@2, 28@4	1@2	—	3.58
334.0	0	—	48@4, 33@6	161@4, 14@6	—	4.37
630.1	0	—	—	3@6, 182@8	62@6, 9@8	7.49
927.6	0	—	—	—	1@6, 254@8, 1@16	8.02

Table 11: Held-out answer F1 across prespecified budgets in profiled GPU time. The final column compares the selector with fixed top 4. Differences use unrounded scores.

Budget (ms)	Selector	Top 2	Top 4	Top 8	All	Sel. − Top 4
109.9	0.192	0.178	0.077	0.000	0.000	+0.115
188.0	0.359	0.335	0.269	0.084	0.000	+0.089
334.0	0.483	0.388	0.525	0.434	0.004	−0.043
630.1	0.577	0.391	0.545	0.559	0.490	+0.031
927.6	0.598	0.405	0.557	0.577	0.598	+0.041
Budget average	0.442	0.339	0.395	0.331	0.218	+0.047

The budget-average selector gain over fixed top 4 is 0.0468. The eight paired block differences are $0.0679, 0.0232, 0.0751, 0.0559, -0.0167, 0.0506, 0.0804,$ and 0.0377. Exact-match accuracy, computed by normalized equality with any gold alias, averages 0.288 for the selector and 0.264 for fixed top 4 across budgets. At the lowest operating point, 90.2% of selector requests are affordable,

compared with 26.2% under fixed top 4. All policies serve every request at the largest budget. Budgets denote affordability on the independently profiled cost function, while raw single-run timings are retained as a separate measurement. Figure 4 shows the answer-F1 curve and the shift from narrower scopes with earlier checkpoints toward wider scopes with later checkpoints as the budget grows. At 334 ms, the selector scores 0.483 versus 0.525 for fixed top 4. At the largest budget, it chooses all paragraphs and matches that policy's 0.598 F1. These results support budget-dependent allocation of evidence scope and generation checkpoint while showing where the shared utility estimates favor a suboptimal scope.



(a) Budget-constrained answer F1

(b) Selected evidence scope and checkpoint

Figure 4: Evidence-scope and checkpoint selection in the 256-question broad-budget cohort. Panel (a) assigns zero utility to unserved requests.

E.6 Exact KV-Cache Rebuilding and Transition Costs

This experiment tests the post-acquisition selection rule in two ways. At matched continuation checkpoints, it asks whether decoding savings offset the transition cost of replacing an active representation. Under a fixed budget, it then asks whether paying that cost makes a later checkpoint affordable. The rebuilding procedure targets exact restoration of the compact representation, corresponding to the $\varepsilon = 0$ case of Eq. (10) under execution compatibility.

Active-cache state and rebuilding procedure. The experiment uses Qwen2.5-1.5B-Instruct with an RTX 4090, SDPA, bfloat16, and batch size four. The cache-rebuilding cohort contains 128 additional routes in eight disjoint 16-case blocks. Each route has 32 ordered edges and no distractors. The canonical one-line response that lists the complete path requires at most 107 tokens under the fixed tokenizer. The parser requires a `PATH:` field but no `FINAL:` field. Utility is one only when the entire parsed path starts at the queried node, ends at the correct destination, and uses only supplied links. After the 8,192-token prompt has been processed, a deterministic selector retains the query and the route records needed to answer it. The rich KV cache is discarded, and the selected request is processed as the exact 2,048-token compact sequence. Token equality checks restoration of the compact reference prompt for every evaluated task. The transition cost d includes structured selection, tokenization, cache reset, and reprocessing the compact request. Both choices start with the rich KV cache already active, so its initial acquisition cost precedes the decision and is excluded from the comparison.

Persistent-rich, rebuilt-compact, and compact conditions use the same cases and are greedily decoded once to 128 output slots, with prefixes scored at $K \in \{96, 112, 128\}$. Here K counts output slots: the first token uses the prefill logits, so the timed continuation executes $K - 1$ cached forward passes. At $K = 112$, each block yields a latency saving, computed as persistent-rich minus rebuilt-compact latency proxy, and a utility gain, computed as rebuilt-compact minus persistent-rich utility. The proxy adds wall-clock transition time to GPU continuation time measured with CUDA events for rebuilt-compact. Persistent-rich has no transition cost. All eight blocks satisfy the validity criteria.

Offsetting the transition cost at matched work. Table 12 shows that the net saving grows with the continuation checkpoint. All 128 rebuilt prompts and complete outputs match compact exactly. At $K = 112$, mean utility is 0.648 for persistent-rich and 0.875 for rebuilt-compact, a gain of 0.227. The mean latency proxy is 5,931.1 ms for persistent-rich and 5,795.1 ms after rebuilding, a saving of 136.0 ms. Post-rebuild continuation saves 832.5 ms relative to persistent-rich, while the mean

Table 12: Exact KV-cache rebuilding across continuation checkpoints.

	Mean utility		Latency proxy (ms)		
Checkpoint K	Persistent	Rebuilt	Persistent	Rebuilt	Saving
96	0.008	0.016	5,073.1	5,062.1	11.0
112	0.648	0.875	5,931.1	5,795.1	136.0
128	0.648	0.875	6,791.2	6,528.8	262.4

transition cost is 696.5 ms. The net saving increases from 11.0 ms at $K = 96$ to 262.4 ms at $K = 128$. The smaller retained footprint reduces each subsequent step's cost, so longer continuations more readily offset the rebuilding cost.

Selecting after acquisition under a fixed budget. A retrospective fixed-budget replay applies Eq. (12) to the post-acquisition candidate set. At 5,825 ms, development calibration predicts $K = 96$ when the rich cache is retained and $K = 112$ after paying the measured transition cost. The rule therefore selects rebuilt-compact. Across the eight held-out blocks, persistent-rich attains mean utility 0.008 at the largest affordable checkpoint, whereas rebuilt-compact attains 0.625. Six rebuilt blocks reach $K = 112$ and two reach $K = 96$ because the measured transition cost is included separately for each block. Affordability in this replay is determined from each block's mean latency proxy, including its measured transition cost. It is not enforced by an online deadline controller. By making a later checkpoint affordable, the post-acquisition rule improves utility in this replay while accounting for the full measured transition cost.

E.7 Compressed Candidates and Their Acquisition Costs

Compression reduces prompt length but incurs a cost to construct the compressed candidate. To test how this cost affects selection, we add LLMLingua-2 compressed candidates (Pan et al., 2024) to the four raw MuSiQue scopes and compare online with precomputed compression, accounting for acquisition through $d_x(j)$ in Eq. (11). Using its multilingual BERT-base MeetingBank model (`llmlingua` 0.2.2), we compress the top-8 and all-paragraph evidence, targeting retention of 50% or 25% of the evidence tokens. The question and verified non-final decomposition chain remain verbatim. Each representation is crossed with $K \in \{2, 4, 6, 8, 16, 32\}$ at the five budgets in Table 11. A separate 256-question development cohort supplies the utility table, and the joint selector ranks raw and compressed candidates using their measured costs. All candidate representations, compression outputs, acquisition costs, budgets, and policy choices are fixed before generating responses on 256 held-out questions.

Two accounting regimes distinguish constructing a compressed candidate from using a precomputed compression output. The online regime adds the measured compressor GPU time to the target model's prompt-processing and continuation cost. The precomputed regime omits only that compression cost. Target-model prompt processing and continuation remain charged in both regimes. Online accounting charges compression only for the selected representation. Constructing unselected candidates for this frozen comparison is excluded.

The budget-level comparator fixes one of all eight representations per budget and accounting regime using development F1. Footprint-only selects the shortest affordable prompt. Widest affordable raw selects the longest affordable raw prompt. Each uses its largest affordable checkpoint.

Table 13 reports the expanded-set policies. With online accounting, the joint selector uses a compressed representation for 23 of 1,280 decisions across requests and budgets and differs from the budget-level comparator by -0.001 F1. When the compression outputs are precomputed, compressed candidates account for 256 decisions and the difference is $+0.004$. These differences use unrounded F1 and are small in both accounting regimes. The change in selection frequency shows how compression costs included in $d_x(j)$ affect which compressed candidates are affordable. In both accounting regimes, the joint selector achieves higher mean F1 than the footprint-only policy, supporting selection that considers utility as well as cost.

Table 13: Budget-average answer F1 after adding LLMLingua-2 candidates. Online accounting includes compressor GPU time. Precomputed accounting omits this cost. Both include target-model prompt processing and continuation.

	Budget-average answer F1	
Policy or fixed representation	Online	Precomputed
Joint selector	0.391	0.389
Budget-level comparator	0.393	0.386
Footprint-only policy	0.343	0.353
Widest affordable raw scope	0.345	0.345
Raw Top 2	0.349	0.349
Raw Top 4	0.327	0.327
Raw Top 8	0.288	0.288
Raw all paragraphs	0.184	0.184
LLMLingua-2 Top 8 / 50%	0.319	0.341
LLMLingua-2 Top 8 / 25%	0.251	0.307
LLMLingua-2 all / 50%	0.221	0.269
LLMLingua-2 all / 25%	0.212	0.264

E.8 Evidence and Decomposition-Chain Effects across Models

This experiment compares context effects across two model sizes on 256 development questions. The supports-only condition provides the paragraphs annotated as necessary for answering the question, while the all-retrieved condition adds the remaining candidate paragraphs. This comparison changes both footprint and distractor content, and the annotated supports serve as a diagnostic reference. A separate comparison retains all paragraphs and replaces a neutral note with the verified intermediate subquestions and answers, matching prompt length to isolate the effect of the chain. Table 14 reports these comparisons for both models. They test the ingredients of evidence selection, not held-out selector transfer between models.

Table 14: Development-set answer F1 by hop count and model at generation checkpoint $K = 32$.

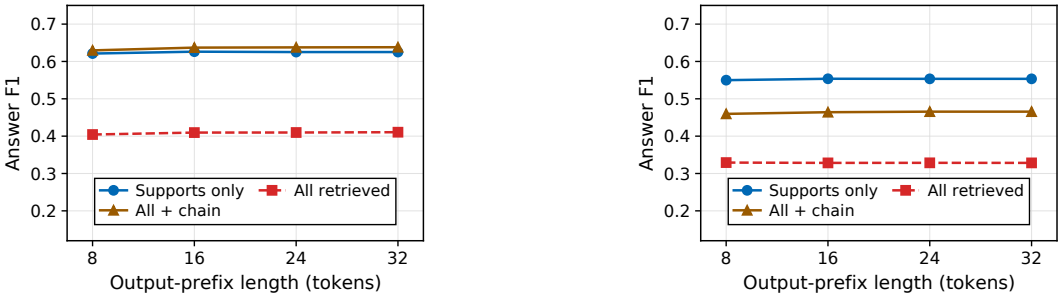
	Qwen2.5-7B			Qwen2.5-1.5B		
Hops	Supports	All	All + chain	Supports	All	All + chain
2 ($n = 85$)	0.654	0.458	0.722	0.598	0.303	0.494
3 ($n = 86$)	0.582	0.404	0.562	0.553	0.345	0.469
4 ($n = 85$)	0.640	0.369	0.631	0.510	0.337	0.434

Qwen2.5-1.5B-Instruct provides a separate-model check of the two directional context contrasts. Its model snapshot identifier is `989aa7980e4cf806f80c7fef2b1adb7bc71aa306`. It uses the same questions, prompts, block allocation, validity rules, and analysis. At $K = 32$, answer F1 for supports only, all retrieved, and all + chain is 0.553, 0.328, and 0.466. Exact match is 0.430, 0.230, and 0.340. Using only annotated supports instead of all paragraphs increases F1 by 0.225. Adding the intermediate chain at matched prompt length increases F1 by 0.137. Both differences are positive in every block. Mean GPU latency is 1,471.5, 1,621.3, and 1,605.7 ms. Retaining all paragraphs rather than only annotated supports increases latency by 149.7 ms, also positive in every block. Both models show higher latency when all paragraphs are retained and higher answer F1 when the intermediate chain is supplied at matched prompt length. The F1 increase is larger for the 7B model.

Relationship between evaluation cohorts. Evaluation cohorts have distinct task identifiers but can share underlying single-hop subquestions. Their results are reported separately rather than pooled as independent replications.

Answer quality across output prefixes. To examine when answer quality emerges, Figures 5a and 5b score prefixes of the same greedily generated 32-token output. GPU latency is reported for the complete run. Most gold answers are short spans, so the curves characterize early answer quality within the short-answer interface. On both models, the condition ordering is already visible

at the 8-token prefix and changes little through the 32-token prefix. These scores describe answer availability, not internal reasoning work or separately measured prefix costs.



(a) Qwen2.5-7B (b) Qwen2.5-1.5B

Figure 5: MuSiQue answer F1 by output-prefix length for both model sizes.

A worked MuSiQue example. Consider the two-hop question, "What movie did the person who acted in the comedy film Crime School win his only Oscar for?" Supports only contains the two annotated paragraphs, Crime School and Humphrey Bogart. All retrieved adds 18 distractors and a note that no intermediate link was supplied. Neutral tokens fill the remaining note allocation. All + chain keeps the same 20 paragraphs and supplies the verified non-final decomposition step "Who acted in Crime School? ⇒ Humphrey Bogart," followed by the unresolved lookup, without stating the gold answer, The African Queen. Both all-retrieved prompts contain 2,510 tokens, versus 220 for supports only. At $K = 32$, both models answer The African Queen from supports only and all + chain. From all retrieved, 7B gives The Omen and 1.5B Jerry Goldsmith.