TAILALLOC: ALLOCATING THE KV CACHE BY TAIL-RISK CONTRIBUTION, NOT STANDALONE RISK

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-context inference caches key–value (KV) pairs for every token read, and at long contexts that cache dominates serving cost. Quantisation and sparse attention shrink it without choosing how many tokens each attention head keeps; KV eviction does, under a fixed budget, so a scoring rule picks which tokens a head discards and an *allocator* sets how much budget it gets. Every published allocator minimises a sum of per-head risks, and that sum is the wrong quantity: it is the model's tail risk only when heads are comonotonic, failing on the same requests, and otherwise overstates it by a median 55% over six captures on three model families. What governs the tail is each head's Euler risk contribution, its average error on the requests where the model's total error is worst; by it, 6–38% of heads are *tail hedges* whose error moves against the model's, and rescored on the tail risk itself no published allocation beats a uniform budget. We therefore propose `TailAlloc`, which estimates these Euler risk contributions on a calibration set and allocates against them with a marginal-gain greedy algorithm. This replaces direct tail averaging with a covariance-based closed-form estimator where tail samples are scarce. Against the exactly solved optimum of the separable objective — the strongest published allocator, its dual gap certified — `TailAlloc` wins 124 of 140 rows under leave-one-capture-out, by a median 1.96 percentage points. Task metrics move too: a contribution-built allocation gains $+2.59$ [+1.78, +3.40] over a matched uniform budget on RULER at 4096, and on LongBench it clears an equal split on two captures of six, the separable optimum on none. We further note that when design-space effects fall below an evaluation's resolution, average benchmark scores do not provide evidence for the corresponding design choice.

1 INTRODUCTION

A language model serving a long context keeps a key–value (KV) cache for every token it has read. The cache grows with the context, and at long contexts it is a first-order cost of serving. Eviction methods keep it small by discarding the tokens a scoring rule judges unimportant. A second decision sits on top of the scoring rule: given a total budget, how much of it should each attention head and each layer receive?

Published answers fall into three groups. The first divides the budget equally. The second divides it unequally along one axis: Ada-KV (Feng et al., 2025) across the heads of a layer, CAKE and layer-DefensiveKV across layers. The third, LU-KV (Tang et al., 2026), convexifies each head's cost curve and solves the allocation exactly with a marginal-gain greedy. Different as they are, all of them minimise the same shape of objective: a sum over heads of that head's own risk. Underneath that sum is an assumption none of them states, because none of them needs to: that the sum of per-head risks *is* the risk of the model.

That assumption holds in exactly one case. For a tail-risk measure such as conditional value-at-risk, the per-head sum equals the model's aggregate risk only when every head fails on the same queries; otherwise it is the largest value the aggregate could take under *any* dependence between heads. Figure 1 shows how far a real model sits from that case. The quantity that governs each head's share of the aggregate tail varies forty-fold across heads and runs down through zero, and 33 of 288 heads are *tail hedges*: their error moves against the model's, so the standalone objective wants to protect them and the aggregate tail does not. **Every KV-cache allocator gives budget to the heads with the**

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.

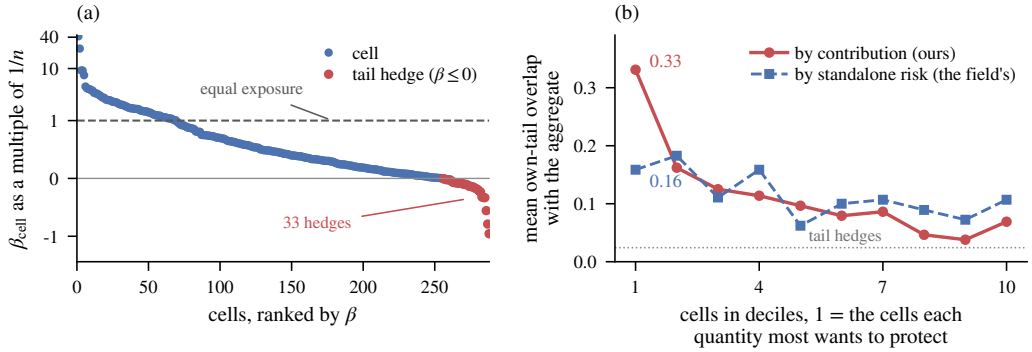

Figure 1: **The quantity every allocator ranks by is not the quantity that governs the tail. (a)** One dot per cell, ordered by β_i, in multiples of $1/n$ — its value if all n cells were equally exposed. By Proposition 1 a dot at or below zero is a cell where the two objectives disagree most. β spans $40\times$ that value down through zero, and 33 of 288 cells are *tail hedges* whose error moves against the aggregate. **(b)** Cells in deciles by each quantity, decile 1 the ones it most wants to protect, at their mean *own-tail overlap*: the share of a cell's own worst 1% of queries that are the aggregate's worst 1%, which (3)'s comonotone case would make 1. Ranked by contribution the profile falls from 0.33 to 0.07. Ranked by standalone risk it does not fall: it wanders across 0.06–0.18 with no trend, and its highest decile is its second, not its first (Spearman $+0.445$ against $+0.139$) — the quantity the field ranks by orders cells by how bad they are, not by whether their badness is the model's. `s0001`, SnapKV, per-cell normaliser, $\alpha = 0.99$; all six captures in Table 5.

worst errors. It should give budget to the heads that are failing at the same time as the model — a measurably different quantity, wrong outright in 6–38% of heads, and worth 1.96 percentage points of tail reduction to fix.

The quantity that does govern the aggregate tail is each head's *Euler risk contribution*: its average error on the queries where the model's total error is worst. The definition is standard in risk theory; what is new is handing it to an allocator as the curve to allocate on. We build that allocator, `TailAlloc`, and three things make it hard. First, the contribution conditions on the aggregate's own tail, which at the 99th percentile holds five of 512 calibration queries, so we replace the tail average by a covariance-based closed form that reads every query. Second, which queries are in the tail depends on the allocation, so estimation and the marginal-gain greedy alternate until the allocation stops moving. Third, a gain on the objective need not reach a task metric, so every task result is a matched pair with a prediction registered before the arm ran, and we ask not only whether the correction helps but whether the benchmark can tell.

Contributions. What is new is the measurement.

1. **The right quantity.** The quantity that governs a model's aggregate tail is each head's Euler risk contribution, not its standalone risk. The two differ per head by its spread times one minus its correlation with the aggregate, and summed over heads the gap is exactly the looseness of the separable bound.
2. **The measurement.** On six captures of three model families the separable objective overstates the tail it bounds by a median 55%, 6–38% of heads are tail hedges, and no row of 140 is free of them. Scored on the aggregate tail, CAKE, Ada-KV and layer-DefensiveKV come out negative in 17 of 18 cells. *That ordering does not transfer wholesale*: matched on RULER, Ada-KV's allocation costs 1.37 points against uniform but layer-DefensiveKV's *gains* 2.07.
3. **The allocator.** `TailAlloc` estimates the contribution from five tail samples with a closed form that beats the empirical tail mean, and one fixed recipe, chosen on five captures and applied unchanged to the sixth, beats the exactly-solved separable optimum by 1.96 pp. A contribution-built allocation reaches a task metric: $+2.59$ [+1.78, +3.40] over questions and [+0.72, +4.73] over task types on RULER at 4096, and on LongBench it beats an equal split on two captures of six where separable allocations manage none.

4. **What the benchmarks can resolve.** The choice between the two objectives is not small: over six captures on two benchmarks it spans -4.86 to $+5.65$ points, and pricing this paper's own claim takes 117755 questions on RULER against 742 on one LongBench capture. Most design choices in this space sit below what the benchmarks resolve, and we report which, so that benchmark means are read as evidence only where they are.

2 Related work

KV-cache compression. Eviction keeps a subset of the cache chosen by a scoring rule — recency and attention sinks in StreamingLLM (Xiao et al., 2024b), pooled attention in SnapKV (Li et al., 2024), the multi-state-RNN view of TOVA (Oren et al., 2024), key norms in K-Norm (Devoto et al., 2024), output perturbation in CriticalKV (Feng et al., 2026b), and the worst case over an observation window in DefensiveKV (Feng et al., 2026a): the six policies harvested here (§6.1). Quantisation (Liu et al., 2024; Hooper et al., 2024) and retrieval-style sparse attention (Tang et al., 2024; Xiao et al., 2024a; Jiang et al., 2024) shrink the cache without choosing a per-cell budget, so §4 does not bear on them; D2O (Wan et al., 2025) merges tokens but carries a layer-level allocation of exactly the shape below.

Budget allocation across heads and layers. We characterise allocators by what each optimises, not by the scoring rule each uses to fill a budget. Ada-KV (Feng et al., 2025) introduced head-wise allocation within a layer, minimising an upper bound on the attention-output eviction loss; CAKE (Qin et al., 2025) allocates across layers by a preference built from attention dispersion and temporal variance, and proposes head-level allocation as future work; DefensiveKV takes its worst case *per cell* and sums it, the right-hand side of (3); and LU-KV (Tang et al., 2026), the strongest separable baseline here, relaxes the head-budget problem onto its convex hull and solves it exactly by a marginal-gain greedy (Fox, 1966; Katoh et al., 2013), so our gains come from a different functional, not a better solver. LKV (Zhou et al., 2026), ReasonAlloc (Liu et al., 2026), HeteroCache (Shi et al., 2026), LAVa (Shen et al., 2025) and CONF-KV (Li & Miao, 2026) appeared during this work and are not characterised here. Every allocator we were able to read optimises $\sum_i f(X_i)$ for some per-cell functional f of that cell's own error, and only this paper's depends on the joint law (Table 9, Appendices G.24, H.1). The sum was adopted for tractability; §4's point is that the quantity made tractable is the wrong one.

Risk measures and attribution. The machinery of §4 is standard: coherence and the comonotone equality condition are Artzner et al. (1999)'s (see also McNeil et al., 2015; Föllmer & Schied, 2016), the optimisation form of CVaR is Rockafellar & Uryasev (2000)'s (see also Shapiro et al., 2021), and the Euler allocation's uniqueness is Tasche (1999)'s, with Denault (2001)'s game-theoretic reading and Bauer & Zanjani (2016)'s marginal-cost one. The obvious alternative to estimating a contribution, the worst case in a shrinking divergence ball, evaluates at a *higher* effective tail level, while our sweep over the fitting level lowers it in 56 of the 92 rows of the four captures it was run on and raises it in none (Appendix G.14).

3 Problem formulation

Cells, curves and budgets. A *cell* i is one (layer, KV-head) pair — what the abstract calls a head. For each cell we record the per-query eviction error, the difference between the attention output on the full cache and on the retained set, at a grid of cache levels $b \in \mathcal{B}$, giving a per-cell *cost curve* $e_i(b)$ and one realisation of it per query. Write $X_i = e_i(b_i)$ for the error random variable of cell i at budget b_i, and $Z = \sum_i X_i$ for the aggregate. Allocation means choosing one level per cell subject to a total, $\sum_i b_i = B$, with the scoring rule that fills each cell's budget held fixed.

The risk measure. $\mathrm{CVaR}_\alpha(Y)$ is the average of Y over its worst $(1-\alpha)$ fraction of outcomes (Rockafellar & Uryasev, 2000); under the mean the two objectives below coincide, because expectation is linear, so the question is real only for a non-additive risk measure, and the tail is what a serving system is judged on.

Two objectives. Published allocators differ in how they estimate per-cell hardness and in how they spend a budget against it, but they agree on what they are spending against: a *separable* objective,

$$J_{\text{sep}}(b) \;=\; \sum_i \rho(X_i)\,, \qquad \rho \in \{\mathbb{E}, \text{CVaR}_\alpha\}, \tag{1}$$

the sum over cells of each cell's own risk. Separability is what makes the problem tractable: convexify each cell's curve and walk a greedy over the pooled marginal gains, which is exactly optimal under Fox's condition (Fox, 1966; Federgruen & Groenevelt, 1986; Katoh et al., 2013) and is what Tang et al. (2026) implement. A serving system, however, is judged on the aggregate. The problem this paper takes as the target is

$$\min_b \;\; \text{CVaR}_\alpha\Big(\sum_i e_i(b_i)\Big) \quad \text{subject to} \quad \sum_i b_i = B, \;\; b_i \in \mathcal{B}, \tag{2}$$

the same constraint set as (1) with the risk taken *of the sum* rather than summed over cells. Section 4 asks when the two coincide, §5 how the target's per-cell quantity can be estimated, and §6 what each buys at a task benchmark.

4 Theory

The difficulty with (1) is that J_{sep} is not the risk of the aggregate in (2). For a coherent risk measure the two are related by subadditivity, and the direction of the inequality is not in doubt:

$$\text{CVaR}_\alpha\textstyle(\sum_i X_i) \;\le\; \displaystyle\sum_i \text{CVaR}_\alpha(X_i)\,, \tag{3}$$

with equality if and only if the X_i are *comonotone*: the query on which one cell has its worst error is the query on which every other cell has its worst too (Artzner et al., 1999; McNeil et al., 2015; Föllmer & Schied, 2016). The right-hand side is therefore the largest value the aggregate's risk could take under *any* dependence leaving each cell's own distribution unchanged, so every published allocator budgets as though all cells failed on the same queries — chosen not for robustness but because a separable problem is exactly solvable.

The Euler decomposition. CVaR_α is positively homogeneous — scaling every cell's error by c scales the aggregate risk by c — which is what makes "this cell's share of the total risk" well defined at all, and gives the Euler decomposition with no remainder:

$$\text{CVaR}_\alpha(Z) \;=\; \sum_i \underbrace{\mathbb{E}[\,X_i \mid Z > \text{VaR}_\alpha(Z)]}_{\text{RC}_i}, \tag{4}$$

where RC_i is the *risk contribution* of cell i: its average error on exactly the queries where the aggregate is failing. Unlike the standalone risks in (1) these sum to the aggregate exactly, and they are the *unique* attribution consistent with optimisation (Tasche, 1999). Allocators rank by $\rho(X_i)$, not by $\partial\,\text{CVaR}_\alpha(Z)/\partial b_i$, so descending it descends the aggregate tail only by accident.

The coincidence condition. The accident is testable. Under an elliptical joint law for (X_i, Z) — the family containing the Gaussian and the multivariate t, which §4.1 tests rather than assumes — the conditional expectation in (4) is linear in Z, with $\mu_i = \mathbb{E}[X_i]$ and $\mu_Z = \mathbb{E}[Z]$:

$$\text{RC}_i \;=\; \mu_i \;+\; \beta_i\,(\text{CVaR}_\alpha(Z) - \mu_Z)\,, \qquad \beta_i \;=\; \frac{\text{Cov}(X_i, Z)}{\text{Var}(Z)}. \tag{5}$$

Proposition 1 (The separating quantity)**.** *Fix a budget grid and a total. Under the elliptical law above, with* $\varrho_i = \text{Corr}(X_i, Z)$ *and* $\lambda_\alpha > 0$*, the two per-cell quantities differ by* $\text{CVaR}_\alpha(X_i) - \text{RC}_i = \lambda_\alpha \sigma_i (1 - \varrho_i) \ge 0$*, summing over cells to exactly the subadditivity gap of* (3)*. The rankings coincide when that gap is constant across cells — in particular when every* $\varrho_i = 1$*, the comonotone case of* (3)*. It is largest where* $\beta_i \le 0$*: a cell whose error moves* against *the aggregate. Constant* β_i *suffices only against a mean-based separable objective (Appendix G.6).*

It claims an ordering, not an optimality gap, the grid being discrete, and its value is that both sides of it are measurable. A cell with $\beta_i \le 0$ is a *tail hedge*. Allocation is decided by the last few tokens moved, and that is where the orders disagree.

4.1 DISTRIBUTION OF THE TAIL

Two questions no algebra settles: how loose is (3) in practice, and is (5) the right approximation? Over the 92 rows of the first four captures — not re-run on the two added later — the separable objective exceeds the aggregate tail risk it bounds by **tens of percent** at cell level. How large the overstatement is and how it is distributed is the companion measurement submission's own result and is not restated here. Aggregating within a layer first reduces it but does not remove it. One escape would remain: a tail carried by one cell at a time — the subexponential regime — would separate again over a common absolute threshold, and it does not (Appendix G.18): the tail is a *diffuse, many-cell* event, the regime in which (5) is a reasonable first approximation, and §5 treats it as one.

5 METHOD

Only one thing changes. The capture, the per-cell error curves, the convexification and the greedy are shared with the separable methods; we change which curve the allocator is handed, from a cell's own tail risk to its error on the queries where the total is failing (Figure 6, Appendix G.17).

Tail sample size. At $\alpha = 0.99$ on 512 calibration queries, (4) is an average over five points — and the five are not five independent observations. Exceedances arrive in episodes: over eight consecutive windows the median aggregate error moves by 0.3% while one window carries a $\mathrm{CVaR}_{0.95}$ twice its neighbours'. The classical summary is the extremal index $\theta \in (0, 1]$ (Leadbetter et al., 1983; Ferro & Segers, 2003; Berghaus & Bücher, 2018), which counts $\theta(1-\alpha)Q$ independent tail events in Q queries rather than $(1-\alpha)Q$. It is a property of the series, so we estimate it on the whole 2048-step decode and apply it to the calibration window: we measure $\theta = 0.234$–0.545 at $\alpha = 0.95$, where the estimate is solid, and 0.31–0.35 on the two Qwen captures at 0.99, where it is too noisy to quote across all four: those five exceedances are worth about *one and a half*. A covariance reads all 512 queries instead. That is the structural half of why the closed form of (5) wins; we found it by chasing a bootstrap bound that never covered (Appendix G.16). That the recipe then *fits* the contribution at the same α it optimises is a mistake we found too late to ship: fitting shallower than the scoring level is worth a paired median of $+0.25$ pp over five leave-one-capture-out folds, no arm in this paper runs it, and Appendix F.3 carries the fold-level evidence and the $\alpha = 0.95$ refusal.

5.1 THE ESTIMATOR FAMILY

Equation (4) defines RC_i as a conditional expectation. Conditioning on the value of the aggregate turns it into an average of a regression function over the aggregate's own tail, where F is the law of Z and $v = \mathrm{VaR}_\alpha(Z)$:

$$\mathrm{RC}_i \;=\; \mathbb{E}_{z \sim F(\cdot \mid Z > v)}[m_i(z)]\,, \qquad m_i(z) = \mathbb{E}[X_i \mid Z = z]\,. \tag{6}$$

Every estimator of RC_i is therefore a choice about m_i: how flexible it is, and on which part of the sample it is fitted. The two obvious choices are the endpoints of that space, and neither is a good default. *Endpoint one, the closed form.* Taking m_i linear and fitting it on every query gives (5) back. Its virtue is sample efficiency — a second moment reads all Q queries — and its vice is model bias, which is visible: against the empirical contribution on held-out data it attains Spearman 0.898 at $\alpha = 0.99$ and 0.980 at 0.95, with median relative errors of 0.26 and 0.11. It is a better estimator exactly where it is a worse model. *Endpoint two, the tail-conditional mean.* Averaging X_i over the queries in the aggregate's tail is unbiased and, at $\alpha = 0.99$, an average of five points. Bagging smooths the boundary the tail set induces (Breiman, 1996; Bühlmann & Yu, 2002; Efron & Hastie, 2016) but cannot manufacture information five points do not contain.

The intermediate family. Two knobs span the space: the degree of m_i (1 or 2) and a locality quantile $q \in \{0, 0.5, 0.8\}$ above which it is fitted, so raising q buys locality and spends sample. It also contains the bagged tail mean and a convex blend of the linear estimator with it, weighted by cross-validation *inside the calibration window*. The allocator is unchanged — convexify and walk the greedy of §6.1, and both halves earn their place (Appendix G.13) — but the contribution curve depends on which queries are in the tail and that depends on the allocation, so estimate and solve alternate: each inner solve is exact for its tail set, the outer loop is the sample-average form of the Rockafellar–Uryasev program (Rockafellar & Uryasev, 2000), and we keep the best iterate; the result is `TailAlloc` (Appendix H.2).

Cost and alternatives. Forming the contribution curves takes 2.9 ms in the median against 19.5 ms for the standalone CVaR curves the separable objective needs; the full alternating loop is 109 ms on a 512-query calibration, about 2.7% of one prefill of the same model on the same card, offline and off the serving path. The capture is the expensive half and it is elastic: calibrating on the first 512 queries and scoring on all 2048 keeps more than 90% of the reduction on both models (Appendices E.1, G.11).

6 EXPERIMENTS

6.1 EXPERIMENTAL SETUP

Measured setup. Every query head in each grouped-query group is evaluated: $n = 288$ cells on Qwen3-8B, $n = 256$ on Llama-3.1-8B and Mistral-7B. The cost curve $e_i(b)$ of §3 is recorded at four cache levels $b \in \{512, 2048, 4096, 8192\}$ over 2048 decode steps — one *query* per decode step, the unit every count here is in — giving 2048 realisations of it per cell. The operating total is 2048 per cell, which at a 32768-token context keeps 6.25% of the tokens.

Tail levels and normalisers. At $\alpha = 0.99$ over 512 queries, CVaR_α is the mean of the five largest. We measure damage at $\alpha \in \{0.95, 0.99\}$ and report everything under two normalisers: *absolute*, and *per-cell*, which divides each cell's error by its median attention-output norm before summing.

Captures, policies and windows. Six captures: four long public-domain prose corpora on Qwen3-8B at 32768 tokens (`s0001`–`s0004`), one on Llama-3.1-8B at 32768, and one on Mistral-7B at 30720. Six published selection policies — SnapKV, CriticalKV, TOVA, StreamingLLM, K-Norm and DefensiveKV — are harvested on each through its own implementation's entry point. A *row* is one capture × policy × normaliser × tail level, so six captures give 144 less the four DefensiveKV is absent from on Mistral: 140, the population every result below is counted over. Two analyses were run before the last two captures existed and are reported over the other four — `s0001`, `s0002`, Llama and Mistral, 92 rows — each saying so where it appears (§4.1, §2). Because a policy that evicts perturbs the decode it runs in, the harvest prefills once and lets each policy choose an index set into that frozen history without acting on it, which is what makes the six policies comparable (§A). Each cell's per-query array is split into three disjoint windows, fixed for every row of every table: calibration $[0, 512)$, where curves and contributions are fitted; validation $[512, 1024)$, where an estimator configuration is chosen; and test $[1024, 2048)$, where every reported figure is measured; that the three are consecutive segments of one generation is what §5 is about.

Units and percentages. Every gain is a reduction in the *aggregate* tail risk relative to the uniform allocation at the same total budget and on the same held-out queries, as a percentage of the uniform allocation's own $\mathrm{CVaR}_\alpha(Z)$; uniform is 0% by construction and larger is better, and differences are in *percentage points* of it, which is what "pp" means throughout. The separable baseline they are taken against is the exact minimiser of (1) on Tang et al. (2026)'s convex-hull relaxation, its dual gap certified by a Lagrangian sweep.

Two allocators. Both run the machinery of §5 — convexify each cell's curve, walk the marginal-gain greedy — and differ only in what the curves measure. `SepAlloc` is handed each cell's own CVaR_α, the separable objective of (1) that the field optimises; `TailAlloc` is handed the Euler contribution of (4), which is this paper's method. Every arm below is named for the curves it was built on.

6.2 THE SUBADDITIVITY GAP

Proposition 1 puts the disagreement at $\lambda_\alpha \sigma_i (1 - \varrho_i)$ per cell, largest where β_i is smallest. It is not small: between 6.2% and 38.2% of cells in a row are tail hedges with $\beta_i \le 0$, no row of the 140 is free of them, and a contribution-aware allocation takes a median 27% of those tokens back (Figure 1, Appendices G.5, G.20). Every arm's retained cache records the per-cell budget it used, so a published allocator's own allocation can be scored on our curves. None of these methods was built for that — each does its job by the per-cell objective of Table 9 — so what follows is a transfer test onto a criterion they were never asked to serve. Uniform is 0% by construction and is SnapKV's own allocation; on that transfer, every published allocation we can read lands *below* uniform, in 17 of 18 cells over two models, two tail levels and three scoring policies (Table 7, Appendices G.10, G.12). The eighteenth is layer-DefensiveKV on its own curves at +0.54%, and with 32 of 288 cells below

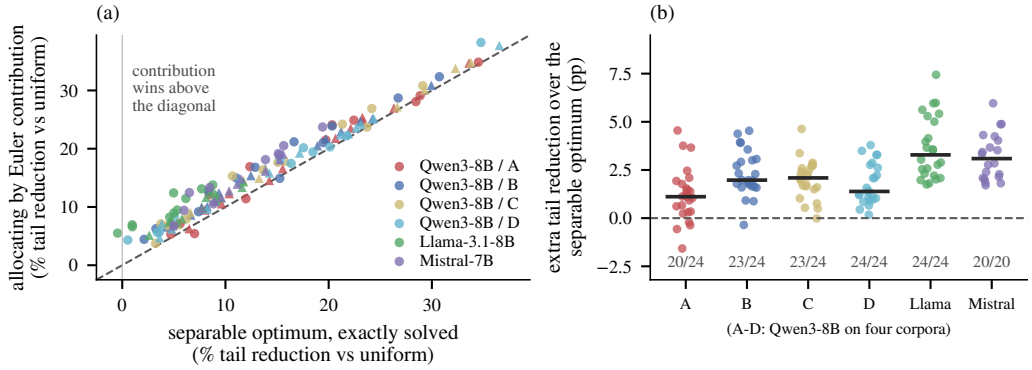

Figure 2: **Allocating by Euler contribution beats the exactly-solved separable optimum on** 134 **of** 140 **rows, by a median of** 2.09 **percentage points.** Held-out queries, estimator chosen on a disjoint validation window. **(a)** One marker per row: horizontal is what the separable optimum reduces the tail by against uniform, vertical is what the contribution-aware allocation reduces it by, both in per cent; circles are $\alpha = 0.99$, triangles 0.95, colour is the capture, dashed line is equality. The separable optimum, though solved exactly, is occasionally *worse* than uniform (-0.4% at its worst); ours never falls below $+3.7\%$. **(b)** The same rows as a difference in percentage points, grouped by capture — this paper's independent unit; bar is the median, count is rows improved. Positive on all six captures, weakest on `s0001` ($+1.11$ pp, $20/24$), under *per-row validation selection*; the fixed recipe of §6.4 is the one that fails on `s0004`.

the lowest measured budget that is an upper bound. At most parity, never a win on this axis: an unequal split of identical convexified cells cannot beat an equal one by Jensen, and a method judged by an objective it never targeted has no reason to (Appendix G.10).

6.3 Tail reduction

Allocating by contribution beats the exactly-solved separable optimum on 134 of 140 rows by a median of 2.09 pp (Figure 2), and the gain is larger at the more extreme tail (median $+2.82$ pp at $\alpha = 0.99$ against $+1.92$ at 0.95), where the closed form is the *worse* model and the better allocator (§5.1); all eight members are selected somewhere (Appendices G.22, G.9).

6.4 Leave-one-capture-out

One *fixed* configuration chosen on five captures and applied unchanged to the held-out sixth — the one analysis here whose replication unit matches the design — gives a median $+1.96$ pp and improves 124 of 140 rows, against $+2.09$ pp and $134/140$ for per-row validation selection and $+2.38$ pp for the per-row maximum on the test window, an upper bound no rule could run. Five of the six folds pick the same configuration: *fit a degree-1 regression of each cell's error on the aggregate over the queries whose aggregate lies above its median, and allocate by the resulting curves* (Appendix D.2). Under a moving block bootstrap sized by the measured θ — an i.i.d. bootstrap is void here, because the tail arrives in episodes — the 90% interval lies wholly above zero in 104 of 140 rows and wholly *below* it in 6, all on `s0004` at $\alpha = 0.99$ (Appendix G.15).

6.5 Calibration corpus

Every allocation above is fitted on a nineteenth-century novel and spent on synthetic retrieval; a fifth calibration corpus drawn from RULER's own distribution closes the transfer matrix, the five captures differing only in the prompt file (Appendix G.2).

At a task metric. On the objective, what specialises is the kind of text, not the capture (Appendix G.2). Driven through the same press at 4096 against one uniform control of 14.59, differing only in the capture fitted on, the *book*-calibrated allocation scores 17.90 and the RULER-calibrated one 15.37: the pair is -2.53 [-3.41, -1.66], clearing on both populations and pointing the wrong way. Registered

Table 1: Task-level result for `SepAlloc`, the separable objective. Matched pairs at compression 0.9375 differing only in the per-head budget vector; Δ is mean `SepAlloc` minus mean `uniform`, intervals 90%. *Items* are the paired questions both arms answered (1300 on RULER; 743 and 734 on LongBench); *tasks* are RULER's 13 types or LongBench's 16; *graded mass* is the fraction of the uniform arm's scores strictly between zero and full marks.

		mean	90% interval, resampling		tasks	graded
Setting	Model	Δ	items	tasks	$\uparrow / = / \downarrow$	mass
RULER 4096, transferred	Qwen3-8B	$+4.21$	$[+3.35, +5.13]$	$[+1.37, +7.42]$	7/4/2	14.1%
RULER 4096, transferred	Llama-3.1-8B	$+1.43$	$[+0.58, +2.31]$	$[+0.11, +2.73]$	8/3/2	11.9%
RULER 32768, native	Qwen3-8B	$+2.38$	$[+1.40, +3.40]$	$[-0.08, +5.76]$	4/4/5	11.4%
LongBench	Qwen3-8B	-0.04	$[-1.10, +1.04]$	$[-0.85, +0.59]$	7/7/7	50.3%
LongBench	Llama-3.1-8B	$+0.35$	$[-0.81, +1.55]$	$[-1.36, +1.34]$	8/2/6	51.6%

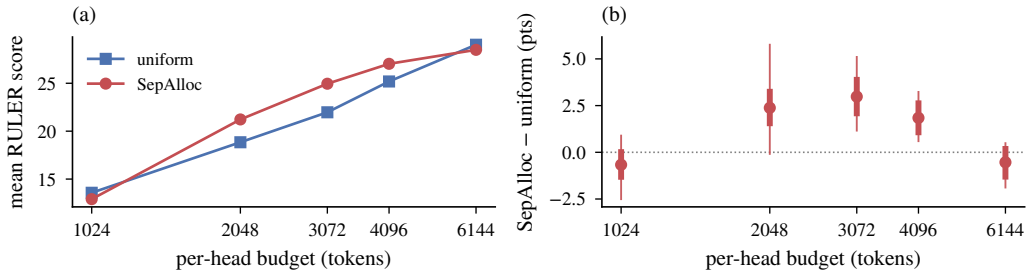

Figure 3: **Score against budget.** Qwen3-8B, RULER at 32768, 1300 paired questions, the allocation rebuilt at each budget. **(a)** Mean RULER score of each arm; `SepAlloc` is the separable-curve arm, not this paper's contribution-built allocator. **(b)** `SepAlloc` minus uniform, 90% paired-bootstrap intervals over questions (thick) and task types (thin): the gain survives doubling and is gone at halving; the 1024 row is censored by the grid (Table 4, Appendix F.1).

prediction B1, written before the second arm ran, said the book arm would land *below* uniform, because the objective calls it 6.47% worse than an equal split here. **The objective ranks these two allocations the wrong way round.** Its row in the resolution table is struck on a prediction registered for this outcome; it is §6.2's disagreement arriving on *our own* vector, with no implementation difference left to blame, and it is now the first of two corpora (Appendices G.2, F.7).

6.6 Task-level evaluation

Does moving the aggregate of per-cell errors — the field's own quantity, not the model's accuracy — move a task metric? Measured in the competitor's own harness at the 6.25% operating point (Appendix F.1).

The arms. Table 1, the iso-quality band and the published-allocation comparisons all score `SepAlloc`: $+4.21$, $+2.14$ and the margin over Ada-KV price the field's own objective against a uniform split, which is a result about that objective and not this paper's claim. `TailAlloc` is scored against it in the A/B below. Items within a task are near-copies, so the cluster bootstrap over tasks is the interval that generalises (Appendices F.1, E.2, G.19).

Replication. Transferred to 4096 `SepAlloc` clears on both populations on two model families, and Mistral-7B agrees in sign without reproducing that; at 32768 only the item-level interval excludes zero (Table 1). On LongBench it gives -0.04 and $+0.35$ with no interval excluding zero, and quadrupling its per-head budget does not move it: $+0.26$ [-0.96, +1.48]. A contribution allocation does move it, below; calibrating on LongBench's own text — which our objective scores 11.52 pp better — is worth -1.14 (Appendices F.1, F.6, F.7); Figure 3 reruns the arms at five per-head budgets with the allocation rebuilt at each.

Published allocations. Ada-KV's own vector is flat across layers to three decimals, so matching it at 32768 isolates the allocation: it costs 1.37 points $[-2.33, -0.43]$ against uniform and `SepAlloc`'s

head-axis arm is $+3.66$ [+2.64, +4.67] over it. Through the same press at 4096, layer-DefensiveKV's own vector *beats* uniform by $+2.07$ [+1.37, +2.80] while our objective calls it 3.80% worse than an equal split: the objective predicts our advantage over it ($+2.14$ [+1.33, +2.95]) and not its failure against uniform, and the candidate mechanism is depth, named rather than repaired. CAKE did not run at all, 1300 refusals of 1300 (Appendices G.23, G.4).

The main comparison. `TailAlloc` against `SepAlloc`, sharing capture, ladder, normaliser, envelope, greedy and calibration window, and differing only in whether the per-cell curves measure the Euler contribution or standalone CVaR: RULER 4096, Qwen3-8B, 1300 paired questions. Both beat uniform — 17.19, $+2.59$ [+1.78, +3.40] against 17.04, $+2.45$ [+1.63, +3.29] — and between them the difference is $+0.15$ [-0.52, +0.83], 5 task types up, 5 flat and 3 down. The reading of that interval, registered before either arm was scored, is that it **did not resolve** — not that the two objectives are equivalent. **And the arithmetic said it could not**: the objective gap prices below the half-width under every rate, needing 5777 questions (Appendix F.2). **LongBench resolves it once in six**: the same choice one provenance field apart is $+2.79$ [+1.50, +4.14] and [+1.03, +4.66] on `s0001`, the separable arm worth -0.20 against uniform and our objective ranking the two the wrong way round by 3.36 pp; on the other five captures none clears under both refusal policies (§§F.7, F.8).

Benchmark resolution. The median half-width of the 57 scored 90% intervals at $n = 1300$ is ± 0.83 points; the transmission from objective to task is between 0.067 and 0.298 points per objective percent. **Two of the five allocating-at-all rows, once measured, came out with the opposite sign to their prediction** and are struck. **Everything else this method chooses sits under the line by that arithmetic**: at the lowest rate the median of the twelve would need 6892 questions, and the contribution-versus-separable choice would need 117755, $91\times$ the compute per arm. The field reports benchmark means for allocation methods, usually without intervals and at one n, where those means are not evidence about the choices they are attached to; this paper did too until it did this arithmetic. The rate itself is checked rather than trusted: on the seven rows carrying a measured task effect it **gets the sign wrong on 4** (Appendix F.4). The effect is small because the greedy reads only the *order* of marginal gains, which the separable objective gets almost right; the recommendation to spend the effort where rank agreement is low is withdrawn (Appendix G.1). Nor could the choice have been priced without running it: ten capture statistics, β, four rank-agreement forms and a transmission rate all failed as predictors of it, the rate wrong in sign four times in eleven (Appendix D.4: fourteen refutations).

7 Conclusion

A sum of per-head risks, the objective every published KV-cache allocator minimises, is the model's aggregate tail risk only when every head fails on the same queries. On six captures the bound is loose by a median 55%, and the quantity that governs the aggregate tail is each head's Euler risk contribution. One fixed recipe, held out capture by capture, beats the separable optimum by 1.96 pp and reaches a task metric on RULER. Whether or not the correction buys task points, an allocator that wants the aggregate tail has to be handed the aggregate's own gradient. The allocator itself helps on two captures of six, we cannot say which in advance, and most design choices here sit below what the benchmarks resolve (Appendix A).

Reproducibility statement

Every number in this paper is produced by a named script in `scripts/` from a named JSON in `data/`, both in the anonymised supplementary bundle submitted with this paper; Appendix C tabulates which script produces which number. The JSON carries the capture, policy, normaliser, tail level and cell count of every row, so any table or figure here can be recomputed without the per-query harvest, which is six captures of roughly ten gigabytes each and is too large to include. The three window conventions, the fixed calibration/validation/test split that every reported figure uses, and the places a script declines to return a number it could have returned are in Appendix C; the interval construction is in Appendix G.15 and the allocator's library gates, including the mutation that verifies each one, are in Appendix C. One exception is recorded rather than hidden: `drift.py` prints and writes no JSON, so the two window medians quoted in Appendix G.16 are the only figures in this paper that require rerunning a script against the harvest to check. A build gate re-derives every quoted

figure that can be recomputed from `data` and fails if the prose and the data disagree; it has caught nine edits of ours that silently dropped a qualification. Because it holds one check per cell of the length split in Appendix F.1, it also refuses a draft that reports some of those cells and not all of them. Every build is run from a cleared latexmk cache, because `latexmk` reports "all targets up-to-date" and exits zero on a stale one.

The arms reproduce, and we checked rather than argued it. Every interval here is a bootstrap over questions, which is the whole uncertainty only if a run repeats itself. `kvpress` decodes with an `argmax` in its own loop and never calls `generate`, so a model's shipped sampling configuration is never read, which is worth checking because Llama-3.1-8B ships `do_sample: true` at temperature 0.6. Re-running one RULER arm on the same card, venv and allocation returned all 1300 predictions byte-identical and every subtask score unchanged (`runs/rerun.md`); the float non-determinism it was testing for did not appear. One arm, one card: it does not establish the same for the A800 or for LongBench's scorer, and the registration said so before the run.

USE OF LARGE LANGUAGE MODELS

LLM-based coding agents were used throughout this work and their use was extensive rather than incidental. Taking the policy's list of activities that must be reported in turn: agents helped develop the conceptual framework (importing Euler risk contributions and comonotonic bounds from portfolio risk into this setting), proposed and refined hypotheses, formulated the mathematical claims, drafted the proposition of §4 and the proof and counterexample of Appendix G.6 — including finding that an earlier statement of that proposition was false, wrote the harvest, allocation, scoring and bootstrap code, cleaned and reformatted the harvested arrays, ran every experiment on the two rented GPU machines, produced the figures, and drafted and revised the text of this paper, including this sentence. They also did the qualitative reading that sits between an experiment and a claim: classifying failure modes, judging which of two disagreeing measurements was measuring the wrong thing, and writing the registered predictions that several results then refuted. No synthetic data was generated: every number comes from a measured forward pass.

Research direction and the choice of which claims to put at risk rest with the human authors, who approved each experiment before it ran and each withdrawal after it. The adjudication of individual conflicting results was in the first instance frequently done by an agent, and what stands behind those judgements is the verification described next, not a claim that a human re-derived each one.

The risk this creates is a plausible number that no experiment produced, and the design of the repository is a response to it. Every quoted figure that can be recomputed from `data` is re-derived by `verify_numbers.py` at build time and the build fails on a mismatch — one check per such figure, with no way to report a subset — and the gate's own wiring is exercised by mutation tests rather than trusted. It exists because the failure it guards against was observed twice, and both times it was re-derivation and not reading that caught it.

The authors are responsible for everything in this paper, including anything an agent produced that the gates did not catch.

REFERENCES

Philippe Artzner, Freddy Delbaen, Jean-Marc Eber, and David Heath. Coherent measures of risk. *Mathematical Finance*, 9(3):203–228, 1999.

Daniel Bauer and George Zanjani. The marginal cost of risk, risk measures, and capital allocation. *Management Science*, 62(5):1431–1457, 2016. doi: 10.1287/mnsc.2015.2190.

Betina Berghaus and Axel Bücher. Weak convergence of a pseudo maximum likelihood estimator for the extremal index. *The Annals of Statistics*, 46(5):2307–2335, 2018. doi: 10.1214/17-AOS1621.

Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

Peter Bühlmann and Bin Yu. Analyzing bagging. *The Annals of Statistics*, 30(4):927–961, 2002.

Michel Denault. Coherent allocation of risk capital. *Journal of Risk*, 4(1):1–34, 2001.

Alessio Devoto, Yu Zhao, Simone Scardapane, and Pasquale Minervini. A simple and effective L2 norm-based strategy for KV cache compression. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024. arXiv:2406.11430.

Bradley Efron and Trevor Hastie. *Computer Age Statistical Inference: Algorithms, Evidence, and Data Science*. Cambridge University Press, 2016. doi: 10.1017/CBO9781316576533.

Awi Federgruen and Henri Groenevelt. The greedy procedure for resource allocation problems: necessary and sufficient conditions for optimality. *Operations Research*, 34(6):909–918, 1986.

Yuan Feng, Junlin Lv, Yukun Cao, Xike Xie, and S. Kevin Zhou. Ada-KV: Optimizing KV cache eviction by adaptive budget allocation for efficient LLM inference. In *Advances in Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=tcisuhGsQZ. arXiv:2407.11550.

Yuan Feng, Haoyu Guo, Junlin Lv, S. Kevin Zhou, and Xike Xie. DefensiveKV: Taming the fragility of KV cache eviction in LLM inference. In *International Conference on Learning Representations*, volume 2026, pp. 43407–43432, 2026a.

Yuan Feng, Junlin Lv, Haoyu Guo, Yukun Cao, S. Kevin Zhou, and Xike Xie. CriticalKV: Optimizing KV cache eviction from an output perturbation perspective. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026b. arXiv:2502.03805.

Christopher A. T. Ferro and Johan Segers. Inference for clusters of extreme values. *Journal of the Royal Statistical Society, Series B*, 65(2):545–556, 2003.

Hans Föllmer and Alexander Schied. *Stochastic Finance: An Introduction in Discrete Time*. De Gruyter, fourth edition, 2016. doi: 10.1515/9783110463453.

Bennett Fox. Discrete optimization via marginal analysis. *Management Science*, 13(3):210–216, 1966.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W. Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. KVQuant: Towards 10 million context length LLM inference with KV cache quantization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H. Abdi, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. MInference 1.0: Accelerating pre-filling for long-context LLMs via dynamic sparse attention. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Naoki Katoh, Akiyoshi Shioura, and Toshihide Ibaraki. Resource allocation problems. In *Handbook of Combinatorial Optimization*, pp. 2897–2988. Springer New York, second edition, 2013. doi: 10.1007/978-1-4419-7997-1_44.

M. R. Leadbetter, Georg Lindgren, and Holger Rootzén. *Extremes and Related Properties of Random Sequences and Processes*. Springer Series in Statistics. Springer, 1983.

Yubo Li and Yidi Miao. CONF-KV: Confidence-aware KV cache eviction with mixed-precision storage for long-horizon LLM, 2026. arXiv:2605.24786.

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*, 2024. arXiv:2404.14469.

Wenhao Liu, Hao Shi, Yunhe Li, Weizhi Fei, Xiangyuan Wang, Mengzhe Ruan, Hanxu Hou, Peisong Wang, Linqi Song, and Shuang Qiu. ReasonAlloc: Hierarchical decoding-time KV cache budget allocation for reasoning models, 2026. arXiv:2606.11164.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. KIVI: A tuning-free asymmetric 2bit quantization for KV cache. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

Alexander J. McNeil, Rüdiger Frey, and Paul Embrechts. *Quantitative Risk Management: Concepts, Techniques and Tools*. Princeton University Press, revised edition, 2015.

Paul J. Northrop. An efficient semiparametric maxima estimator of the extremal index. *Extremes*, 18 (4):585–603, 2015. doi: 10.1007/s10687-015-0221-5.

Matanel Oren, Michael Hassid, Nir Yarden, Yossi Adi, and Roy Schwartz. Transformers are multi-state RNNs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024. arXiv:2401.06104.

Ziran Qin, Yuchen Cao, Mingbao Lin, Wen Hu, Shixuan Fan, Ke Cheng, Weiyao Lin, and Jianguo Li. CAKE: cascading and adaptive KV cache eviction with layer preferences. In *The Thirteenth International Conference on Learning Representations*. OpenReview.net, 2025.

R. Tyrrell Rockafellar and Stanislav Uryasev. Optimization of conditional value-at-risk. *Journal of Risk*, 2(3):21–41, 2000.

Alexander Shapiro, Darinka Dentcheva, and Andrzej Ruszczyński. *Lectures on Stochastic Programming: Modeling and Theory*. Society for Industrial and Applied Mathematics, third edition, 2021. doi: 10.1137/1.9781611976595.

Yiqun Shen, Song Yuan, Zhengze Zhang, Xiaoliang Wang, Daxin Jiang, and Nguyen Cam-Tu. LAVa: Layer-wise KV cache eviction with dynamic budget allocation, 2025. arXiv:2509.09754.

Zhiyuan Shi, Qibo Qiu, Feng Xue, Zhonglin Jiang, Li Yu, Jian Jiang, Xiaofei He, and Wenxiao Wang. HeteroCache: A dynamic retrieval approach to heterogeneous KV cache compression. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, 2026. arXiv:2601.13684.

Richard L. Smith and Ishay Weissman. Estimating the extremal index. *Journal of the Royal Statistical Society, Series B*, 56(3):515–528, 1994.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. Quest: Query-aware sparsity for efficient long-context LLM inference. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

Ziyao Tang, Pengkun Jiao, Xinhang Chen, Wei Liu, Shiyong Li, and Jingjing Chen. Predicting future utility: Global combinatorial optimization for task-agnostic KV cache eviction, 2026. arXiv:2602.08585v2.

Dirk Tasche. Risk contributions and performance measurement. Technical report, Lehrstuhl für mathematische Statistik, Technische Universität München, 1999. First draft 29 June 1999, revised 18 February 2000. No DOI is issued for this working paper.

Zhongwei Wan, Xinjian Wu, Yu Zhang, Yi Xin, Chaofan Tao, Zhihong Zhu, Xin Wang, Siqi Luo, Jing Xiong, Longyue Wang, and Mi Zhang. D2O: Dynamic discriminative operations for efficient long-context inference of large language models. In *The Thirteenth International Conference on Learning Representations*, 2025. arXiv:2406.13035.

Chaojun Xiao, Pengle Zhang, Xu Han, Guangxuan Xiao, Yankai Lin, Zhengyan Zhang, Zhiyuan Liu, and Maosong Sun. InfLLM: Training-free long-context extrapolation for LLMs with an efficient context memory. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024a.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations*, 2024b. arXiv:2309.17453.

Enshuai Zhou, Yifan Hao, Chao Wang, Rui Zhang, Di Huang, Jiaming Guo, Xing Hu, Zidong Du, Qi Guo, and Yunji Chen. LKV: End-to-end learning of head-wise budgets and token selection for LLM KV cache eviction, 2026. arXiv:2605.06676.

A LIMITATIONS

Benchmark dependence. RULER moves on two model families; LongBench for one allocation on one *capture* of six — though against an equal split it clears on two of those six and its separable control on none (§F.8). At 4096 the two-axis arm clears both populations and beats the head-axis arm; at 32768 only the question population, and the layer axis alone is null at both. Two arms beat uniform where our objective says they lose: layer-DefensiveKV and LongBench's book-calibrated vector (§F.7); the first has a candidate mechanism and not a test, only one front-loads (§G.4).

Unit of replication. The 140 rows are transformations and policies over those six, so a win count describes consistency, not a test; only §6.4's leave-one-capture-out matches the design, and corpus and model stay confounded outside Qwen — on RULER the objective choice resolves *against* us as often as for us, both non-Qwen captures clearing negative and two Qwen ones positive, a split we registered in advance as unpredicted and cannot break with two families (§F.8). `s0004` is adverse — all six rows wholly below zero, yet $24/24$ under per-row selection, so what fails is the fixed recipe and not the family. Held out, the gain fell from $+2.23$ to $+1.96$ pp when those two captures were added; we report the smaller. **Nor can we say *which* captures it helps**: ten statistics registered in advance order the six no better than $|\rho| = 0.60$, where only a near-perfect ordering would count at this n (§D.3). And two of the six rows cannot be scored without naming a refusal policy, which moves them by more than the benchmark resolves (§F.9).

Model scale. The six captures span three families but one order of parameter count. Proposition 1 is a statement about a joint law and carries no dependence on model size, but every quantity in it is measured at one scale here, and grouped-query attention moves the cell count with the model, so the 288 and 256 cells of §6.1 are not a fixed unit. We do not claim these magnitudes hold at the 30B class the field now evaluates on. Given that ten registered capture statistics fail to predict which of the six captures the method helps on (§D.3), we expect that question to need measuring rather than arguing, and we have not measured it.

Benchmark resolution. The choice between the two objectives is not small, but RULER at 4096 does not settle it: doing so would need 117755 questions at the most conservative estimate, and one LongBench capture settled it in 742. LongBench resolves it on one Qwen3-8B capture and on none of the other five; one provenance field apart, the two objectives are $+2.79$ [+1.50, +4.14] apart there on both populations, while our objective ranks them the wrong way round by 3.36 pp — §6.5's reversal arriving a second time, on a second corpus. The largest task-level margin this paper reports, $+4.21$ over a uniform budget, is won by an arm that allocates on standalone risk rather than on the contribution, and that arm is null on LongBench on two families (§6.6, App. F.1).

Scope of the home advantage. (§6.5, App. F.7): the matrix bounds what the objective says a shift costs, not what it costs. **Three further bounds**: no policy is measured as deployed (§6.1), the elliptical form is biased yet allocates better (§5.1), and the aggregate is *defined* as a sum of per-cell errors, perhaps why objectives do not reach tasks (App. G.7).

The measurement claim. Not that anyone deploy this allocator — it helps on two captures of six, and §D.4 is fourteen registered attempts to say which. The independent claim is the measurement one: a per-head sum bounds the aggregate tail only under comonotonicity, is loose here by a median 55%, and every allocator we read is tuned against it. That is wrong whether or not the correction buys task points.

B GUIDE TO THE APPENDICES

Reviewers are not required to read an appendix. This one is long because every number in the body is re-derived from a named JSON by a named script at build time, and the derivations are kept rather than summarised. The table below is the short path through it: each row is a question a reader of §§4–6 might want settled, and the appendix that settles it.

Question	Settled in
Does allocating by contribution beat the exactly-solved separable optimum, and under which selection rule?	§D.2, §G.15
Does the correction reach a task metric, and on which benchmarks?	§F.1, §F.2
Why is the task-level effect small?	§G.1
Can these benchmarks resolve the choice at all, and at what sample size?	§F.4, §F.2
What did we predict before the data existed, and what refuted it?	§D.1, §D.4
Why are quantisation, retrieval sparsity and merging not compared against?	§G.24
Is the convex envelope necessary, or would the contribution alone do?	§G.13
What does forming the contribution curves cost?	§E.1
How is every number in the paper reproduced?	§C

C REPRODUCTION

C.1 SCRIPTS

Script	Data	Numbers it produces
`aggcvar3.py`	`aggcvar3_*`	comonotonicity gap; fitting-level sweep
`bigjump.py`	`bigjump_*`	Hill index, tail concentration (§4.1)
`betarc.py`	`betarc_*`	closed form against the empirical contribution
`betadiag.py`	`betadiag_*`	β spread, hedge counts (Table 5)
`hedgemap.py`	`hedgemap_*`	hedge stability across policies, depth profile
`rcshrink.py`	`rcshrink_*`	the estimator family (Figure 2)
`loco.py`	`loco.json`	leave-one-capture-out (§6.4)
`drift.py`	—	the eight-window stationarity check
`extremal.py`	`extremal_*`	θ, cluster sizes, effective events
`blockci.py`	`blockci_*`	block-bootstrap intervals (§6.4)
`guarantee.py`	—	the DKW sample-size requirement
`cost.py`	`cost_s0001.json`	the timings of §5
`tasktab.py`	`tasktab_*`	the task tables and both bootstraps (Table 1)
`axispredict.py`	`axispredict_*`	the axis decomposition (Table 6)
`puballoc.py`	`puballoc_*`	published allocations scored (Table 7)

`aggcvar2.py`, discussed under Refusals below, is an earlier variant retained for the record and produces no reported number.

C.2 WINDOWS

Two window conventions appear in this work and they are not interchangeable. The estimator and contribution scripts use calibration $[0, 512)$, validation $[512, 1024)$ and test $[1024, 2048)$, which is the split §6.1 states and every number in this paper uses. An earlier family of scripts concerned with the allocator rather than the estimator uses calibration $[0, 768)$, an online probe window $[768, 1280)$ and test $[1280, 2048)$; none of its numbers is quoted here, and the split is recorded so that a reader comparing the two repositories is not misled by a shared file naming scheme. In both conventions the windows are fixed regardless of any other setting, so every variant in a table is scored on identical data.

C.3 REFUSALS

Each of these is a place a script declines to return a number it could have returned. They are listed because a refusal that is not written down tends to be quietly removed later.

1. `extremal.py` defines the exceedance set as the top $k = \lceil (1-\alpha)Q \rceil$ order statistics — the same points CVaR_α averages — rather than as $Z >$ the interpolated α-quantile. With near-ties in the upper tail the strict form returned an empty set on Mistral and four exceedances where twenty were expected on Llama. A definition that can silently return nothing is not one worth keeping.

2. `guarantee.py` returns no bound, rather than a number, when $\varepsilon \geq 1-\alpha$ makes the DKW bound collapse to the sample maximum. It is true and says nothing, so it is not reported as a bound.

3. `rcshrink.py` and `aggcvar3.py` print the maximum over configurations taken on the test window beside the validation-selected figure, so the size of the selection bias is visible in every row rather than inferred.

4. `aggcvar2.py` is kept in the repository and labelled non-reportable in its own docstring: it selected on the test window and swept 14 of 14 rows of the subset it ran on. It is retained because the size of that bias — median 0.09, maximum 3.14 pp against the protocol of §6 — is itself a result.

5. `blockci.py` reports both block-length rules on every row and whether they agree, rather than reporting the one that gives the tighter interval.

6. The allocation solver withholds a gain figure when the requested total is not reachable on the finite level grid, and reports its own dual gap when it is.

C.4 LIBRARY GATES

The allocator machinery is a library with tests rather than a script, and the guards below were each verified by a mutation that turns the test naming it red. `kvtail.envelope` refuses a curve on fewer than two levels, levels that are not strictly increasing, a curve missing a level, and a non-finite value, and asserts that cell keys keep insertion order rather than being sorted lexicographically — which matters because cell keys are `"<layer>.<head>"` strings and lexicographic order puts `"10.0"` before `"2.0"`. One *equivalent* mutant is recorded and declined: flipping the hull-pop comparison from >= to > keeps collinear points that the interpolation then passes over, giving bit-identical output. The integration gate is the one that carries weight: greedy on the convexified curves is pinned to the Lagrangian sweep of §6.1, which reaches the same allocation by a different algorithm. The suite runs 1426 passed and 1 skipped at the commit this paper is written from.

D REGISTERED PREDICTIONS

D.1 PRE-REGISTRATION

Each experiment below was written up — the arms, the quantity, the branches, and what each branch would mean — and committed to version control *before* the arm that would settle it was run. The files are `runs/*.md` in the supplement, each carrying the commit that fixed it and, where the registration turned out to be wrong about something, an addendum beside it rather than an edit to it. Results are reported here where they belong; this table exists because the record reads differently collected than dispersed.

run	registered	fired	outside all branches
`bookarm`	B1–B3, does the objective's sign transmit	B3	no
`iso6144ctx4096`	C1–C3, replication at a second context	C1	no
`objiso`	M1–M3, the objective isolated	M1	no
`sevenrung`	A, objective side	A held	no
	B1–B3, task side	none	**yes**
`llamalb`	L1–L4, a second model family	L3	no
`betaxfam`	B1–B3, does β predict divergence	none	**yes**
`mistrallb`	P1–P4, a third model family	P3	no
`s0002lb`	S1–S3, a second capture of one model	S2	no
`s0034lb`	T1–T3, two more captures	T2 / T1[‡]	no
`rulersq`	U1–U2, V1–V2, a second benchmark	U1, V1	no[§]
`whenhelps`	ten candidate predictors, one rule	none	no
`sqsix`	A1–A4, B1–B2, is "exactly one" a rule	A4, B1	no
`mistralruler`	M1–M2, the last empty cell	M1[¶]	no
`rerun`	D1–D3, does an arm reproduce itself	D1	no
`lbcalpair`	R1–R3, L1–L3, corpus varied with model fixed	R2, L1	no

Two of the fifteen landed outside *every* branch considered in advance, and both say so in the column that records it.

Five more registrations are in the supplement and not in the table above, because they do not have its shape. They are listed here so that "every" means every. `runs/iso6144.md` registers P1–P3 on the 2048–6144 matched-objective pair and `runs/lbcapture.md` registers L1–L3 on whether the LongBench null and the RULER transfer failure are the same thing; both were committed before their arms ran and both are read where they are used, in §F.5 and §F.7 — **L2 is falsified there**, on vectors matched to one provenance field, with the objective and the task scored on the same corpus. `runs/lbcap2.md` refines `lbcapture`'s protocol once the corpus existed and adds no branch of its own; `runs/lbbook.md` registers an arm's *design* before it ran rather than a prediction about its outcome; and `runs/laddervscorpus.md` is a note recording a reading we then had to withdraw — it attributed a 3.43-point gap to the ladder, the ladder was held fixed, the gap stayed at 2.53, and §G.2 says so.

[‡] **`s0034lb` cannot be scored without naming a refusal policy.** Its branches were written before the confound of §F.9 was known: T2 ("one or both clear") fires under the default policy and T1 ("both are null") under the pessimistic one. We report it as split rather than choosing the policy that makes the registration look better, and note that the policy which makes it look better is the one that was the default when the branches were written.

[§] **`rulersq`'s registered branches all concern the comparisons against an equal split, and both fired.** Its striking result — the cross-benchmark square of §F.8 — is in the cell for which the registration explicitly declined to predict, on the stated grounds that RULER had already failed to resolve that comparison once and predicting the same failure twice is not a prediction. That reasoning was wrong. The cell is not outside a branch; there was no branch, and a reader should discount it accordingly against the rows above it that were called in advance.

[¶] **`mistralruler`'s M1 fired in the direction the registration declined to predict.** M1 asked only whether the cell would resolve, and it does. The same file recorded, before the run, that a negative result would make two of two non-Qwen families favour the separable objective, that this would be a pattern worth naming, and — in those words — that it was being written down then so that it could not afterwards be named as though it had been foreseen. It came out negative. §F.8 reports the pattern as unpredicted, which is what it is.

advance. In `sevenrung` the three task-side branches were placed at $+4.21$, $+1.68$ and $+1.01$ to $+1.21$ points and the arm returned -0.67; in `betaxfam` the mechanism was refuted on all three of its parts at once. Being outside the registered range is reported as its own outcome and not rounded to the nearest branch.

The record has a shape, and it is narrow. It is not that objective-side predictions held and task-side ones failed: `betaxfam`'s refuted B1 and B2 are objective-side, involving no benchmark at all, and `objiso`'s M1 and `iso6144ctx4096`'s C1 are task-side and both fired. What does hold is about the transmission band, the one instrument here for carrying an objective difference to a task metric.

It has been registered against **eleven** times and its record is **four sign failures, five hits and two under-predictions, the worse of them by** $200\times$ — it now misses more often than it hits:

run	band predicted	measured	
`bookarm`	-0.94 to -1.93	$+3.31$	wrong in sign
`sevenrung`	$+1.01$ to $+1.21$	-0.67	wrong in sign
`mistrallb`	$+0.34$ to $+1.51$	$+0.82$	inside
`s00021b`	$+0.20$ to $+0.88$	$+0.51$	inside
`rulersq`	$+0.20$ to $+0.88$	$+2.86$	under by $3.3\times$
`sqsix`, `s0004`	$+0.14$ to $+0.63$	$+0.74$	just above
`sqsix`, `s0003`	$+0.01$ to $+0.03$	$+\mathbf{5.65}$	**under by** $200\times$
`sqsix`, llama	$+0.40$ to $+1.78$	-4.86	wrong in sign
`mistralruler`	$+0.34$ to $+1.51$	-0.85	wrong in sign
`lbcalpair`, RULER	$+0.24$ to $+1.06$	$+0.46$	inside
`lbcalpair`, LongBench	$+0.24$ to $+1.06$	$+1.11$	just above

An earlier version of this paragraph said the band had missed every time and missed in sign both times. That was written when the record held only the first two rows and it is left recorded here rather than quietly replaced, because a claim about a record that is still being added to is exactly the kind this paper spends its length warning about. The middle two rows are also the two where the band was asked to price the *objective choice itself* on a fresh capture, and it placed both — including that neither would clear the resolution (§F.8).

The last three rows were registered as a test of this instrument and it failed them. `runs/sqsix.md` priced all three before they ran, from the same provenance field, and named the consequence of each outcome in advance. `s0003` is the sharpest: its two vectors are 0.095 pp apart on the objective, the smallest gap in the study, so the band placed it *last of all six captures* at a ceiling of $+0.03$ points. It returned $+5.65$, clearing on both populations — the largest matched effect in this paper. The capture the arithmetic ranked least promising produced the biggest measurement, at two hundred times its predicted ceiling.

What that costs the twelve-choice table. §6.6 multiplies each design choice's objective gap by this band and concludes that eleven of twelve fall below the ± 0.83 resolution. Three choices the band priced below resolution have since been measured: one came in at $200\times$ its ceiling and cleared, one at $3\times$ and cleared, one with the opposite sign and cleared. **The table is therefore a record of what the arithmetic said, not evidence about the choices it is attached to**, and we now read it that way. The conclusion it was built to support — that a design space can produce effects below what an evaluation resolves — survives as a possibility the data still shows in places, but it is no longer established *by this pricing*, because the pricing is wrong often enough and largely enough that a small predicted effect carries almost no information about the measured one.

This is a stronger negative than the one we set out to report, and it falls on our own instrument first. The honest generalisation is not that these effects are too small to see. It is that **a design choice cannot be priced without running it**, and the best rate we could fit — on our own data, on the benchmark it was fitted on — does not survive contact with the measurements it was fitted to predict.

The fifth row is the fourth row's objective gap, measured again on a second benchmark, and this is the sharpest thing we can say about how the rate depends on the benchmark. `s0002`'s two vectors are 2.95 pp apart whichever benchmark they are scored on, so the band issues one prediction, $+0.20$ to $+0.88$, for both. LongBench returned $+0.51$ and RULER returned $+2.86$. **A single number derived from the objective gap cannot predict both, so the transmission rate is not a property of the allocation pair; it is a property of the pair and the benchmark together.** §F.4 fits it on RULER and the paper applies it to LongBench throughout, which these two rows show is not licensed. We leave the band where it is, with its width, because it is the only instrument here that carries an objective difference to a task metric at all — but every use of it in this paper should be read as an order of magnitude and not a prediction.

Where a registration recorded which branch its author expected, it was right in `objiso` and wrong in `llamalb`, and in the latter the expectation was for the two branches the arm refuted.

This is also the honest reading of what registration bought. It did not make the predictions better. It made it impossible to describe them afterwards as though they had been.

D.2 LEAVE-ONE-CAPTURE-OUT

The configuration for each fold is chosen by the median, over the rows of the five training captures, of that configuration's validation score net of the row's separable baseline. No row of the held-out capture enters the selection at any point.

Held-out capture	Configuration picked	Median	Rows improved
llama, 32768	`reg q0.5 d1`	$+3.15$ pp	$24/24$
mistral, 30720	`reg q0.5 d1`	$+2.62$ pp	$20/20$
s0001, 32768	`reg q0.5 d1`	$+1.01$ pp	$21/24$
s0002, 32768	`reg q0.5 d1`	$+2.35$ pp	$24/24$
s0003, 32768	`reg q0.5 d1`	$+1.77$ pp	$22/24$
s0004, 32768	`reg q0.8 d1`	$+0.16$ pp	$13/24$
Pooled	—	$+1.96$ pp	$124/140$

For the comparison made in §6.4: within a fold, the spread of the median gain across all eight configurations is 0.98 pp on llama, 1.61 on mistral, 0.87 on s0001, 0.51 on s0002, 0.66 on s0003 and 1.35 on s0004, against a 2.99 pp spread across the six folds. Adding two captures widened that comparison rather than narrowing it: which capture you hold out still matters about twice as much as which member of the family you pick, and now by a larger margin than it did at four.

The `s0004` fold is the one that pays for saying this out loud. It is the only fold that picks a different configuration and the only one that collects almost nothing, and it is also the fold whose training set is the only one lacking `s0004` — which is to say the choice of 0.5 over 0.8 is carried by that capture and not by a majority that would survive without it. Five-of-six agreement is therefore weaker evidence than the count suggests, and we report the count with the mechanism rather than the count alone.

D.3 ONE WEAKER CAPTURE

Of the six captures, two are weak, and they are weak in different ways and under different protocols. Both readings below are of the *fixed recipe* of §6.4; under per-row validation selection neither capture is weak at all — `s0001` improves $20/24$ rows and `s0004` improves $24/24$ (Figure 2b) — and that contrast is the finding, not a caveat to it.

`s0001` is *quiet*: $11/24$ rows with an interval wholly above zero, median $+1.00$ pp, and nothing wholly below. `s0004` is *adverse*: $8/24$ above, and it holds all six rows in the paper where the interval lies wholly below zero, all at $\alpha = 0.99$ (§6.4). It is also the only leave-one-out fold that picks a different configuration and the only one that collects almost nothing ($+0.16$ pp, $13/24$). Two instruments that share no machinery — the block bootstrap and the leave-one-capture-out fold — agree on `s0004`, so the failure is a property of the capture under a fixed recipe and not of one estimator.

The direction of that contrast matters for what to run. A capture on which every row improves when the configuration is chosen per row, and on which six rows are reliably worse when one configuration is fixed across all captures, is not a capture the method cannot handle — it is a capture the *recipe* cannot handle. What §6.4 buys in simplicity it pays for here, and 1 capture in 6 is the measured price.

We looked for a predictor among the theory's own quantities: the comonotonicity gap, the fraction of tail hedges, the dispersion of β_i, and the extremal index. The most promising was θ, which at the capture level correlated with the advantage at $\rho = -0.80$ and had a mechanism to go with it. It did not survive: centred within capture the sign reversed to $+0.43$, and centred within capture *and* tail level — the level drives both quantities — it was $\rho = -0.011$, permutation $p = 0.91$. That analysis ran over the first four captures and we have not re-run it on six; we leave it as it was rather than restate it over a row set it never saw, because a relationship that already changed sign with the slicing does not become one by being recomputed.

What the two new captures did buy is the shape of the failure rather than a predictor for it. At four captures the method had a weak capture; at six it has a *failure region*, and the region is a tail level rather than a corpus — $\alpha = 0.99$ on `s0004`, with the same capture behaving normally at $\alpha = 0.95$.

That is a more useful object than a capture-level correlation would have been, and it is the one thing here that four captures could not have shown.

Candidate predictors. The paragraph above is a post-hoc search and reads as one. `runs/whenhelps.md` therefore fixed the whole procedure — candidates, outcome, test, and the threshold that would count as a finding — and was committed before `s0003` and `s0004` had task numbers at all. Ten capture-level statistics: β dispersion, hedge fraction, median β, two rank correlations between standalone risk and the contribution, the hedge token delta, the objective's own predicted gain, the objective gap, cell divergence between the matched pair, and budget moved between them. Outcome: the contribution allocation against the uniform control, one number per capture, six captures. Test: Spearman, with the threshold set by what $n = 6$ can support — a perfect ordering is $p = 0.0028$, which ten tests take to 0.028 under Bonferroni, so **only a perfect or near-perfect ordering counts and $|\rho| \leq 0.943$ was fixed in advance as no predictor found**.

Nothing came close, under either refusal policy. The largest $|\rho|$ against the registered outcome is 0.600 (`beta_median`) under the default policy and 0.429 (`hedge_fraction`) under the pessimistic one. All forty correlations — ten candidates, two refusal policies, and both the registered outcome and the matched difference reported alongside it — are in `data/whenhelps.json`.

This is a real and reportable negative, and it is the one a practitioner feels: **"it helps on two captures of six" cannot currently be turned into "it helps when . . ."**. None of the quantities this paper's own theory supplies — not the β dispersion that Proposition 1 makes the governing one, not the hedge fraction, not the objective gap — orders the captures by how much the allocation is worth on them. A reader deciding whether to use this method on their own workload has no capture-level test to apply in advance, and we would rather say so than leave the absence to be noticed.

The threshold is the reason this reads as a negative and not a finding. Two of the forty tie for the largest magnitude, at exactly $\rho = -0.886$ ($-31/35$, which is where a Spearman on six points lands): `budget_moved` against the matched difference under the pessimistic policy, and — **the one a reader would most want to be real** — `objective_gap` against the matched difference under the default one, at an exact permutation $p = 0.033$ before correction.

That second one says the objective gap between the two allocations predicts how far apart LongBench puts them, in rank, on six captures. It is the paper's own theory delivering its own headline mechanism, and we do not report it as a result. It is one of forty numbers computed from six points, its p is 0.33 after Bonferroni, it is not stable across the refusal policies ($\rho = +0.143$ under the other one, which is a sign flip), and the rule that makes it not a finding was fixed before `s0003` and `s0004` had task numbers. We report it because a reader is entitled to know what the search turned up, and because it is the clearest illustration in this paper of what the registrations are for: without the threshold written down in advance, this is the sentence that would have been the paper's second contribution.

D.4 Refuted predictions

The refutations below are argued where they happened. They are collected here because the count is the result: a single failed prediction is a footnote, and fourteen of them, each written down before the arm that settled it ran, is a statement about whether a design choice in this space can be reasoned about instead of measured.

Read with §D.3 and §F.4, which are the same finding by two other routes: ten capture statistics registered in advance order the six captures no better than $|\rho| = 0.60$ where the registered threshold was a perfect ordering, and the transmission band — the one instrument here for carrying an objective difference to a task metric — has been registered against eleven times and is wrong in sign on four of them.

1. *pooling several calibration corpora beats picking one* — pooled +1.62% against an oracle single corpus's +2.92%.
2. *the low-budget failure is the measurement grid pinning cells* — relieving it moved the floor 53.5% → 27.8% and the objective +3.72% → +2.14%, the wrong way.
3. *fitting the contribution shallower helps at every tail level* — at α 0.95, leave-one-capture-out gives +2.12 chosen against +2.12 shipped.

4. *the size of β's non-constancy predicts what the correction is worth* — capture-level Spearman -0.800, the opposite sign to the one the reading requires.
5. *some configuration makes contribution-versus-separable resolvable at $n = 1300$* — best of 33 configurations is +2.95 pp, predicting 0.43 points at the lowest rate against a 0.83 half-width.
6. *measuring a finer budget ladder buys a measurable task gain* — +1.46% of objective, 0.21 to 0.43 predicted points, below the 0.83 resolution under every estimate.
7. *a small per-head budget is what makes an allocation worthless* — quadrupling the median per-head budget from 543 to 2144 tokens leaves the allocation at +0.26 [-0.96, +1.48] over 747 paired examples.
8. *how much budget an allocation change moves tells you whether a benchmark will see it* — the resolved and unresolved sets overlap at 13.2%.
9. *calibrating on the corpus you serve buys a task-level gain, as its +9.55 pp objective advantage predicts* — the objective calls the book allocation -6.47% against an equal split and it scores +3.31 [+2.48, +4.13] over uniform; the corpus-matched arm is -2.53 [-3.41, -1.66] against it.
10. *the rank agreement of the two curves' marginal rates says what the wrong objective costs on a new workload* — orders the loss at +0.865 on 600 permutations and at -0.082 across the 30 real rows.
11. *restricting that agreement to a band around the greedy's stopping threshold rescues it* — orders the loss at +0.824 on 600 permutations and at +0.071 across the 30 real rows.
12. *weighting each misordered pair by how much marginal rate it misorders rescues it* — orders the loss at +0.876 on 600 permutations and at -0.321 across the 30 real rows, with the wrong sign.
13. *comparing the two greedies' bought SETS rescues it* — orders the loss at +0.879 on 600 permutations and at -0.047 across the 30 real rows.
14. *measure the rank agreement and spend the effort where it is low* — -0.600 over 5 captures spanning 0.027 of rank agreement, against +0.067 over the 29 rows those captures average.

What this is not. It is not a claim that nothing in this space is predictable, and not a licence to stop trying: three of the objective-side registrations in §D.1 fired exactly as written. It is a claim about one specific move — pricing a design choice's task-level worth from a quantity measured on the objective — which this paper made fourteen times in as many forms as it could think of, and lost every time.

E THE ESTIMATOR

E.1 ESTIMATOR COST

Every timing here is for a 512-*query* calibration — 288 cells, 512 queries, four budget levels — which is the window §6.1 fits the estimator on. It is *not* the calibration that produced the allocations the task arms ran: those are fitted on the whole 2048-query decode, and we have not timed that. Scale the numbers below with care, and read them as the cost of the estimator rather than of the shipped pipeline. Forming the contribution curves takes 2.9 ms in the median over 21 repeats, against 19.5 ms to form the per-cell standalone CVaR curves that the separable objective itself requires. The asymmetry is structural rather than incidental: a covariance against the aggregate is one pass of sums over the window, while a per-cell tail statistic needs the order statistics of that cell's own error at every level. Convexifying costs 1.3 ms and the greedy 5.5 ms, so one solve is under 10 ms and the full alternating loop of §5.1 — six rounds of estimate-and-re-solve — is 109 ms. Peak memory for the calibration array is 4.7 MB at 512 queries and 18.9 MB at 2048, the second by arithmetic rather than measurement. All of it is offline and once per (model, policy) pair: nothing runs on the serving path, which reads the allocation as a fixed table. What we do not measure anywhere is the cost of moving a per-head budget table into a production kernel; this paper prices computing an allocation, not adopting one.

Whether 109 ms is cheap depends on a denominator, so we measured one on the same card, the same model and the same dtype: one prefill of Qwen3-8B in bfloat16 at a 32768-token context on

an NVIDIA A800-SXM4-80GB takes a median 4075 ms over five repeats and peaks at 29.29 GiB. The entire offline calibration at that window therefore costs about 2.7% of a single prefill of the model it is calibrating — once, per (model, policy) pair, off the serving path. The same figure for the 2048-query calibration the shipped allocations use is not measured and we do not extrapolate it: the array is four times larger and the tail statistics do not scale linearly in the query axis. Data: `data/cost_s0001.json`, `data/prefillcost_s0001.json`.

E.2 Transfer to a shorter context

The task-level allocation of §6.6 is calibrated at a 32768-token context and applied unchanged at 4096. The press floors each head at `window_size` tokens; at 4096 the intended shares are eight times smaller, so clipping could have flattened the allocation toward uniform and a weak result could then have been blamed on the mechanism rather than on the method. We checked it with the press's own apportionment before any score was read. No head reaches the floor at either context — 0 of 288, both times — and rounding the shares to whole tokens costs the same at both, 4450 tokens of total deviation from the intended split at 32768 against 4456 at 4096. The one change is that the allocation is less differentiated at the shorter context: median head spread $4.19\times$ against $5.71\times$, from integer rounding on smaller shares. Clipping is therefore not available as an explanation for the result in either direction.

F Task-level evidence

F.1 Task tables

Task arms. Every `SepAlloc` vector this paper scores on a task metric — both tables of §6.6, the axis arms and the published-allocation comparisons — is built by one script whose per-cell curves are that cell's own CVaR_α: the *separable* objective of (1), the quantity the rest of this paper argues against. So $+4.21$, $+2.14$, the iso-quality band and the margin over Ada-KV all price an envelope greedy on standalone risk against a uniform split: a real result, not this paper's claim, which reaches a task metric only in the A/B — where it clears, $+2.59$ [+1.78, +3.40] over questions and [+0.72, +4.73] over task types against the same uniform budget, on a vector fitted on a held-out calibration window.

The two benchmarks do differ sharply in one measurable way — RULER's metric is nearly binary where LongBench's is graded — but that obvious explanation does not survive the test that would confirm it, so we report the shape as a property of the two benchmarks and not as the reason for the split, which we do not have.

Mistral-7B-Instruct-v0.3, RULER at 4096, is a third model family and is reported in §6.6 rather than in Table 1, because its sample is smaller than the other rows and putting it beside them would imply a comparability it does not have. The allocation is built from its own capture by the same recipe and applied unchanged; the pair is matched as every other pair here is. It scores 13.53 under uniform against 14.76 under the allocation, a paired $+1.23$ at $n = 650$ with none dropped and no refusals, 8 types up, 4 flat and 1 down. The item-level interval is [+0.09, +2.41] and the type-level interval is [-0.35, +3.00], which does not exclude zero, with 75% of it in one task type: `niah_single_1` alone is that much of the mean and the mean without it is $+0.33$.

The sample is 650 rather than the 1300 of the other RULER rows because the first attempt was lost to the harness fault described below and there was no time to repeat it at the larger fraction. Both arms share the same draw, so the smaller sample costs interval width and nothing else — but it is why this row is the weakest of the three, and we would rather say that than let a reader conclude the model is less amenable to the method.

Harness change. Every Mistral run initially returned a scorer failure. The harness's Mistral attention class passed `cu_seqlens_k` to the varlen flash-attention call without the `.to(torch.int32)` cast its three sibling classes apply — the two Qwen classes cast at assignment and the Llama class casts at the call site, and the Mistral class did neither. We added that cast, by line index and with an assertion that nothing else moved; the diff is a single changed line.

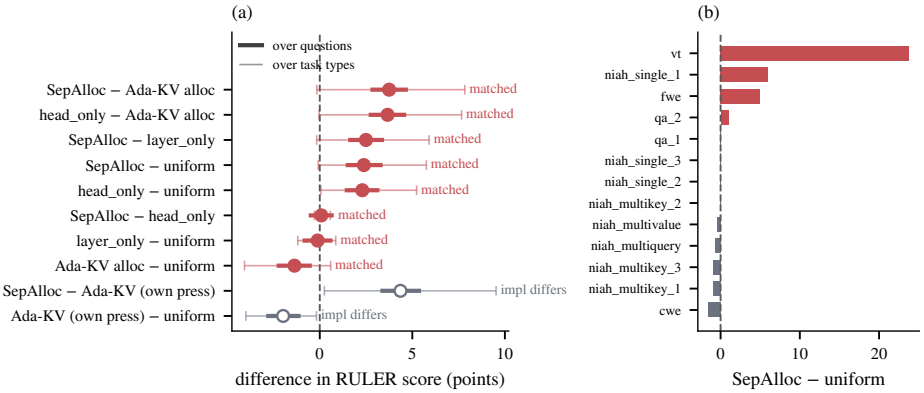

Figure 4: **RULER at** 32768**, the native context, with both intervals drawn.** Qwen3-8B, 1300 paired questions, four arms. **(a)** Every pairwise difference in mean RULER score (horizontal axis, points) with two 90% paired-bootstrap intervals: the thick inner bar resamples questions, the thin capped bar resamples RULER's thirteen task types. Those answer different questions, and the type-level bar is the one that generalises — at this context it crosses zero for the matched pairs. Filled markers are matched by construction: same script, press class, kernel and scorer, differing only in the per-head budget vector, so the difference is allocation. Hollow markers involve the competitor's own press, which re-derives a budget per input and so differs in more than implementation. **(b)** Per-type difference for `SepAlloc` minus `uniform`; 77% of the mean is one type, `vt`. The transferred setting at 4096 is summarised in Table 1 and broken out per type in Table 2.

The Qwen and Llama code paths are untouched, and no other number in this paper came through patched code.

Scoring is the harness's own rule reimplemented per sample, because a paired interval needs per-sample scores and the harness reports only aggregates. The reimplementation is checked against those aggregates and agrees on all 13 subtasks for every arm at both contexts. Only questions every arm answered are scored; none were dropped, so $n = 1300$ of 1300 in both tables. The transferred row is `tasktab_ruler4096_frac020.json` and the native row is `tasktab_ruler32768_frac020.json`, both under `data`.

Table 2 is the column a reviewer would go looking for, so we print it. It is the per-type difference behind both rows of Table 1, and it is where the two settings differ in character: at 4096 seven types gain and the two that lose give up a point each, while at 32768 five types lose and one type, `vt`, supplies 77% of the mean.

Table 2: `SepAlloc` minus `uniform`, in RULER points, per task type. Same arms and same 1300 questions as Table 1; the two columns are the two contexts. Positive is better.

Task type	4096, transferred	32768, native
`cwe`	+20.70	−1.60
`fwe`	+14.00	+5.00
`niah_single_1`	+11.00	+6.00
`vt`	+7.80	+23.80
`niah_multikey_1`	+2.00	−1.00
`niah_multivalue`	+0.75	−0.50
`niah_multiquery`	+0.50	−0.75
`niah_multikey_2`	0.00	0.00
`niah_multikey_3`	0.00	−1.00
`niah_single_2`	0.00	0.00
`niah_single_3`	0.00	0.00
`qa_1`	−1.00	0.00
`qa_2`	−1.00	+1.00
mean over questions	+4.21	+2.38

Why the allocation moves RULER and not LongBench we have not measured. The obvious explanation for the split is that a tail objective prices the worst 1% of per-query attention error, so what it reduces is the frequency of a query going wrong because one head lost the token it needed — and a graded metric turns that lost token into a slightly worse paraphrase, averaging it into a middling score, so the tail event never becomes a wrong answer. If that were the mechanism it would hold *inside* a benchmark as well as across two: the graded task types would be the ones the allocation moves least, and the rank correlation between a task's graded mass and its matched gain would be negative. It is not negative anywhere.

Benchmark	Model	tasks	graded, gain	base, gain	graded, base
RULER-4096	Qwen3-8B	13	$+0.423$	$+0.247$	$+0.516$
RULER-4096	Llama-3.1-8B	13	$+0.401$	$+0.242$	$+0.214$
RULER-32768	Qwen3-8B	13	$+0.368$	$+0.533$	$+0.445$
LongBench	Qwen3-8B	16	$+0.241$	-0.256	$+0.188$
LongBench	Llama-3.1-8B	16	$+0.041$	$+0.244$	$+0.012$

The correlation is positive in *all five* settings, and over the 71 pooled task rows it is $+0.128$ rather than negative. Within RULER the gains fall on the more graded types rather than the binary needle tasks, which is the opposite of what the mechanism predicts. A second reading closes the escape: graded mass is itself correlated with a task's baseline score at $+0.492$ pooled, so *graded* and *has room to move* are not separable here and neither can be credited.

So the shape difference between the benchmarks is real and is reported as a property of them. It is not the reason the allocation moves one and not the other, and we do not have that reason. The prediction was written down before the test was run, which is why this is a refusal and not a search that came up empty.

Both candidate explanations now stand in the same position. The metric-shape one is tested above and refused; the other is unmeasured and stated as a question — does RULER's synthetic retrieval concentrate the answer in tokens one head must retain, where LongBench's question answering and summarisation spread it? Nothing here answers either. The score mass itself is Table 3.

Table 3: Where each benchmark's per-example score mass sits, on the uniform control arm over the paired examples. A metric that is nearly binary can register a lost token as a wrong answer; a graded one averages it into a partial score. The body quotes the last column.

Setting	Model	at zero	at full	strictly between
RULER 4096, transferred	Qwen3-8B	77.7%	8.2%	14.1%
RULER 4096, transferred	Llama-3.1-8B	76.7%	11.4%	11.9%
RULER 32768, native	Qwen3-8B	75.3%	13.3%	11.4%
LongBench	Qwen3-8B	34.5%	15.2%	50.3%
LongBench	Llama-3.1-8B	33.8%	14.6%	51.6%

Mean scores behind the rows, which the body gives as differences. At 4096 RULER goes from 14.59 to 18.81 on Qwen3-8B and 16.14 to 17.57 on Llama-3.1-8B. At 32768 RULER goes from 18.84 under uniform to 21.22 under the allocation; on LongBench it goes from 34.15 to 34.11 on Qwen3-8B and 34.32 to 34.67 on Llama-3.1-8B, which is the sense in which LongBench scores roughly twice what RULER does.

Ada-KV appears twice in these tables: once as its measured allocation alone, driven through our press, and once as its own published implementation, which re-derives a per-head budget from every input. We label which is which rather than pooling them, having had to withdraw an earlier reading that confused the two.

Every matched pair at 32768 behaves the way ours does: clearing zero over items and not over task types, the intervals being $[-0.08, +5.76]$ for our gain over uniform, $[-4.07, +0.59]$ for Ada-KV's loss and $[-0.16, +7.83]$ for our gain over Ada-KV. Ada-KV's shipped press separately loses 1.98 points, but it re-derives a budget from each input's attention weights, so it differs from the matched arm in implementation, in per-input adaptation and in whether the allocation was transferred at all; no subtraction separates three confounded differences, and we do not attempt one. At 4096, where

no matched Ada-KV arm was run, the shipped press is not distinguishable from uniform (-0.29, $[-0.98, +0.36]$) — a statement about the pair of systems and not about its allocation.

How much of each mean is one task is worth stating with it. At 4096 the largest single type is `cwe` at 38% of the gain on Qwen3-8B, with the mean without it $+2.84$, and `niah_multikey_1` at 27% on Llama-3.1-8B, with the mean without it $+1.13$; at 32768 it is `vt` at 77% and the mean without it is $+0.60$. On LongBench a share of the net movement is not a meaningful quantity, because the net movement is null; the per-task differences run from -4.10 to $+2.50$ on Qwen3-8B and -7.45 to $+7.50$ on Llama-3.1-8B.

The LongBench length split, in full. The allocation is calibrated at 32768 tokens and LongBench is far shorter — median example 5422 tokens on Qwen3-8B and 5438 on Llama-3.1-8B — so each model's table is split at its own median length. All four cells are printed because reporting only some of them would be selecting on the outcome: Qwen3-8B gives $+0.52$ $[-0.89, +1.92]$ on the short half and -0.60 $[-2.25, +1.04]$ on the long, Llama-3.1-8B $+1.39$ $[+0.18, +2.69]$ short and -0.69 $[-2.72, +1.28]$ long. Three of the four are null; the fourth, Llama's short half, is one cell of four at 90% and both long halves are negative, so we read the four together and not the one that cleared zero.

Refusals, in full. The press raises rather than running a layer whose budget after compression is below the observation window it must keep, and the harness would otherwise write the exception into the answer field where it scores as a confident wrong answer. On LongBench the allocated arm refuses 6 times against uniform's 1 on Qwen3-8B and 14 against 1 on Llama-3.1-8B, so scoring refusals as answers would charge the allocated arm for a failure mode it causes and then penalise the score as well. They are nulls, dropped from both arms, which is why 743 of 750 and 734 of 750 examples are paired. Every RULER arm refuses zero times.

The cluster bootstrap of §6.6 resamples these thirteen rows with replacement, 5000 times, and takes the 5th and 95th percentiles of the resampled mean. It is wider than the question-level interval on both rows because thirteen clusters is a small sample, which is the honest cost of asking the question a reader actually asks. Superseded by these two tables, and recorded because it was quoted in an earlier draft: an initial 32768 run at a quarter of this sample gave $+1.49$ $[-0.34, +3.29]$ over questions (`tasktab_ruler32768_frac005.json`).

Another budget. The numbers of §6.6 sit at one compression ratio, so the arms were rerun at four per-head budgets, the allocation rebuilt at each (Table 4). It survives doubling and is gone at halving, where the interval covers zero. Over the band where it clears, the allocated arm reaches the uniform arm's score at 1755 tokens per head against 2048 and 3188 against 4096 — at least 14% less cache, on the more conservative of two readings of an interpolation that is not itself a measurement. The 1024 row is part of the finding: that allocation is not smaller but *censored*, 53.5% of the 288 cells pinned to the ladder's bottom rung against 2.8% at 4096, and relieving it does not help — adding a rung at 256 halves the pinned fraction to 27.8% and makes the allocation *worse* on the held-out window, $+3.72\%$ of objective falling to $+2.14\%$. The grid was acting as a capacity control, and the null at 1024 is not an artefact of it.

Table 4: The same arms at four operating budgets. RULER 32768, Qwen3-8B, 1300 paired questions, 90% bootstrap over questions. One script, press class, kernel, scorer and total per row; only the per-head budget vector differs, and it is rebuilt at each budget. The 2048 row is Table 1's 32768 row. The 3072 midpoint was run after the interpolation below was registered, and it lands *above* both readings of it.

per head	ratio	uniform	`SepAlloc`	Δ, 90% over questions
1024	0.96875	13.57	12.91	-0.67 [-1.47, +0.17]
2048	0.9375	18.84	21.22	$+2.38$ [+1.40, +3.40]
3072	0.90625	21.97	24.95	$+2.98$ [+1.93, +4.04]
4096	0.875	25.19	27.03	$+1.84$ [+0.91, +2.78]

The midpoint. The 1755-token and 3188-token crossings of §6.6 were read off two endpoints, and the two registered readings of that interpolation — linear in b_0 and linear in $\log b_0$ — put the allocated arm at 24.1 and 24.6 RULER points at 3072. It measures 24.95, above both: the two-endpoint reading was conservative rather than optimistic, which is the direction that costs this paper nothing to

have been wrong in. With the midpoint in the curve the saving at the 4096 reference moves from 15.9% to 22.2% of KV cache at equal score, and the 3072 reference gives 26.7%. At Qwen3-8B's geometry — 36 layers, 8 KV heads, 128 per head, two bytes, so 147456 bytes a retained token — the 4096 reference saves 8.0 GiB at a batch of 64. **The claim the paper makes does not move**: 14% is set by the 2048 reference, which the midpoint does not touch, and it stays the quoted figure. Both readings in full: the allocated arm reaches the uniform arm's score at 1755 tokens per head against 2048 and 3188 against 4096, and read as linear in $\log b_0$ the crossings are 1680 and 3174, the two readings differing by up to 3.7 points of saving (14.3% and 22.2% against 18.0% and 22.5%). We quote the smaller.

Censoring. Of the 288 cells, 53.5% sit on the ladder's bottom rung of 512 tokens at $b_0 = 1024$ and 0.3% on its top, against 17.4% and 1.7% at 2048 and 2.8% and 17.4% at 4096, while the mass moved away from uniform is nearly the same at all three (26.7%, 26.9% and 24.0% of the movable maximum). So the 1024 vector is not a smaller version of the others; it is the same amount of movement pressed against a boundary. The obvious reading is that a ladder stopping at 512 cannot decide a problem this tight, and that extending it downward would let the greedy choose. **That arm has now run, and it refutes the reading.** On eight rungs with a floor of 256 the pinned fraction falls from 53.5% to 27.8% — half the censoring relieved — and the objective on the held-out window falls with it, from +3.72% to +2.14%. Given more freedom to starve cheap cells the greedy takes it, fits the calibration window better and generalises worse: the grid was acting as a capacity control. The registered thresholds were PASS at objective $\geq +5.00\%$ with floor below 35% and REFUTED at a gain moving less than 0.5 pp with the floor still above 45%; it hit neither, so it is reported as ambiguous with the direction named, and no task arm was run on it. What it does settle is narrower and it is the thing the 1024 row needed: the null there is not an artefact of the grid. Data: `ladder256.json`. `censor.py` recomputes every fraction here from the three shipped allocation vectors in `data`, refusing if a file's mean is not the budget it is registered at or if any cell is off its own ladder.

F.2 THE A/B

Pre-registered arithmetic. The arms are far enough apart for the comparison to matter: they agree on 64.2% of cells and 10.4% of the budget separates them. The two allocations' objective gap is 2.19 pp, priced across the transmission band at 0.15 to 0.65 points against a half-width of 0.68, needing 5777 questions — a pricing the same band has since missed by 200× on a *smaller* gap, so it is reported as what the arithmetic said and not as a reason the effect is small (§F.4); nor does moving the operating point help, as the rest of this section shows.

Two allocations, one capture (`s0001`), one ladder, one per-cell normaliser, one lower convex envelope, one steepest-descent greedy, one 512-query calibration window and one total. They differ in what the per-cell curves measure — the Euler contribution of (4) against the standalone $\mathrm{CVaR}_{0.95}$ of (1) — and in nothing else, because both come from the same script with one flag moved. The separable arm is the control rather than the shipped `mkalloc.py` vector precisely because the shipped vector is fitted on every query while the contribution estimator is fitted on the calibration window alone; comparing against it would confound the quantity with the fitting window.

The two arms are far enough apart for the comparison to mean something: they agree on 64.2% of cells, and 10.4% of the budget has to move to turn one into the other, 42% as far as the allocation itself moves from uniform.

Moving the pair does not rescue it either: over 33 configurations the gap is positive in 29, median +1.31 pp, and the best anywhere predicts 0.20 to 0.88 points — all of it under what this benchmark resolves, which is why §6.6 reports the RULER comparison as unresolved rather than as a null.

Registered before either arm was scored (`data/abdist.json`): the arms share 64.23% of their 288 cells, 10.42% of the budget has to move to turn one into the other, which is 41.96% of how far the shipped allocation moves from uniform, and on the held-out window the contribution arm scores +11.88% against the separable arm's +9.70%, a gap of 2.19 pp. The script refuses outright above 95% identical cells, because at that point the experiment measures the grid. Also registered: that a difference smaller than the instrument's resolution is to be reported as *did not resolve* and never as equivalence.

Arm	mean	90% over questions	types ↑ /=/ ↓
uniform control	14.59	—	—
separable curves	17.04	+2.45 [+1.63, +3.29]	7/5/1
contribution curves	17.19	+2.59 [+1.78, +3.40]	10/3/0
contribution − separable		+0.15 [-0.52, +0.83]	5/5/3

RULER at 4096 on Qwen3-8B, 1300 paired questions, none dropped, both arms matched by construction. The type-level intervals are [+0.63, +4.56] and [+0.72, +4.73] for the separable and contribution arms against uniform, and [-0.62, +0.96] for the difference between them. Data: `tasktab_ab095_ruler4096.json`.

Power. Across the transmission band of §F.4, 0.067 to 0.298 task points per objective percent, a 2.19 pp objective gap is 0.32 to 0.65 task points. The difference interval's half-width at $n = 1300$ is 0.68, so the whole band sits under it: at the lowest rate excluding the effect needs about 5777 paired questions, 4.4× the compute of this pair, and at the highest the predicted effect is still smaller than what the interval can see. `gapsearch.py` then builds both allocations at 33 configurations — four captures, a four-rung and a seven-rung ladder, $\alpha \in \{0.90, 0.95, 0.99\}$ and $b_0 \in \{1024, 2048, 4096\}$ — and the objective gap is positive in 29 of them, median +1.31 pp, largest +2.95. The A/B already ran at 2.19, near the top of that distribution, and the largest configuration anywhere predicts 0.43 to 0.88 task points against the same 0.68. Only at the top of the band does the best setting in the whole surface exceed the half-width, and then by 0.20 points; under the other two estimates there is no setting where this comparison becomes resolvable at $n = 1300$. Either way the null cannot be escaped by moving the experiment to a configuration this paper already has. Data: `gapsearch.json`, `resolution.json`.

F.3 Scoring and fitting at different tail levels

The result. Scoring at 0.99 and fitting shallower instead, with the level chosen leave-one-capture-out over five captures and two model families, is worth a paired median of +0.25 pp on 3 of the 5 folds — mean +1.73, over folds running from −0.63 to +5.42, so five captures fix the dispersion and not the size, and the difference of the two columns' medians is +1.57 pp and is not the improvement. At $\alpha = 0.95$ the same procedure gives +2.12 against +2.12 and we refuse it, which is what the mechanism predicts: that window leaves twenty-six tail points and the estimator is not starving. No arm in this paper runs it.

The recipe of §6.4 estimates the Euler contribution at the same α it optimises. At $\alpha = 0.99$ that means fitting on the five tail points of §5. Decoupling the two — score at 0.99, fit at a shallower level, everything else identical — is one argument to the same estimator, and the objective-side gap over the exactly-solved separable optimum moves as follows (median over five captures on two model families, $b_0 = 2048$, four-rung ladder):

fitting α	0.80	0.90	0.95	0.99
median gap, scoring at 0.99	+4.01	+4.12	+3.70	+2.12

The last column is the shipped coupling. Choosing the fitting level is itself a choice, so it is made leave-one-capture-out: the level is chosen on four captures and the gap is read on the fifth.

held out	fit picked	gap	shipped	Δ
`s0001`	0.95	−2.02	−2.27	+0.25
`s0002`	0.95	+2.93	+2.95	−0.01
`s0003`	0.95	+3.70	+0.10	+3.60
`s0004`	0.80	+7.55	+2.12	+5.42
`mistral_30720`	0.80	+4.42	+5.06	−0.63
median		+3.70	+2.12	+0.25

The column that carries the result is the last one, and it was not the one we first quoted. The design is paired — each fold's chosen and shipped gaps are read on the same held-out capture —

so what it supports is a paired median of $+0.25$ pp on 3 of the 5 folds, with a mean of $+1.73$, over folds running from -0.63 to $+5.42$. The difference of the two columns' medians is $+1.57$ pp, and the draft quoted that as the improvement: it is a different quantity whenever the fold sitting at one column's median is not the fold at the other's, and here `s0003` sits at one and `s0004` at the other. What five folds establish is therefore the *dispersion* and not the size — the same change is worth nothing on two captures and five points on a third — and saying so is a stronger statement than a point estimate the design does not carry. No fold ever picks 0.99. The fold where shallower fitting *hurt* is `mistral_30720`, the only one from a second model family, and it stays in rather than being set aside — the result is worth having only if it survives the capture that argues against it.

It does not hold at $\alpha = 0.95$, and we refuse it there. The same procedure gives $+2.12$ chosen against $+2.12$ shipped, and a paired median of exactly 0.00 on 2 of 5: nothing under either reading, so the in-sample gain at that level is a selection artefact and is not quoted. The mechanism predicts this asymmetry: at 0.95 the same 512-query window leaves twenty-six tail points rather than five, and an estimator that is not starving has nothing to gain from being handed a shallower level. An improvement that appeared at both levels would have been more suspicious than one that appears only where the sample runs out. Neither result is in any arm this paper ran; both were found after the arms ran. Data: `alphafit5_a0.99.json`, `alphafit5_a0.95.json`.

F.4 The transmission rate

§6.6 prices design choices by multiplying an objective-side difference by a rate. There are four defensible estimates of that rate and they span a factor of three, so the section carries the band rather than a point.

Fit. It fits our own four allocations at $R^2 = 0.98$ and all eight matched pairs at 0.44; the sign record on the rows carrying a measured task effect is below. The ± 0.83 half-width §6.6 prices against is a pooled figure over two regimes, 0.809 at context 4096 and 0.950 at 32768.

Priced choices. Allocating at all is worth $+6.35\%$ to $+16.03\%$ of objective at the five budgets of Table 4, priced between $+0.42$ and $+4.77$ points, and four of five clear under every estimate. All five have since been measured, and two came out with the opposite sign to their prediction: $b_0 = 1024$, priced at $+0.42$ to $+1.89$ and measured -0.67, and $b_0 = 6144$, priced at $+0.90$ to $+4.03$ and measured -0.53 — the second registered before it ran, and its rung is half of a pair 0.14 pp apart on the objective and $+3.42$ [+2.15, +4.71] apart on the task (§F.5). Both are struck. A finer choice clears under the highest estimate only — the corpus of §6.5, $+9.55\%$ and $+0.64$ to $+2.84$ predicted points — and it is one of four rows measured against a prediction and come out with the opposite sign, so it is struck rather than counted. Of the eight others, four clear only under the highest estimate and four never clear.

The first of them used to be conditioned on the outcome. It averaged the rate over the arms whose task effect cleared zero, and that rate is then used to decide which design choices this benchmark can see at all. Estimating the visibility constant from the choices that turned out to be visible is a selection on the dependent variable, and it can only push the rate up. The unconditional estimate — every row carrying a measured task effect, no condition on what that effect was — now joins the band as its lowest member, and under it $b_0 = 1024$ stops clearing.

When that estimate was added this was recorded as a coherence fix — the row the model got the sign wrong on had stopped being a row the table called visible. The sixth measured row withdrew that. $b_0 = 6144$ clears under every estimate *and* is measured with the opposite sign, so the table does again exactly what widening the band was said to have stopped it doing. The arithmetic and the measurement are not reconciled here; they disagree, and the disagreement is the result.

estimate	fitted on	pairs	points per objective %
all measured	every row with a measured effect	5	0.067
arms that clear	the arms that clear zero	3	0.159
all pairs	through-origin least squares	8	0.203
our own	the same, measured objective only	4	0.298

The eight pairs are every matched comparison at RULER 32768 and the 4096 A/B for which both an objective-side and a task-side difference exist. The four are those whose objective column is measured on the arm's own vector rather than interpolated from a published one.

Every choice this method offers, priced under both ends of the band. Predicted task points are the objective-side difference times the rate; *clears* says under how much of the band the prediction exceeds the ± 0.83 resolution of §6.6. It is a property of the prediction, so a row can clear and still be wrong: *struck* marks a row whose task effect has since been measured with the opposite sign, which is not a choice worth making whatever the arithmetic says. Two rows are struck and they are struck for different reasons: at $b_0 = 1024$ the allocation being priced is the censored one (§F.1), and on the corpus row nothing is wrong with either allocation (§6.5).

design choice	objective	predicted	clears
allocate at all vs uniform, $b_0 = 3072$	$+16.03\%$	$+1.07$ to $+4.77$	always
allocate at all vs uniform, $b_0 = 4096$	$+15.99\%$	$+1.07$ to $+4.76$	always
allocate at all vs uniform, $b_0 = 6144$	$+13.56\%$	$+0.90$ to $+4.03$	struck
allocate at all vs uniform, $b_0 = 2048$	$+13.42\%$	$+0.90$ to $+3.99$	always
calibrate on your workload, book to RULER	$+9.55\%$	$+0.64$ to $+2.84$	struck
allocate at all vs uniform, $b_0 = 1024$	$+6.35\%$	$+0.42$ to $+1.89$	struck
calibrate on your corpus, book to book	$+4.51\%$	$+0.30$ to $+1.34$	highest only
contribution vs separable, best of 33	$+2.95\%$	$+0.20$ to $+0.88$	highest only
seven ladder rungs instead of four	$+1.46\%$	$+0.10$ to $+0.43$	struck
contribution vs separable, median of 33	$+1.31\%$	$+0.09$ to $+0.39$	never
polish the allocation by local search	$+0.42\%$	$+0.03$ to $+0.13$	never
fit the contribution shallower at 0.99	$+0.25\%$	$+0.02$ to $+0.08$	never

The ladder row. `runs/sevenrung.md` was committed before the three extra rungs were harvested. Its objective-side prediction held: the seven-rung RULER-calibrated allocation scores $+5.20\%$ against an equal split on its own held-out window where the four-rung one scores $+3.08\%$, so the extra rungs give the envelope greedy more places to put a token and it uses them. Its three task-side branches — the ladder is the whole gap ($\approx +4.21$), the capture advantage persists ($\approx +1.68$), or the transmission band ($+1.01$ to $+1.21$) — were all missed, in the same direction:

arm	score	Δ vs uniform	90% over questions
uniform control	14.59	—	
RULER-calibrated, four rungs	15.37	$+0.78$	[+0.17, +1.42]
RULER-calibrated, seven rungs	14.70	$+0.11$	[-0.52, +0.71]
seven rungs − four rungs		$\mathbf{-0.67}$	[-1.21, -0.14]

Three extra rungs buy 2.12 pp of objective and cost 0.67 task points, with the question-level interval excluding zero. **The ladder is not the book arm's advantage; on this capture it is a cost.** Together with `runs/bookarm.md`, which showed the gap is not the capture in the direction anyone expected, both halves of the 3.43-point gap between the shipped seven-rung book allocation and the four-rung RULER-calibrated one are now accounted for and neither is what it looked like. It is also the fourth row in this table measured against a prediction and returned with the opposite sign, and the only one that carried three specific numeric branches and missed all three. The four-rung arm rescored $+0.78$ [+0.17, +1.42] beside it as a duplicate control, on its own arm set so the published table's pairing mask did not move. What this does *not* show is anything about seven rungs on the book capture, where the shipped allocation lives and scores $+11.88\%$ on its own window; only `rulercal`'s ladder changed. Data: `tasktab_indist7.json`.

Three rows moved after the table was first built. Two moved *down*: the shallower-fitting row used to read $+2.35\%$, the difference of two column medians on a paired design (§F.3), and the paired median those folds support is $+0.25\%$; the local search is this paper's own newest result (§G.3) and is priced by the same line as everything else, because a method that exempted its own additions from the arithmetic it applies to the field would not be making the argument it thinks it is. The third is new and is the only finer choice that clears: the two calibration rows are the same design choice at two distances, one novel to another and a novel to the evaluation distribution (§6.5), and the effect roughly doubles with the distance.

The fit is good inside the family and bad outside it. Over the four it reaches $R^2 = 0.98$; over all eight, 0.44. The pairs that break it are the published allocations, which is §6.2's disagreement seen on one scale: layer-DefensiveKV's own vector is -4.31% of objective and $+2.07$ points of RULER score. Within the family this objective was designed for, the objective tracks the score; on a vector built to optimise something else, it does not.

One out-of-sample row, and what it does and does not settle. The band was fitted on matched pairs at context 32768 from book captures, and until tonight its only check was the three rows it is displayed against — rows it was partly fitted on. The corpus arm of §6.5 is a row it could not have seen: a different capture, context 4096 rather than 32768, 256 tokens a head rather than 2048, with both sides measured. Its $+3.08\%$ of objective prices at 0.45 to 0.92 points and it measures $+0.78$, an implied rate of 0.252: inside the band. That removes one objection and no more. It does not make the model a predictor of individual outcomes, the sign error at the smallest budget stands, and the model is still used only to place effects relative to a line. `xmitcheck.py` refuses rather than reports if that row lands outside, because the resolution analysis is built on this model. **And it is not the only out-of-sample row any more.** The same capture supplied a second one hours later — the book-calibrated arm against the RULER-calibrated one, priced at $+1.40$ to $+2.84$ points and measured at -2.53 — which is outside the band and on the other side of zero. One out-of-sample row landed inside and one landed opposite, on the same capture and the same benchmark, which is a fair summary of what this model is worth. Data: `xmitcheck.json`, `tasktab_indist.json`.

Two reasons not to read $R^2 = 0.98$ as predictive power. Three of those four pairs share the same uniform control, so the four points are not independent and the figure says they lie on a line, not that the line locates a fifth. And the rate gets the *sign* wrong on three of the six rows where a task effect was actually measured: at $b_0 = 1024$ it predicts $+0.42$ to $+1.89$ and the measurement is -0.67; on the corpus row of §6.5 it predicts $+1.40$ to $+2.84$ and the measurement is -2.53; and on the ladder row it predicts $+0.10$ to $+0.43$ and the measurement is -0.67. **The four failures are not the same failure.** The 1024 row is the censored allocation of §F.1, whose objective gain was bought by a vector the grid deformed, so the input to the model was already suspect. The 6144 row sits one rung below the ladder's top, where the allocator has less room to move budget — but the objective column already absorbs that, at $+13.56\%$ against $+16.03\%$ at the peak, and the task delta falls further than the objective did. The corpus row has no such excuse at all: both allocations are interior, both are scored on the same corpus at the same budget, and the objective simply ranks them the wrong way round. The ladder row has the least room of the four: its objective-side prediction was registered and *held*, seven rungs scoring $+5.20\%$ against four rungs' $+3.08\%$, and its three task-side branches were committed before the arm ran and missed in the same direction — the extra rungs buy objective and cost task points. So the rate is used to place effects relative to a line, never as a predictor of an individual outcome, and every conclusion drawn from it rests on factors of two and five. Data: `transmit.json`, `resolution.json`.

F.5 Two allocations with the same objective

§F.4 prices an objective-side difference by multiplying it by a rate, which treats the objective as a *scale*: more of it buys proportionally more task metric. A stronger property is at stake in this paper's argument and can be tested on its own. Is the objective a *sufficient statistic* for what an allocation is worth — so that two allocations scoring the same on it are worth the same on the task, whatever else differs between them?

The budget ladder of Table 4 answers it twice, and neither pair was chosen: the ladder produced them. Each rung's shipped vector, scored on its own capture's test window by the procedure of §F.4:

per head	ratio	objective (in-sample)	task Δ
1024	0.96875	$+6.35\%$	-0.67
2048	0.9375	$+13.42\%$	$+2.38$
3072	0.90625	$+16.03\%$	$+2.98$
4096	0.875	$+15.99\%$	$+1.84$
6144	0.8125	$+13.56\%$	-0.53

A claim withdrawn here. An earlier form of this table carried three rungs — 1024, 2048, 4096 — and closed by observing that the objective column is monotone in the budget while the task column is

not. Those three rise. The five do not: the objective peaks at 3072, which is where the task column peaks too. Both are single-peaked at the same rung, so that reading is withdrawn. `objpred.py` recomputes the three published rungs by the same code path and refuses unless it reproduces them to 10^{-9}, so this is a longer table and not a rescaled one.

What the correction leaves is sharper than what it took. Two rows of this table carry the same objective and are worth three points apart on the task. $b_0 = 2048$ and $b_0 = 6144$ carry objectives $+13.42\%$ and $+13.56\%$, 0.14 pp apart, at three times the operating budget, and deliver $+2.38$ and -0.53. **That rung was registered before it ran** — its target was $b_0 = 2048$'s measured interval, $[+1.40, +3.40]$, and the decision rule, the three competing predictions and the ladder-ceiling alternative were committed with it.

The difference of differences. Every rung ran the same 1300 questions in the same order, so each contributes a per-question difference $d_r(q)$ between its allocated and its uniform arm and any two rungs can be differenced question by question. That is not $\mathrm{median}(d_r) - \mathrm{median}(d_s)$, and it is not two intervals inspected for overlap; both discard the pairing, and this project has already published a figure that came from differencing two columns' medians on a paired design (§F.3). All ten pairs are reported rather than the two the analysis was written for. 1300 of 1300 questions were answered by all ten arms.

rungs	Δ objective	Δ task	90% over questions	90% over task types
$1024 - 2048$	-7.07 pp	-3.05	[-4.28, -1.79]	[-6.13, -0.27]
$1024 - 3072$	-9.68 pp	-3.65	[-4.98, -2.35]	[-5.77, -1.76]
$1024 - 4096$	-9.64 pp	-2.51	[-3.81, -1.19]	[-4.94, -0.38]
$1024 - 6144$	-7.21 pp	-0.13	[-1.38, +1.11]	[-2.33, +2.29]
$2048 - 3072$	-2.61 pp	-0.60	[-1.83, +0.63]	[-2.45, +1.37]
$2048 - 4096$	-2.57 pp	$+0.54$	[-0.78, +1.83]	[-1.99, +3.47]
$\mathbf{2048 - 6144}$	$\mathbf{-0.14}$ pp	$\mathbf{+2.92}$	[+1.51, +4.30]	[-0.28, +6.72]
$\mathbf{3072 - 4096}$	$\mathbf{+0.04}$ pp	$\mathbf{+1.14}$	[+0.06, +2.23]	[-0.38, +2.98]
$3072 - 6144$	$+2.48$ pp	$+3.52$	[+2.18, +4.86]	[+0.95, +6.42]
$4096 - 6144$	$+2.43$ pp	$+2.38$	[+1.22, +3.48]	[+1.06, +3.81]

The matched-objective pair is $+2.92$ [+1.51, +4.30]. The bootstrap mass on the wrong side of zero is 0.000, so nothing here turns on where a quantile falls. The other matched pair, $3072 - 4096$, agrees in direction at $+1.14$ [+0.06, +2.23]; that one clears zero by 0.06 and was re-drawn under twenty independent bootstrap seeds, and twenty of twenty still exclude it. **At** 32768 **the task-type interval is [-0.28, +6.72] and does not exclude zero**, so on the population this paper elsewhere argues is the one that generalises, the difference was not established there. Both are reported and neither is chosen.

A second context length. `runs/iso6144ctx4096.md` was committed before these arms ran and registered three readings: replicates, does not, or inverts. The press takes `budgets` as a normalised weight vector, so the same two vectors deploy 768 and 256 tokens a head at 4096 rather than 6144 and 2048; the objective side does not move, being a property of the vectors and the calibration capture, and only the deployment context changes.

context	Δ task	90% over questions	90% over task types
32768	$+2.92$	[+1.51, +4.30]	[-0.28, +6.72]
4096	$\mathbf{+3.42}$	[+2.15, +4.71]	[+0.81, +6.26]

It replicates, and on the task-type population it does what the original could not: $[+0.81, +6.26]$ excludes zero, so at this context the pair clears on both populations, which is this paper's standard for a strong claim and which this pair had not previously met. Bootstrap mass at or below zero is 0.0000 over questions. Three things belong beside it. The difference is up on 7 of the 13 task types, so six move the other way, and the cluster interval is the bootstrap that prices exactly that. `cwe` carries 40% of the total at $+17.70$, with `qa_2` most negative at -4.00. And the registered uninformativeness condition — a large move in the 0.8125 control's zero-score fraction, 64.5% at 32768 — did not trip, at 66.5%; the 0.9375 control at this context scores zero on 77.9% of examples, a harder regime than any control the registration named, and it is reported here because it was not the arm the condition named.

Scored by `ladder4096_score.py`, which is `tasktab_cur.py` with the results directory made overridable and the per-question vectors written out. A copy of a scorer is a second scorer, so it was required to reproduce the 0.8125 rung's $+0.91$ [-0.03, +1.88] and its subtask means exactly before its 0.9375 number was used for anything; `ladder4096.py` refuses unless the task and answer columns are identical across all four frames. Data: `ladder4096.json`.

And the relation fails in the other direction too. $b_0 = 1024$ and $b_0 = 6144$ are 7.21 pp apart on the objective — half the range the whole ladder spans — and -0.13 [-1.38, +1.11] apart on the task, which is nothing. So the objective can move a great deal while the task does not, and stay still while the task moves three points. It is not a sufficient statistic for what an allocation is worth, and on this ladder it is not a monotone function of it either.

What is registered and what is not. The 2048–6144 pair is, twice: `runs/iso6144.md` was committed before its arms started and `runs/iso6144ctx4096.md` before the replication's, and the readings above are the ones they specify. The eight other pairs are not — every arm in them ran before the analysis was written, and `laddershape.json` records `registered: false` for that reason.

Why not $b_0 = 8192$. It is the ladder's top. Every cell would already sit at the maximum rung, so any move down must be paid for by a move up that no rung exists for, the only feasible vector there is the uniform one and the objective gap is identically 0.00%. `objpred.py` does not argue that: it runs the real allocator at 8192 and requires what comes back to BE uniform, with the rung below as a negative control so a broken allocator cannot pass as a real ceiling. An arm there would have spent six GPU-hours measuring a guaranteed null.

Data: `objpred.json`, `laddershape.json`, `objtask.json`.

F.6 THE LONGBENCH BUDGET

The per-head budget an arm receives is the context length times one minus the compression ratio, so an arm at a fixed ratio deploys whatever budget the input's length gives it. LongBench's contexts are short. Tokenised with Qwen3-8B's own tokenizer over the 743 paired examples, at the 0.9375 the paper ran:

per-head budget	p10	p25	p50	p75	p90
tokens	143	275	543	902	1112

82.6% of examples sit below $b_0 = 1024$ and 97.6% below 2048; only 18 of 743 reach 2048, too few to form an interval. Put the other way, the per-head budget those 743 examples deployed has median 543 tokens, 83% of them below 1024 and 98% below 2048, where Table 4 puts the allocation at -0.67. Budget alone does not explain it — RULER at 4096 deploys 256 tokens per head and gains $+4.21$ — but it limits what the null is evidence for. Binned against the boundaries the sweep of Table 4 measured rather than boundaries chosen here, the allocation is $+0.26$ [-0.92, +1.47] below 1024 ($n = 614$) and -1.69 [-4.78, +1.23] between 1024 and 2048 ($n = 111$), while Table 4 itself measures -0.67 [-1.47, +0.17] at $b_0 = 1024$. So the LongBench arms were run almost entirely inside the band where our own data says there is little to detect. The median length split already in §F.1 cannot see this: its long half has a median length of 9713 tokens, which at this ratio is a per-head budget near 600, so it compares two points inside the same band.

The budget hypothesis has now been tested directly, and it is refused. At compression 0.75 the median LongBench example gets 2144 tokens a head rather than 543 — four times as many, and the regime where the RULER sweep of Table 4 measures $+2.38$ [+1.40, +3.40] — and the allocation is worth $+0.26$ [-0.96, +1.48] over 747 paired examples, 5 task types up, 1 flat and 10 down. Nor is there a gradient inside that run: binned by deployed budget it is $+0.23$ below 1024, $+1.58$ between 1024 and 2048 and -0.30 above 2048, which is not monotone, and the three bands differ in task composition anyway. Three examples of 750 are dropped, one press refusal on the allocated arm and two nulls, removed from *both* arms by the pairing and counted rather than absorbed; the cruder whole-CSV failgate marked that run dead on the single refusal, which is what it is for, while the paired scorer handles refusals and reports them. So quadrupling the budget does not move the null,

and what is left unexplained about LongBench stays unexplained rather than being quietly attributed to the operating point.

This was never offered as the explanation of the LongBench null, and the paper's own strongest result already refused it as a general budget law: RULER at 4096 deploys 256 tokens per head — a quarter of what the 0.96875 arm deploys at 32768 — and gains +4.21. The press takes `budgets` as a normalised weight vector, so what an arm deploys is a *shape* and the absolute budget comes from the ratio and the input; the two are independent. What the budget distribution supports is narrower: the comparison was made where there is little to detect, which limits what the null is evidence for. The budgets above are tokenised rather than estimated — LongBench's own `length` field is counted with a different tokenizer and has median 5422 — and the block refuses if Qwen3-8B's tokenizer will not load rather than falling back to characters per token. Data: `lbtab_budget_0.9375.json`.

F.7 The LongBench null

LongBench is this paper's one flat benchmark and it survived every explanation offered for it. The budget was refused twice: by inference first — LongBench deploys a median 543 tokens a head at this ratio while RULER at 4096 deploys 256 and gains +4.21 — and then by experiment, at four times the budget, where the allocation is worth +0.26 [-0.96, +1.48] (§F.6). Context length does not explain it either.

The hypothesis, registered before the capture existed. Every LongBench arm in this paper runs an allocation calibrated on a nineteenth-century novel, and the transfer matrix of §E.2 says carrying an allocation to a different KIND of text is not merely weaker but worse than not allocating. LongBench is news, code, meeting transcripts, government reports and multi-document QA. That is at least as far from Moby-Dick as RULER is.

The capture. `rulercorpus.py` built it unchanged — it groups by task and slices context, and LongBench carries both — which is worth recording: the in-distribution machinery generalises to a second benchmark by argument alone. All 16 task types, one held-out row each, 32126 tokens inside a 32768 window, disjoint from the harness's evaluated subsample with the intersection asserted empty.

Objective-side gates. Both were registered and both are computed from the harvest, so the run could stop before spending a task arm.

• **L2 held.** Scored on this capture's own held-out window, the book-calibrated vector is −8.84% against an equal split — worse than not allocating, as on the RULER capture where it is −6.47%. Had it come out positive the hypothesis was dead whatever a task metric said, and the arm would not have run.

• **L3 refuted.** The registered prediction was that this capture's home advantage would be SMALLER than RULER's 9.55 pp, because RULER is synthetic and internally heterogeneous while LongBench is natural prose of several kinds. It is 11.52 pp, larger. Whatever the axis is, it is not simply distance from natural language.

At a task metric. LongBench at ratio 0.9375, fraction 0.20, 742 of 750 examples paired across all four arms, matched by script and press class:

pair	Δ	90% over examples	90% over the 16 tasks
book, separable − uniform	+0.10	[-0.96, +1.15]	[-0.69, +0.71]
book, contribution − uniform	**+2.59**	[+1.34, +3.86]	[+0.96, +4.18]
LongBench-cal. − uniform	+1.45	[+0.30, +2.62]	[+0.45, +2.93]
LongBench-cal. − book, contribution	**−1.14**	[-2.35, +0.02]	[-2.24, +0.59]
LongBench-cal. − book, separable	+1.35	[+0.22, +2.51]	[+0.24, +2.97]
book, contribution − book, separable (7-rung)	+2.49	[+1.20, +3.78]	[+0.91, +4.20]
book, 4-rung separable − uniform	−0.20	[-1.25, +0.86]	[-1.00, +0.37]
book, contribution − book, 4-rung separable	**+2.79**	[+1.50, +4.14]	[+1.03, +4.66]

The fourth row is the corpus effect, and it is negative. `alloc_book_4rung.json` differs from the LongBench-calibrated vector in *one* provenance field, the capture: allocator, alpha, both fit parameters, arm, curve, fitted window, heads, layers, ladder, normaliser, operating budget, ratio, both regularisers and the scoring window are identical. Against it, calibrating on the benchmark's own text instead of on a nineteenth-century novel is worth -1.14 [-2.35, +0.02], and $[-2.24, +0.59]$ over the task types. The earlier $+1.49$ was this comparison plus an objective change and a ladder change; it was reported as the corpus effect and it was not one.

LongBench. The book-calibrated contribution vector is worth $+2.59$ [+1.34, +3.86] over uniform, and $[+0.96, +4.18]$ over the sixteen task types: the largest effect any arm has on this benchmark, on the harder population as well as the easier one, from an allocation fitted on a novel.

The objective choice. `runs/objiso.md` was committed before its arm ran. The last two rows are the same allocation recipe differing in *one* provenance input, `curve`: same capture, same four-rung ladder, same operating budget, ratio, α, both regularisers, both fit parameters, both windows, same arm and normaliser. **The separable objective's allocation is worth nothing on this benchmark**, -0.20 [-1.25, +0.86] against an equal split, and the contribution's is worth $+2.59$; between them $+2.79$, clearing on both populations. This is the choice this paper argues for, isolated at a task metric, and it is the comparison §6.6 reports as unresolved on RULER at $+0.15$ [-0.52, +0.83].

It wins from behind. On the held-out window of the very capture the task is scored on, the objective puts the separable vector 3.36 pp *ahead* (-5.48% against -8.84%), and on `s0001`'s own test window 2.28 pp ahead ($+7.01\%$ against $+4.73\%$); both are the aggregate CVaR this paper argues everyone should optimise. The benchmark puts the vector the objective rejects 2.79 points ahead. The registered uninformativeness condition — the separable arm's metric shape moving far from the 34.5%-zero the other arms share — did not trip, at 34.1%.

A second model family. `runs/llamalb.md` was committed before either arm ran and registered four readings. The third fired. On Llama-3.1-8B, same benchmark, same ratio and fraction, the same recipe on that model's own capture, 677 of 750 paired:

arm	mean	Δ vs uniform	90% over questions
uniform control	33.39	—	
separable, 7-rung (shipped)	33.87	$+0.47$	[-0.79, +1.80]
separable, 4-rung (matched)	33.75	$+0.36$	[-0.92, +1.64]
contribution, 4-rung	33.18	-0.21	[-1.46, +1.01]
contribution − separable		$\mathbf{-0.57}$	[-1.77, +0.59]

Neither side of that disagreement is one run. The book-contribution arm was run a second time on the same allocation, data, ratio and fraction, and the two result files are byte-identical (`sha256 6389d770...`). The harness is deterministic here, so these intervals carry no run-to-run component and the gap between the two model families is a difference between them rather than two draws from one distribution.

Nothing clears *under the policy this table uses*, and §F.9 shows that is doing work here. **The objective-choice advantage on LongBench survives on one capture under both refusal policies** (§F.8), and it is also the objective's largest promise going unpaid: on `llama_32768`'s own window the contribution vector reduces the aggregate tail by $+6.95\%$ against the separable one's $+0.96\%$, a 5.99 pp lead — the widest of any capture, and the capture §6.4's leave-one-out table puts top at $+3.153$ pp with 24 of 24 rows won.

And the comparison was not thin. The obvious reading of a null between two matched allocations is that they were nearly the same vector, and it is wrong here. The two Llama allocations differ in 80 of 256 cells, 31.2%, with 15.67% of the total budget separating them; the Qwen pair that produced $+2.79$ differs in 93 of 288, 32.3%, with 16.78% separating them. The pair that moved the benchmark and the pair that did not are separated by the same amount to within a point of cells and a point of budget, so whatever produced the -0.57, it was not two allocations with nothing to disagree about. Data: `betaxfam.json`.

Which way the dropped rows cut. 73 of 750 are dropped and not evenly: 70 press refusals on the separable arm against 21 on the contribution one and 1 on the control, every one of them the sliding window's floor failing on a short input where a spread allocation puts too little on some layer. The separable vector's layer spread is $7.12\times$ against the contribution's $4.73\times$, so it is undeployable on 9.3% of this benchmark against 2.8%. The pairing therefore drops the separable arm's worst cases, which is a bias in the contribution arm's favour, and the contribution arm loses anyway.

An earlier version of this paragraph stopped there and concluded the negative was safe. Knowing which way a bias points is not a bound on how large it is, and this one is large: scoring those refusals as zero rather than dropping them turns -0.57 into $+2.45$ [+1.09, +3.83], which clears (§F.9). The -0.57 is what this paper's policy reports and the $+2.45$ is what the pessimistic one reports, the gap is 3.02 points, and the two arms refuse 21 times against 70. **This capture does not settle the question in either direction.** What no metric on the intersection can show is the deployability gap itself: 729 of 750 examples against 680.

The confounded $+2.49$ row stays above it. Removing it once a clean version exists would hide that this paper reported the confounded number first, and the two agreeing to within 0.3 points is worth seeing: `runs/bookarm.md` spent an hour attributing a RULER gap to that same ladder before holding it fixed and finding the gap intact, and `runs/sevenrung.md` has since measured the ladder directly and found it worth -0.67 (§F.4).

Splitting at the median context (5412 tokens) does not rescue the corpus reading. The book-calibrated vector clears on both halves, $+3.04$ [+1.48, +4.68] below the median and $+2.15$ [+0.25, +4.12] above it. The corpus-matched vector is -1.91 [-3.37, -0.48] against it on the short half — clearing zero in the wrong direction — and -0.38 [-2.20, +1.52] on the long.

Costs and benefits. It costs the transfer-failure account of the flat benchmark, which was this section's reason for existing and which the paper carried into its abstract. Allocations are *not* worth what they are worth only on the corpus they were fitted on: on this benchmark the out-of-corpus allocation is the better one.

It buys a sharper instance of this paper's own claim. On LongBench's held-out text the objective scores the corpus-matched vector at $+2.68\%$ and the book-calibrated one at -8.84% — an 11.52 pp advantage for calibrating on the benchmark, the largest home advantage measured anywhere here, and a verdict that the book vector should not be allocated at all. At the metric the advantage is -1.14 and the vector the objective rejects is the best arm on the benchmark. Both signs are wrong, on vectors matched to one field, with objective and task scored on the same corpus. `runs/bookarm.md` is the same failure on RULER, registered as B3 before its arm ran; this is the second corpus, and gate L2 — registered, and read before any arm here — is the prediction it falsifies.

Data: `lbtab_book.json` (all eight rows; `lbtab_lbcal.json` is the three-arm predecessor, retained because it is what the earlier numbers were read from), `xfercap_lbcal.json`, `lbcorpus_prov.json`. Registration: `runs/lbcapture.md`, `runs/lbcap2.md`, `runs/lbbook.md`.

F.8 Six measurements

The contribution-against-separable choice has now been measured on LongBench six times, each on a matched pair differing in the single provenance input `curve`, each against the same benchmark at the same ratio and fraction, and — from `runs/betaxfam.md` — each pair separated by about the same amount, 31.2 to 33.2% of cells.

capture	model	paired	contribution − separable	over tasks	clears
`s0001`	Qwen3-8B	742	$+\mathbf{2.79}$ [+1.50, +4.14]	[+1.03, +4.66]	**both**
`s0003`	Qwen3-8B	678	$+\mathbf{3.67}$ [+2.37, +5.07]	[+1.51, +6.45]	both†
`s0004`	Qwen3-8B	727	$+0.84$ [-0.32, +2.05]	[+0.09, +2.19]	types
`mistral_30720`	Mistral-7B	730	$+0.82$ [-0.31, +1.96]	[-0.07, +1.60]	no
`s0002`	Qwen3-8B	727	$+0.51$ [-0.56, +1.61]	[-0.25, +2.18]	no
`llama_32768`	Llama-3.1-8B	677	-0.57 [-1.77, +0.59]	—	no†

† **These two rows depend on the refusal policy and the other four do not** (§F.9). They are the only two captures whose arms refuse at very different rates — 70 against 9 on `s0003` and 21 against 70 on `llama_32768` — and they are the only two arms in the set that starve a layer to 4096 tokens. Scoring refusals as zero rather than dropping the row turns `s0003`'s $+3.67$ into -0.13 and `llama_32768`'s -0.57 into $+2.45$.

Five of the six point estimates are positive, and **one of the six clears the benchmark's resolution under both refusal policies**. The rest were each registered before their arms ran, with the null branch written out as an outcome and, in the four run today, named as the expected one (`runs/llamalb.md`, `runs/mistrallb.md`, `runs/s0002lb.md`, `runs/s0034lb.md`).

The two Qwen rows are the ones to look at. They share the model, the context length, the benchmark, the ratio, the fraction, the arm, the normaliser, the ladder, the operating budget, the estimator, the uniform control and — checked by digest — the same scorer file. They differ in the corpus the allocation was calibrated on, and $+2.79$ becomes $+0.51$. Whatever the $+2.79$ is, it is not a property of Qwen3-8B that a second capture of Qwen3-8B reproduces.

Against an equal split. The matched difference asks which objective. The question a practitioner faces is whether to allocate this way at all, which is each arm against a uniform budget — the same control, the same paired examples:

capture	contribution − uniform	separable − uniform	which beats it
`s0001`	$+2.59$ [+1.34, +3.86]	-0.20 [-1.25, +0.86]	contribution
`mistral_30720`	$+1.46$ [+0.33, +2.57]	$+0.63$ [-0.38, +1.66]	contribution
`s0003`	$+4.01$ [+2.71, +5.35]†	$+0.35$ [-0.78, +1.47]	contribution†
`s0002`	$+0.50$ [-0.68, +1.67]	-0.01 [-1.06, +1.07]	neither
`s0004`	-0.87 [-2.03, +0.24]	$\mathbf{-1.71}$ [-2.82, -0.61]	neither¶
`llama_32768`	-0.21 [-1.46, +1.01]	$+0.36$ [-0.92, +1.64]	neither¶

¶ **"Neither" is about beating an equal split, not about resolution, and on these two rows the distinction matters.** `s0004`'s separable cell already excludes zero in the table as printed — at -1.71 it is resolvably *worse* than a uniform budget — and under the pessimistic refusal policy all four cells of these two rows exclude zero and all four are negative (-1.58, -1.91, -1.21, -3.67). So on `s0004` and `llama_32768` the honest reading is not that the benchmark cannot see these allocations; it is that under one of the two defensible policies it sees them clearly and they are losing. The rows marked † above change which objective wins; these two change whether the benchmark resolves them at all.

On LongBench the contribution allocation clears on two captures of six under both refusal policies, on the question and task-type populations both times, and the separable allocation clears on none of the six under either. The two are `s0001`, whose task-type interval is [+0.96, +4.18], and `mistral_30720` at [+0.26, +2.15]; their vs-uniform numbers move by 0.14 and 0.10 between the policies and keep clearing. `s0003`'s $+4.01$ does not survive: it becomes -0.02 (§F.9).

The separable optimum is not merely useless here. On `s0004` it is worse than an equal split under both policies, -1.71 [-2.82, -0.61] and -1.91 [-3.01, -0.81], with 3 task types up and 13 down; on `llama_32768` it is -3.67 [-5.14, -2.16] under the pessimistic policy. What the field optimises, solved exactly, costs task points on two of six captures and gains on none.

The contribution allocation loses on those same two captures, and this paragraph would be dishonest without the sentence. Under the pessimistic policy it is -1.58 on `s0004` and -1.21 on `llama_32768`, both intervals excluding zero. So the loss side is *symmetric*: under the policy that assumes the worst about refusals, both objectives are significantly worse than an equal split on the same two of six captures, and the objective this paper argues for is not exempt. What is asymmetric is only the win side, and that is the whole of the claim: **the contribution allocation clears above an equal split on two captures under both policies and the separable optimum on none of the six under either.** A reader who takes away that our objective is safe and theirs is dangerous has taken away more than the table supports.

This is a statement about LongBench and not about the allocations. On RULER at 4096 both arms beat an equal split and both clear: $+2.59$ [+1.78, +3.40] for the contribution curves and $+2.45$ [+1.63,

+3.29] for the separable ones (§F.2). The two benchmarks disagree about whether the separable objective buys anything at all, which is the same disagreement §6.6 prices: RULER at 1300 questions resolves a smaller effect than LongBench at 750, and an arm that clears on one and not the other is what a resolution difference looks like. With six captures this is a count and not a test, and the two that clear are not the two with the largest objective difference, so it is reported as what it is: the objective this paper argues for produced an allocation that beat an equal split twice, and the objective it argues against produced one that never did.

A second benchmark. Registered three times: `runs/rulersq.md` put two captures' contribution and separable vectors on RULER at 4096, and `runs/sqsix.md` then registered the generalisation as a rule and tested it on three more captures before any of them ran, and `runs/mistralruler.md` registered the sixth as a completeness run that could not revive it. Same ratio, same fraction, vectors deployed unchanged; 1289 to 1300 paired questions.

capture	LongBench, drop	LongBench, zero	RULER
s0001	$+\mathbf{2.79}$ [+1.50, +4.14]	$+\mathbf{2.76}$	$+0.56$ [-0.36, +1.48]
s0002	$+0.51$ [-0.56, +1.61]	$+0.50$	$+\mathbf{2.86}$ [+1.91, +3.78]
s0003	$+\mathbf{3.67}$ [+2.37, +5.07]	-0.13	$+\mathbf{5.65}$ [+4.71, +6.62]
s0004	$+0.84$ [-0.32, +2.05]	$+0.33$	$+0.74$ [-0.01, +1.51]
llama_32768	-0.57 [-1.77, +0.59]	$+\mathbf{2.45}$	$-\mathbf{4.86}$ [-5.93, -3.83]
mistral_30720	$+0.82$ [-0.31, +1.96]	$+1.10$	$-\mathbf{0.85}$ [-1.56, -0.13]

The rule we registered is false. It required every capture to be resolvable somewhere. s0004 is resolvable on neither benchmark: $+0.84$ on LongBench under both policies, $+0.74$ [-0.01, +1.51] on RULER, missing by a hundredth on the lower bound. An earlier draft of this paragraph stated the rule over the first two captures, and it is withdrawn rather than softened.

What survives is smaller and still worth the table. On s0001 and s0002 the choice is resolvable on exactly one benchmark each, and a *different* one: a reader given either benchmark alone would name a different capture as the one where the objective choice matters. That is a fact about two captures, not a law.

The larger fact is the spread. Across these cells the matched objective choice is worth between -4.86 and $+5.65$ points, and *both extremes clear on both bootstrap populations*. Three resolved cells favour the contribution objective and two favour the separable one — on llama_32768 at RULER the separable allocation beats contribution by 4.86 and beats an equal split by $+6.47$ [+5.33, +7.63], which is the largest vs-uniform effect in this paper and belongs to the objective we argue against. So this is not a small effect that evaluations cannot see. It is a large effect whose *sign* and *location* nothing we measure predicts: not the objective gap (§F.4, refuted twice below), not the benchmark, and not any of the ten capture statistics of §D.3.

Both negatives are the two non-Qwen families, and we did not predict that. `runs/mistralruler.md` registered, before the arm ran, that a negative here would put two of two non-Qwen families behind the separable objective, that this would be a pattern worth naming, and that it was being written down at that moment precisely so that it could not afterwards be presented as foreseen. So it is reported as a pattern we did not call, and it is confounded three ways over: the four Qwen captures share a model, a dump lineage and a layer count, so "non-Qwen" is not one variable; two families are not a sample; and the two negatives are 4.86 and 0.85, an order of magnitude apart, the first the largest effect in this paper and the second barely clearing. What it does do is survive as one more cell the objective gap fails to call: mistral's gap is the second largest in the study, and it points down. That cell clears on questions and not over task types — $[-2.19, +0.17]$ over the thirteen, 8 of them down — so it is the weaker of the two negatives in every sense except its sign.

And one capture where allocating is simply wrong. On s0004 at RULER both allocations are resolvably worse than an equal split — contribution -1.44 [-2.18, -0.69], separable -2.19 [-3.00, -1.36], on both populations, 6 and 8 task types down. With the separable arm's -1.71 on LongBench, two benchmarks now agree that allocation costs points on this capture.

Refusals, and a gate that hid them. s0003's contribution arm refuses 9 times and llama_32768's separable arm 11; every other arm here refuses zero times. Both asymmetries are small enough to

move their cells by about 0.2 points against effects of 5.65 and 4.86, so neither conclusion turns on §F.9's policy. We report the counts because our own dead-row gate reported clean on all seven arms while reading none of them — it skips the per-row frame by design and was handed exactly that file — and the four arms of `runs/rulersq.md` turned out to be genuinely clean only by luck.

Is RULER simply easier? Its uniform control scores 14.59 against LongBench's 34.06, and a weak baseline is easier to move; `runs/sqsix.md` recorded before the run that it could not settle this. One cell does. A scaling cannot reverse a sign, and under the pessimistic refusal policy `llama_32768` is $+2.45$ [+1.09, +3.83] on LongBench and -4.86 [-5.93, -3.83] on RULER — opposite signs, both intervals excluding zero, the same two vectors. No positive scaling maps one to the other. Two weaker counterexamples point the same way and are kept because they do not depend on a policy: `s0001`'s matched difference does not clear on RULER though it clears on LongBench, and `s0002`'s separable arm does not beat an equal split on RULER at all ($+0.16$ [-0.63, +0.91]). A uniformly easier benchmark produces neither. **The sign argument holds under one of the two defensible policies and not the other**: under the drop policy LongBench is -0.57 and does not clear, so the sign disagreement is not established there, and the argument is reported as policy-conditional. It was also not registered, which makes it an observation rather than a test.

The transmission band. Two of these were priced through the band of §F.4 *in advance* and two are priced here after the fact, which is stated because it changes what the agreement is worth:

capture	priced	objective gap	band predicts	measured
`mistral_30720`	in advance	$+5.06$ pp	$+0.34$ to $+1.51$	$+0.82$, inside
`s0002`	in advance	$+2.95$ pp	$+0.20$ to $+0.88$	$+0.51$, inside
`llama_32768`	after the fact	$+5.99$ pp	$+0.40$ to $+1.79$	-0.57, sign
`s0001`	after the fact	-3.36 pp	-0.23 to -1.00	$+2.79$, sign

The band placed both registered predictions correctly, including that neither would clear the ± 0.83 resolution — and it is wrong in *sign* on both of the others. **The one comparison that clears is the one the band gets most wrong**: the objective preferred the separable vector on `s0001` by 3.36 pp, which prices a loss of up to a point, and the benchmark returned $+2.79$.

Interpretation. Read as a method result it is mixed rather than thin: one matched comparison in six is resolvable, while the allocation itself beats an equal split on two captures of six and its separable control on none. The paper says both in §A rather than around them. Read as a measurement it is the sharpest thing here. The same design choice, held matched to one provenance field and measured six times on the same benchmark, produces one interval that excludes zero and five that do not; and where the objective difference was priced in advance, the arithmetic said in both cases that the effect would be too small for this benchmark to see, and in both cases it was. That is §6.6's claim about resolution, tested on this paper's own recommendation and not only on other people's.

We do not pool the six. The captures differ in model and corpus, the independent unit is the capture, and six is not a sample; the median above is a description of the table and not an estimate of anything.

F.9 THE REFUSAL POLICY

The press refuses when a layer's budget after compression falls below the observation window it must preserve. `lbtab` treats that refusal as missing data rather than as a wrong answer, so the pairing drops the row from *both* arms and the count is reported. The reason is sound — scoring a refusal as the near-zero string it is would charge an allocation twice, once by losing the answer and again by scoring the loss — and it carries a hazard that the same comment names: the policy lands on whichever arm refuses more often.

On two of six captures it lands hard enough to change the answer, and in opposite directions. Below, *drop* is that policy and *zero* scores a press refusal as zero instead — the pessimistic bound for whichever arm refuses more:

capture	refusals		contribution − separable		shift
	contrib	sep	drop	zero	
s0001	2	2	$+\mathbf{2.79}$ [+1.50, +4.14]	$+\mathbf{2.76}$ [+1.49, +4.07]	-0.03
s0002	21	21	$+0.51$ [-0.56, +1.61]	$+0.50$ [-0.59, +1.59]	-0.01
s0003	70	9	$+\mathbf{3.67}$ [+2.37, +5.07]	-0.13 [-1.69, +1.43]	-3.80
s0004	21	9	$+0.84$ [-0.32, +2.05]	$+0.33$ [-0.88, +1.52]	-0.51
llama_32768	21	70	-0.57 [-1.77, +0.59]	$+\mathbf{2.45}$ [+1.09, +3.83]	$+3.02$
mistral_30720	14	20	$+0.82$ [-0.31, +1.96]	$+1.10$ [-0.04, +2.24]	$+0.28$

Bold marks an interval excluding zero over questions. **s0001 clears under both policies and is the only capture that does.** s0003 clears only under *drop* and llama_32768 only under *zero*.

Direction. A refusal is caused by the most starved layer, so the refusal count is a function of the allocation and not of the benchmark. Over all twelve LongBench arms, the minimum layer budget and the refusal count line up: 8704 tokens gives 2 refusals twice, 7168 gives 9, 9 and 14, 5632 gives 20 and 21 three times, and 4096 gives 70 twice. Pearson between log minimum layer tokens and refusals is -0.915, and **the only two arms that starve a layer to** 4096 **are the only two with** 70 **refusals** — s0003's contribution arm and llama_32768's separable arm. Those are exactly the two captures whose answer moves.

Size. Scoring one refusal as zero rather than dropping it moves that arm's mean by roughly its own mean over the paired count, so the shift should be the refusal *difference* times $\bar{x}/n$. Predicted against measured over the six captures gives a Pearson of $+0.999$ at slope 1.22. This is not a source of noise to be averaged away; it is a deterministic offset whose sign and magnitude can be computed before the arm runs.

Resolution. Neither policy is obviously correct. Dropping a refusal treats it as missing, which it is; scoring it zero treats it as a failure, which it also is, since a deployed allocation that starves a layer below the window floor would fail on exactly those inputs. We keep *drop* as the reported policy, because it is the one every number in this paper was computed under and because the alternative charges an allocation twice, and we report both wherever they disagree. **The claim that survives is the one that survives both**: on LongBench the objective choice is resolvable on s0001 and on no other capture under any policy applied to all six alike.

A matched comparison between two allocations that refuse at different rates is confounded by this, whoever runs it, and the remedy is cheap: report the refusal counts, and if they differ materially, report both policies. Data: refpolicy.json.

Main-text numbers. Everything above is about the six matched capture pairs. The paper's headline LongBench claims are in Table 1 and are *different arms*, so they were re-scored under both policies rather than assumed to inherit the answer:

pair	model	drop (published)	refusals as 0	verdict
tailalloc − uniform	Qwen3-8B	-0.04 [-1.10, +1.04]	-0.21 [-1.29, +0.89]	null, null
tailalloc − uniform	Llama-3.1-8B	$+0.35$ [-0.81, +1.55]	-0.18 [-1.42, +1.01]	null, null
contribution − uniform	Qwen3-8B	$+1.45$ [+0.28, +2.58]	$+1.38$ [+0.25, +2.55]	clears, clears

None of them turns on the policy. The two LongBench nulls the paper reports stay null and the one LongBench arm that clears keeps clearing; the largest movement is 0.53 points, on the Llama null, against a half-width of 1.18. The drop-policy column reproduces the committed tables to four decimals at identical paired counts, so the only difference between the columns is the policy.

We tried to settle this with arithmetic first and the data refused. The natural move is to bound the shift by the refusal difference times the mean over the paired count — the same quantity that predicts the six rows above at $r = 0.999$. It is not an upper bound: the measured shift *exceeds* it by 21% on s0003 and 24% on llama_32768, and inflating it enough to be safe left the Llama row marginal (1.14 against a half-width of 1.18). A relationship strong enough to predict is not thereby strong enough to bound, which is why these four cells were measured.

G SUPPLEMENTARY ANALYSIS

G.1 AN ALMOST-RIGHT ALLOCATION

Effect size. A greedy on the lower convex envelope spends each token where the marginal gain is largest and never reads how large, so any distortion preserving the *order* of marginal gains induces the same allocation. Over the 92 rows where both quantities join, the separable objective overstates the tail by tens of percent while agreeing with the contribution's ordering at median Spearman 0.935: half wrong about every value, almost right about every pair. Tested by intervention over 30 (capture, arm) rows the mechanism holds and the diagnostic it suggests does not — across rows no statistic of the two curves orders the loss, and one has the wrong sign — so **the recommendation to spend the effort where rank agreement is low is withdrawn**: the number has to be got by building both allocations and scoring them. The rest of this section is that test.

Two of this paper's measurements read as a contradiction. The separable objective overstates the aggregate tail it bounds by tens of percent (§4.1), and replacing it with the right quantity is worth one to three percentage points that no task benchmark resolves at any n anyone will pay for (§6.6). An objective wrong by half that costs almost nothing to keep needs an explanation, and there is one.

A greedy allocator consumes a ranking, not a value. Fox's algorithm on the lower convex envelope repeatedly spends a token where the marginal gain is largest; it never reads how large that gain is. Any distortion of the objective that preserves the *order* of marginal gains therefore induces exactly the same allocation, however wrong its numbers. The separable objective is such a distortion nearly everywhere:

Over those 92 rows the separable objective overstates the aggregate tail by tens of percent while agreeing with the contribution's ordering at a median Spearman of 0.935 (interquartile 0.902–0.956). How large the overstatement is and how it is distributed is the companion measurement submission's own result and is not restated here.

The two quantities live in different files and are *joined* rather than assumed to line up: the rank agreement is in the β diagnostics over 140 rows and the overstatement in the aggregate sweep over 92, they share (capture, policy, normaliser, tail level), and 92 is therefore what can be said about the pair. Being 55% wrong about the size of every cell's risk while agreeing on the order of nearly every pair of them is exactly the configuration in which the correction is real, provable, and small.

The spread is real. The minimum rank agreement is 0.620, not 0.935: the least-agreeing decile of rows sits at 0.727 against 0.974 for the most-agreeing one, and those rows are not rare enough to ignore. The two diagnostics do not move together, and that is the point. The least-agreeing decile overstates the tail by 58.1% and the most-agreeing by 50.4% — nearly the same. How loose the bound is and how badly it reorders the cells are different questions, and only the second governs what an allocator does.

Intervention test. The paragraph above is correlational, and on inspection of the code it is measured on the wrong field: `rcshrink.greedy` never compares two curve values, it compares marginal *rates* on the envelope and the *sign* of the best one. Four scripts, each with its predictions committed to git before it was run, test the mechanism directly on 30 (capture, arm) rows over five captures and two model families.

The proposition, checked rather than asserted. If the greedy reads only the rate ordering and the sign, any strictly increasing sign-preserving transform of the rate field must leave the allocation bit-identical. Applied as the most value-destroying such transform — every rate replaced by its rank — the index vector is identical on every row, and `ratepath.py` refuses if it is not. The sign-preservation is not decoration: a transform mapping negative rates to positive ranks would buy steps the real greedy refuses, and that stopping test is the one place a *value* enters the algorithm. The guard was itself unreachable at $b_0 = 2048$, where the budget runs out before the rates turn negative; raised to the top of the ladder the mutation that removes sign-preservation moves 3 to 7 cells on every row that has a non-positive rate.

Rank error is causal. Permuting the contribution's rate field, its value multiset held exactly so that the value error is zero by construction, costs objective in proportion to how far the permutation moves the order: pooled Spearman $+0.885$ over 2910 perturbed allocations, non-increasing across every bin.

Registered prediction P1 said the rate-rank agreement would be *lower* than the 0.935 the paragraph above quotes, because differencing amplifies disagreement, and it is: median 0.863.

And it is not diagnostic. Four statistics were tried, three of them registered in advance as replacements after the one before it failed. Every one of them orders the objective loss within a row, on permutations; none of them orders it across rows, on the real separable objective, and value-weighted discordance has the wrong sign:

Spearman against objective loss	rate rank	near λ^*	weighted	bought set
within a row (permutations)	$+0.865$	$+0.824$	$+0.876$	$+0.879$
across rows (the real one)	-0.082	$+0.071$	-0.321	-0.047

The second column is the result. The band statistic is a refutation rather than a result only because its own registered control caught it: W2 predicted that agreement near the greedy's stopping threshold would be *higher* than away from it, and it is lower in 30 of 30 real rows — and lower in 600 of 600 random permutations, which have no structure to have produced it. The band is where the rates are nearly tied, and ranking near-ties is noise for anyone.

Both columns are true at once because the misorderings are structured. At matched rank distance the separable objective is less discordant than a permutation in 27 of 30 rows and loses 2.92 pp less; how structured it is varies across workloads more than how much. That is a sharper statement than "rank is all that matters" and a weaker one than the paragraph above made.

The recommendation is withdrawn. This appendix used to close by telling a practitioner to measure the rank agreement, which needs no allocation built, and spend the effort where it is low. That rested on a capture-level correlation over five captures spanning 0.027 of rank agreement, and $+0.067$ over the 29 rows those five captures average; `predictgap.json`'s capture-level figure is -0.800, the sign the reading required, against -0.800 with the *wrong* sign for β spread. Five points spanning three hundredths of a statistic are not evidence for a rule, and the rule is withdrawn rather than qualified. The mechanism it was drawn from survives; what is gone is the shortcut. A practitioner who wants to know what the correction is worth on their workload has to build both allocations and score them — two greedies and two CVaRs, cheap in absolute terms and expensive only relative to a diagnostic that does not exist. As a duplicate control, this code path rebuilds both allocations from the curves and re-scores them, and its median separable-minus-contribution gap reproduces the shipped $+1.96$ pp to 0.01 pp.

None of this rescues the task-level null and it is not offered as one: the null stands and is reported as §6.6 reports it. Data: `whysmall.json`, `predictgap.json`, `ranklaw.json`, `ratepathagg.json`.

G.2 The transfer matrix

The specialising factor. The diagonal wins only 3 of the 5 columns, so within a corpus type it barely matters which book you calibrated on. Across it, it decides whether allocating beats not allocating: every book allocation is below an equal split on RULER's own queries, and the home advantage runs from $+9.55$ to $+15.32$ pp, of which we quote the smallest. At a task metric the matrix survives as a measurement of the objective; the reading that made it a recommendation does not (§6.5), and the reversal arrives on the capture where the objective-side case is strongest.

Every allocation in this paper is calibrated on a nineteenth-century novel and spent on synthetic retrieval, and §A used to name a corpus-matched allocation as the one lever we declined. The lever now exists. `rulercorpus.py` builds a calibration corpus from RULER contexts, and **the only thing that makes it legitimate is disjointness**, so it is enforced rather than assumed: the harness evaluates a 20% subsample drawn per task with a fixed seed, the script reconstructs exactly that subsample, takes its text only from the complement, and asserts the intersection is empty before writing anything. It also refuses to write a corpus shorter than the context it will be dumped at, because a prompt recycled to fill 32768 tokens is not a sample from the distribution any more.

This is also where this paper's own claim is largest: the separable objective's allocation gains $+0.03\%$ there where the contribution's gains $+3.08\%$, a gap of 3.04 pp against the 1.96 median over the book captures.

Objective reduction against an equal split, each allocation scored on the held-out window of the capture naming its column:

calibrated on	RULER	s0001	s0002	s0003	s0004
RULER	+3.08	+0.37	−3.05	−2.50	−0.94
s0001	−6.47	+4.73	−1.17	+3.85	+0.03
s0002	−12.24	+8.18	+5.07	−0.46	−2.89
s0003	−12.06	+3.21	−3.45	+3.85	+0.81
s0004	−12.07	+1.44	−8.57	−6.24	+2.66

Three predictions were registered before the matrix ran. **X1 holds.** All four book allocations are below an equal split on RULER's held-out queries. **X2 is refuted, and the refutation is the better result.** The diagonal was predicted to win every column and wins three of five; s0002's allocation beats s0001's own on s0001, +8.18 against +4.73. Capture identity is not the axis — corpus *type* is — which is also why pooling several books does not beat picking one, a refusal this project had already measured and could not previously explain. **X3 holds thinly and is reported as thin:** the RULER column is the widest, at 15.32 pp against a book maximum of 13.64, and that maximum is itself inflated by the RULER row sitting inside it.

The 2×2 the body quotes is the conservative corner of this matrix. s0001 is the *least* damaged book allocation on RULER by a factor of two, so the +9.55 pp home advantage is the smallest of the four; against s0002 it is +15.32. We quote +9.55 because s0001 is the capture the rest of the paper scores.

How little of the right corpus is enough. Rebuilt from k positions of the RULER capture instead of all 512, the allocation beats the full-window book allocation's -6.47% at $k = 32$ — a third of a tail point — on 6 of 6 seeds, and beats an equal split from $k = 128$. Two different thresholds, both worth knowing: 32 positions of the right corpus to beat the wrong corpus, 128 to beat doing nothing. The registered monotonicity prediction is *refuted* — 64 gives $+1.09\%$ and 96 gives -0.23%, and the widest seed spread is 5.04 pp — so the curve is read as a band and not a line.

What the matrix does not say. The shipped allocation, the one the task arms of §6.6 run, gains +2.38 points on RULER at this operating budget, and every four-rung book allocation in the matrix is below an equal split on RULER's objective. That reads as a contradiction and is not one, because they are not the same vector: the matrix is on the four-rung ladder and the shipped allocation on the seven-rung one, their per-layer profiles agree at Spearman 0.444, and 109 of 288 cells are identical. The shipped vector has not been scored on an in-distribution capture at all; that needs a seven-rung harvest of it, which does not exist. Sideways, the same comparison is another instance of what §G.1 is about: 38% cell agreement between two ladders buys $+1.46\%$ of objective.

The task-level pair, registered before it ran. On RULER at context 4096 and compression 0.9375 — 256 tokens a head, 1300 shared questions, matched by script and press class, differing only in the capture the allocation was fitted on:

allocation	score	Δ vs uniform	90% by question / by type
uniform control	14.59	—	—
book-calibrated	17.90	+3.31	[+2.48, +4.13] / [+0.62, +6.36]
RULER-calibrated	15.37	+0.78	[+0.17, +1.42] / [-0.08, +1.72]
RULER minus book		−2.53	[-3.41, -1.66] / [-5.49, -0.02]

The RULER-calibrated arm clears one of the two populations this paper requires of a strong claim, with cwe carrying 46% of its mean and +0.46 left without it. The book-calibrated arm clears both. The difference between them clears both, in the direction opposite to the one the objective gives.

Three predictions were written while the first arm was scored and the second was still on the card. **B1 said the book arm would land below** 14.59, because the objective calls that allocation 6.47% worse than an equal split on this text and the band prices that at −0.94 to −1.93 points; it measured 17.90. **B2 said the pair would clear zero in the objective's direction**; it clears in the other one. **B3 said that if B1 failed this way**, the transmission model would be wrong in sign rather than merely imprecise on the one row that could show it, the specialisation finding would be an objective-side

phenomenon with no demonstrated task consequence, and the corpus row of §F.4 would come out. All three of those follow, and the row is struck.

B1 was the sharpest test of the objective in this paper and it was set up to be: every other transmission row is a positive objective predicting a positive task effect, so none of them could have caught a sign error at all. This one could, and did.

What that leaves standing. Two scopes first: the capture matches the *corpus* and not the task format, and it is one capture and one press arm, on the four-rung ladder the shipped seven-rung vector agrees with on 109 of 288 cells. The transfer matrix is unaffected as a measurement — four-rung allocations against four-rung allocations on held-out windows, with the 9.55 pp home advantage measured rather than inferred. What is refuted is the reading that it predicts task outcomes. That converts it from a recommendation into a *validity failure of the objective as a proxy*, measured directly rather than argued: the disagreement §6.2 reports on layer-DefensiveKV's vector, now on ours, with both arms built by one script at one budget on one ladder and scored on one corpus. There is no implementation difference left to attribute it to.

A correction we owe the record. An hour before this arm landed we noted that the shipped seven-rung book-calibrated allocation beats the four-rung RULER-calibrated one by 3.43 points and attributed the gap to the *ladder*, that being what differed. With the ladder held fixed the book allocation still wins, by 2.53. It was never the ladder, and the note is superseded on that point (`runs/laddervscorpus.md`, `runs/bookarm.md`).

The same capture, run through the other two experiments. One capture and one press arm — only `snapkv` was harvested here — so these are kept in their own directory and pooled into nothing. The polish mechanism replicates on the different corpus type with the same signs and the same ordering: scored at 0.99, searching at 0.95 gives $+0.28$ and $+0.26$ on the two normalisers and searching at 0.99 gives -0.70 and -0.96. The rank-law result gains a clause rather than a qualification: on this capture the separable objective's loss lands *inside* the random band at its own rank distance, where on the book captures it sat below the band in 27 of 30 rows. How structured the misorderings are is itself corpus-dependent, which is the same thing the failure of rank distance as a cross-row diagnostic was already saying. Data: `xfermat.json`, `xfer2x2.json`, `xferindist.json`, `howlittle.json`, `shipshape.json`, `indistcap/`.

G.3 The local search

Fox's greedy is exact for a separable objective and the aggregate tail is not one, so the shipped allocation is optimal for a first-order surrogate. A budget-neutral exchange search on the aggregate itself recovers $+0.42$ pp $[+0.15, +0.70]$ over 120 rows and improves 47 of 60 — and five tail points are enough to *select* among eight estimators and not enough to *search* against. Measured over those 120 rows and five captures, on the test window, with the bootstrap resampled *by capture* because six press arms share one:

scored at	searched at	median	90% by capture
0.99	0.95	$+0.42$ pp	[+0.15, +0.70]
0.99	0.99	$+0.06$ pp	[-0.82, +0.74]
0.95	0.95	$+0.05$ pp	[-0.01, +0.21]
0.95	0.99	-0.59 pp	[-1.51, -0.44]

Coupled at 0.99 it is $+0.06$ $[-0.82, +0.74]$; searched deep and scored shallow, -0.59 $[-1.51, -0.44]$. Registered Q4 — searching at the shallower level beats coupling it — holds on 95 of 120 rows. Registered Q3 holds too: $+0.42$ is smaller than the $+1.96$ that changing the *quantity* is worth, so the first-order story still carries the paper. The mechanism is the one §F.3 reports on the fitting side, and putting the two together sharpens it. **The same five tail points are adequate for a selection and inadequate for a search.** Replacing the shipped validation-objective criterion with rate-rank agreement — the arm `ranksel.py` was written to run — is worth -0.06 pp, and its *premise* was wrong. The registration asserted that selecting on five tail points leaves about a point of bias on the table, and two independent code paths say it does not: the shipped pipeline records its own peek-minus-chosen gap as `selection_bias_pp` in `rcshrink_*.json`, a median 0.049 pp over all 140 rows, and the re-implementation here measures 0.023 pp over the 120 rows on which

all three criteria could be scored. Two row sets, the same order of magnitude, and both a fortieth of what was assumed. The bias is not uniformly negligible — it is *concentrated*, with a mean of 0.295 pp, a maximum of 3.137 and 28 of the 140 rows above half a point — but it is not what the median row has to give up, and there was nothing for a better criterion to capture. The `mse_uni` control, which has more support than five points but is not a rank statistic, lands in the same place at -0.04 pp, so registered R3 is refuted as well: what failed is not the choice of statistic. One choice among eight is a degree of freedom the deep tail can afford; a search over exchanges is not. Data: `methodagg.json`.

G.4 An allocator that beats us

§6.6 reports that layer-DefensiveKV's own vector beats uniform on RULER by $+2.07$ while our objective calls it worse than an equal split. That is the strongest evidence in this paper against its own objective, and it has a candidate mechanism visible in the allocations themselves. Tokens per layer as a share of the total, 36 layers read in thirds:

allocation	early	middle	late	spread	objective	task
uniform	0.333	0.333	0.333	1.00×	—	—
Ada-KV	0.333	0.333	0.333	1.00×	−8.93%	−0.94
ours	0.343	0.274	0.383	4.27×	+13.42%	+4.21
layer-DefensiveKV	0.446	0.326	0.228	8.86×	−4.31%	+2.07

The mechanism is in the vectors — layer-DefensiveKV front-loads, giving the first third of layers 0.446 of the budget against the last third's 0.228, where our objective weights layer 3 and layer 33 identically — and it is a named absence, with our own late-weighted arm as the case against it. Ada-KV being flat to three decimals is a confirmation rather than a finding: it is a within-layer allocator and §G.9 describes it as one. Ours is mildly late-weighted, 0.040 of the budget behind between first third and last. layer-DefensiveKV front-loads hard, 0.218 of the budget ahead, nearly twice the share in the first third that it gives the last.

What the objective cannot see. The aggregate is a tail statistic of the *sum* of per-cell errors with every cell weighted equally, so a unit of attention error at layer 3 and a unit at layer 33 enter it identically. They do not enter the model identically: an early error is transformed by every subsequent layer, and this objective has no term for that. An allocator that front-loads is buying something the objective does not price, and layer-DefensiveKV front-loads more than anything else here.

Why this is not a fix and is not presented as one. Reweighting the objective by depth until layer-DefensiveKV scores well would be fitting the objective to the single observation that embarrasses it, and that observation is $n = 1$: one allocator, one benchmark, one model. Worse, the second data point does not fit. Both of the two arms that beat what their objective score predicts — layer-DefensiveKV and ours — do so, and only one of them front-loads; we are mildly *late*-weighted. What can be said without fitting is that the disagreement has a candidate mechanism, that the mechanism is a named absence in the objective rather than a mystery, and that it makes a prediction this paper does not have the arms to test: allocators concentrating budget early should beat their objective score, and allocators concentrating it late should fall short. Two points is not a test of that, and the confirming case and the case that does not fit are both named above. Data: `depthblind.json`.

G.5 The optimality condition

G.6 Proposition 1

Proof. Under an elliptical joint law for (X_i, Z) the conditional expectation is linear, $\mathbb{E}[X_i \mid Z] = \mu_i + \beta_i(Z - \mu_Z)$, so conditioning on the aggregate's tail and using (4),

$$\mathrm{RC}_i = \mu_i + \beta_i \left(\mathrm{CVaR}_\alpha(Z) - \mu_Z\right) = \mu_i + \lambda_\alpha \varrho_i \sigma_i,$$

where the second equality substitutes $\mathrm{CVaR}_\alpha(Z) - \mu_Z = \lambda_\alpha \sigma_Z$ and $\beta_i = \varrho_i \sigma_i / \sigma_Z$. The same family gives $\mathrm{CVaR}_\alpha(X_i) = \mu_i + \lambda_\alpha \sigma_i$, and subtracting gives

$$\mathrm{CVaR}_\alpha(X_i) - \mathrm{RC}_i \;=\; \lambda_\alpha \sigma_i \left(1 - \varrho_i\right) \;\geq\; 0, \tag{7}$$

Table 5: The quantity of Proposition 1, measured at the uniform budget on held-out queries, and quoted in §6. Ranges and medians run over the 140 rows, and β_i over the n cells within a row; $1/n$ is the value β_i would take under equal exposure. The condition asks for one number in the first three lines and gets a spread in all three.

Quantity	Measured
β_i, smallest and largest over all rows	-0.0240 to $+0.1628$
$\max_i \beta_i$ as a multiple of $1/n$	median $14\times$, up to $47\times$
Cells with $\beta_i \le 0$ (tail hedges)	6.2–38.2%, median 16.0%
Rows containing no hedge at all	0 of 140
Change in tokens given to hedges, contribution vs. separable	median -27%
Spearman(standalone risk, risk contribution), per row	median 0.938, min 0.620

non-negativity being $\varrho_i \le 1$. Summing over cells, $\sum_i \mathrm{CVaR}_\alpha(X_i) - \mathrm{CVaR}_\alpha(Z) = \lambda_\alpha \sum_i \sigma_i(1 - \varrho_i)$, which is the gap in (3); it vanishes exactly when every $\varrho_i = 1$, recovering the comonotone equality condition. Two rankings by quantities differing by a cell-independent constant coincide, which gives the two sufficient conditions stated. □

Numerical check. `scripts/prop_recheck.py` draws four million samples from a random positive-definite Gaussian covariance and estimates RC_i as $\mathbb{E}[X_i \mid Z \ge q_\alpha(Z)]$ — the definition, not the closed form above — then compares $\mathrm{CVaR}_\alpha(X_i) - \mathrm{RC}_i$ against $\lambda_\alpha \sigma_i(1 - \varrho_i)$. Over three covariances spanning $\varrho_i \in [-0.18, +0.82]$, with hedge cells present in two of them, the largest per-cell disagreement is 0.003 at that sample size. **The summing half of the statement is an identity and not evidence**: Euler contributions sum to $\mathrm{CVaR}_\alpha(Z)$ by construction, so that total agrees to machine precision whatever the joint law, and it is reported as a check on the accounting rather than on the elliptical assumption. The proposition was false as first stated — see below — which is why it is verified along a path that shares neither code nor reasoning with the correction.

An earlier version. An earlier statement of this proposition said that **constant β_i implies the two orders coincide**, without restricting ρ in (1). That is false for $\rho = \mathrm{CVaR}_\alpha$, which is the branch this paper's own separable comparator uses. Constant β_i makes $\mathrm{RC}_i - \mu_i$ a cell-independent constant, so it pins the contribution order to the order of the *means* — not to the order of the standalone CVaRs, which carry an extra $\lambda_\alpha \sigma_i$. The two differ whenever σ_i varies.

A counterexample with a valid covariance: four cells, $\sigma = (2.09, 1.01, 1.80, 1.33)$, eigenvalues $(0.34, 0.79, 3.16, 6.13)$ all strictly positive, and $\beta_0 = \beta_1$ to machine precision while $\sigma_0 \ne \sigma_1$. Choosing $\mu_0 - \mu_1 = -1.11$ makes $\mathrm{RC}_0 - \mathrm{RC}_1 = -1.11$ and $\mathrm{CVaR}_\alpha(X_0) - \mathrm{CVaR}_\alpha(X_1) = +1.11$: equal β, exactly opposite orders.

What this does and does not cost the paper. The corrected proposition is the stronger statement — it gives the discrepancy exactly, rather than a sufficient condition for it to vanish, and it identifies (3), (4) and (7) as one identity seen three ways. Every empirical claim built on it stands: $\beta_i \le 0$ still marks the cells where the gap is largest, since ϱ_i and β_i share a sign, and the measured dispersion in β is still evidence that the discrepancy is not constant. What changes is what the paper may assert in one direction: constant β is sufficient only against a mean-based separable objective, and *most published allocators are mean-based*, which is why the condition is the right one to have measured.

G.7 The sum-of-cells assumption in the aggregate

Every quantity in this paper is a statement about $Z = \sum_i X_i$, the sum over cells of per-query attention-output eviction error. That the aggregate is that sum is a modelling choice and not a measurement. Errors at different layers do not compose linearly through a network: a deviation introduced in an early layer is transformed by every layer above it, and nothing here establishes that adding the raw per-cell deviations tracks the damage the model actually suffers.

What the choice does not affect. The comparison this paper makes is between two functionals of the *same* X_i: $\rho(\sum_i X_i)$ against $\sum_i \rho(X_i)$. Every allocator in Appendix H.1 aggregates by summing a per-cell statistic, so the sum is not our assumption imposed on other people's methods — it is theirs, and we adopt it to compare like with like. Nor does (3) depend on it: for any coherent ρ and any

aggregation that is not additive in the per-cell errors, the separable sum is an upper bound attained only under comonotonicity, so the claim that the field optimises a worst-case-dependence bound survives a different choice of Z. What would not survive is the attribution: RC_i in (4) is the Euler contribution *to this* Z, and a different aggregate has different contributions.

What it may explain. This paper reports, repeatedly, an objective-side difference that does not reach the task metric: four sign inversions, a 5.99 pp lead on Llama-3.1-8B paying nothing, a transmission band spanning a factor of three. We attribute that to the benchmark's resolution and to the objective not being a proxy for it, and both are supported. A third possibility is not excluded and is worth stating plainly: the surrogate may be the weak link, and a sum of raw per-cell deviations may simply be a poor stand-in for what a task loses. Distinguishing the two would need an aggregation validated against task damage rather than assumed, which is not something this paper does or can do with the captures it has.

The corrected proposition. The quantity separating the two objectives is $\lambda_\alpha \sigma_i (1 - \varrho_i)$, not $\beta_i = \varrho_i \sigma_i / \sigma_Z$. The two share a sign but not an ordering: dispersion in β can rise while the discrepancy does not, because σ_i enters the two differently. The registration was refuted before we knew that, and the correction of Appendix G.6 supplies the mechanism after the fact — which is weaker evidence than a prediction, and is labelled as such.

G.8 Dispersion in β

Proposition 1 puts the per-cell disagreement at $\lambda_\alpha \sigma_i (1 - \varrho_i)$, and β_i shares a sign with ϱ_i. It is tempting to read that quantitatively through β — more dispersion, more divergence — and `runs/betaxfam.md` registered exactly that reading, as the mechanism that would explain why the objective choice moves LongBench on Qwen3-8B (§F.7) and not on Llama-3.1-8B. The registration was committed before the quantities were computed. **All three of its predictions are refuted.**

On the arm and normaliser the matched allocations were built from (`snapkv`, `percell_o`, $\alpha = 0.99$), across the three captures that carry a matched LongBench pair:

capture	model	$\max_i \beta_i/(1/n)$	hedges	cells differ	budget	Spearman
`s0001`	Qwen3-8B	$41.61\times$	11.5%	32.3%	16.78%	0.904
`llama_32768`	Llama-3.1-8B	$17.46\times$	15.6%	31.2%	15.67%	0.901
`mistral_30720`	Mistral-7B	$9.07\times$	6.2%	33.2%	17.19%	0.933

β dispersion spans $4.6\times$ across the three, and neither quantity that would have to follow it moves: the two allocations differ in 31.2 to 33.2% of their cells whatever the dispersion, and the two curves rank cells the same way to within 0.03 of Spearman. The capture with four and a half times the β spread produces a pair no further apart than the capture with the least.

This is a limitation of how far the proposition can be pushed, not a counterexample to it: it claims an ordering under exact constancy and says nothing about magnitudes, which is why it is stated that way. What the measurement removes is the informal version — that a capture with more β spread has more to gain from the contribution objective — and with it the mechanism that would have explained the Llama non-replication. That non-replication stays unexplained and this paper offers no mechanism for it.

One detail of the registration is worth reporting, because it decided the answer. The prediction required both β dispersion and the hedge fraction to be lower on Llama than on Qwen. Under `percell_o` the hedge fractions are 11.5% and 15.6% and the second half fails. Under the other normaliser they are 18.1% and 9.0%, both halves hold, and the mechanism would have been reported as confirmed. The normaliser was pinned to the one the allocations record in their own provenance before either table was read, which is the only reason the confirming choice was not available.

G.9 The allocation axes

Re-solving under each constraint alone, the head axis reaches more of the joint reduction than the layer axis on every capture, both ladders and both tail levels, and never much more than a quarter of it the other way: a low ceiling on any purely layer-wise allocator (Appendices G.21, G.22).

The allocator turns two knobs: how many tokens a layer gets, and how a layer's tokens are split across its heads. To price them we re-solved the allocation twice more per capture, once constrained to move only the layer totals and once constrained to move only the split within a layer, each by the same greedy on the same lower convex envelope at the same total budget. The comparison is therefore of the constraint, not of two implementations of a greedy. This is not a new instrument: the same decomposition is computed by `granenv.py` over six arms and both normalisers, and handed the same ladder the two agree to about a point (layer 9.3 against 10.5, head 56.0 against 58.1 on the row they share).

Table 6: Each axis priced in this paper's own objective, at $\alpha = 0.99$ under the per-cell normaliser. Every row is on the *same* four-level ladder $\{512, 2048, 4096, 8192\}$, which is the ladder the four captures of this table measure — the axis decomposition has not been re-run on `s0003` and `s0004`, so this table stays at four while the pooled tables are at six; rows on different ladders are not comparable and are not put in one table. *Joint* is the reduction in aggregate tail risk against uniform at the same budget; *layer* and *head* are each constrained solve's reduction as a share of that joint reduction. *Stranded* is the fraction of the total budget the head-only solve cannot place on the ladder.

Capture	Model	Joint	Layer	Head	Stranded
`s0001`	Qwen3-8B	11.08%	9.3%	56.0%	3.99%
`s0002`	Qwen3-8B	11.03%	21.0%	52.7%	2.86%
`llama_32768`	Llama-3.1-8B	9.00%	6.1%	44.3%	4.10%
`mistral_30720`	Mistral-7B	11.43%	22.8%	57.6%	3.81%

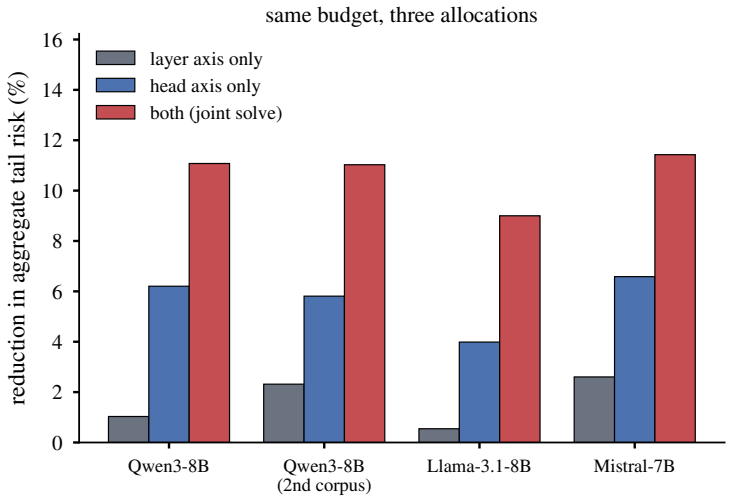

Figure 5: **The head axis carries more than the layer axis on every capture.** The same numbers as Table 6, drawn as absolute reductions rather than shares. For each capture, three bars: the best layer-only allocation, the best head-only allocation, and the joint solve — same greedy, same lower convex envelope, same four-level ladder and the same total budget, re-solved under each constraint, so the bars differ by the constraint and by nothing else. Vertical axis is the reduction in aggregate tail risk against a uniform budget, per cent, at $\alpha = 0.99$. The take-away is the ordering within each capture, which also holds at $\alpha = 0.95$; the bars are not summed, because a sum read off one ladder does not survive the other.

What survives is an ordering and a ceiling: the head axis alone always reaches more of the joint reduction than the layer axis alone, and the layer axis alone never reaches much more than a quarter of it. That holds on every row of Table 6, on the seven-level reading below, and at $\alpha = 0.95$ as well, where the shares run 13.7–24.1% for the layer axis against 47.6–76.8% for the head axis and the ordering holds on all four captures this decomposition covers (the `_a095` files). It is a within-row comparison, so unlike a sum it does not depend on the ladder, which is why it is the form the claim is made in.

What does *not* survive is any statement about the two shares summing. Two of the captures also carry a seven-level ladder, and on it their shares are $20.2/75.3$ and $26.3/71.5$, summing to 95.5% and 97.8% — against 65.3% and 50.4% for those same two captures in Table 6. That gap is a property of the ladder, not of the axes. A constrained solve takes coarser steps than the joint one — the layer-only solve moves eight cells at once — so on a short ladder it strands budget it cannot place, about 4% of the total here against none for the joint solve. A sum read off one ladder would not survive the other, so we do not read it. Data: the `axispredict_*` files in `data`, with the four-level rows in the `_lv4` variants.

An external check. Every constrained solve above is ours; Ada-KV's measured allocation is not, and it happens to land on the same axis. It gives every layer the same total (16384 tokens, spread $1.00\times$) and varies only inside a layer, so it *is* a head-axis allocator in exactly the sense of this section, produced by someone else's method rather than by our own constraint. Our head-only arm shares its axis, total, press, kernel and scorer, so the two differ only in *what they optimise*, and §6.2 reports that difference at $+3.66$ [+2.64, +4.67] against the -1.37 [-2.33, -0.43] Ada-KV's own allocation costs over uniform, and the two-axis arm $+3.75$ $[+2.74, +4.76]$. Nothing about the pair is arranged by us except the choice to compare them. It is *not* the cleanest form of this paper's claim, which an earlier draft called it: both of our arms here are built by `mkalloc.py`, whose per-cell curves are standalone CVaR, so the pair separates two allocators at one objective. The claim's own comparison is §F.2.

Task level. Both one-axis arms exist at *both* context lengths, and they separate two claims that looked like one. **The layer axis alone is null twice**: -0.11 [-0.93, +0.69] at 32768 and -0.05 [-0.73, +0.62] at 4096, so the comparison a layer-wise allocator has to answer — ours against that arm — is $+2.49$ [+1.53, +3.47] and $+4.26$ [+3.44, +5.10]. **The head axis carries most of the rest at both**, $+2.30$ [+1.35, +3.22] and $+3.29$ [+2.47, +4.14], clearing the task-type interval too. **What changes with context length is whether the two are complementary**: the layer axis on top of the head axis buys $+0.08$ [-0.60, +0.75] at 32768 — nothing — and $+0.92$ [+0.25, +1.62] at 4096, clearing the question interval though not the type one ([-0.24, +2.20]). Where a cell holds a quarter of the tokens, which layer they go to starts to matter. All four arms here are built by `mkalloc.py`, so this decomposition is of the *axes* and not of the objective (§6.6).

G.10 Published allocations at every budget

Priced again at 1024 and 4096, all *fourteen* scorable published cells are negative. This is the table §6.2 reads. Its grids extend the budget ladder to 12288, because the published vectors spread harder than ours and a third of the cells would otherwise be unscorable; they are therefore not comparable to Table 6, which does not extend it. A reviewer is entitled to ask whether 2048 is the budget at which the ordering happens to hold, so the middle row of each block is the operating point of §6.1 and the other two price it again at 1024 and 4096. Ada-KV moves the head axis only, CAKE and layer-DefensiveKV both axes, and ours both; uniform is 0% by construction and is SnapKV's own allocation.

Reading a published shape at a different budget moves the magnitude and not the sign either: Ada-KV read at $b = 4096$ instead of 512 goes from -16.07% to -7.01% and CAKE from -29.33% to -32.65%, both still negative. The four refusals are CAKE on Qwen3-8B at 4096 (32 cells above the grid), Ada-KV on Qwen3-8B at 4096 (9 cells above the grid), layer-DKV on Llama-3.1-8B at 4096 (3 cells above the grid), Ada-KV on Llama-3.1-8B at 4096 (1 cell above the grid).

One thing fell out of this that we did not go looking for: our own reduction grows with the operating budget on both models — 8.77%, 13.29%, 17.34% on Qwen3-8B and 9.23%, 10.69%, 13.63% on Llama-3.1-8B. More budget to place is worth more, which is the opposite of the intuition that allocation matters most when budget is scarcest. Three points on two models is a trend and not a law, and we report it as one.

Table 7: Reduction in aggregate tail risk against a uniform budget at the same total, per operating budget, $\alpha = 0.99$, per-cell normaliser, shapes read at $b = 512$. † marks an upper bound — that vector puts cells below the lowest measured budget, where the envelope understates their cost — and every bound here is negative, so the ordering holds a fortiori. Every one of the fourteen scorable published cells is negative and every sign repeats at $\alpha = 0.95$; none of these methods claims to optimise this objective. *n/s* is a refusal rather than an omission: rescaling that vector to the higher budget puts cells above the highest measured level, and above the grid the envelope yields nothing that is not an invention.

Model	b_0	Ours	CAKE	layer-DKV	Ada-KV
Qwen3-8B	1024	$+8.77\%$	$-16.98\%^\dagger$	$-2.93\%^\dagger$	$-10.50\%^\dagger$
	2048	$+13.29\%$	$-29.33\%^\dagger$	$-3.80\%^\dagger$	-16.07%
	4096	$+17.34\%$	n/s	$-1.55\%^\dagger$	n/s
Llama-3.1-8B	1024	$+9.23\%$	$-5.66\%^\dagger$	$-2.62\%^\dagger$	$-1.42\%^\dagger$
	2048	$+10.69\%$	$-10.71\%^\dagger$	$-6.73\%^\dagger$	-5.28%
	4096	$+13.63\%$	$-15.40\%^\dagger$	n/s	n/s

G.11 Calibration size

The solve is milliseconds; the capture is 2048 decode queries over a 32768-token context, and it is the part a reader pays before using any of this. So the allocation is rebuilt from the first N queries and scored on all 2048: each row is what a shorter calibration would have bought, on data it did not see.

	Qwen3-8B		Llama-3.1-8B	
Calibration queries	held out	of ceiling	held out	of ceiling
64	5.96%	53.8%	3.47%	38.5%
128	8.54%	77.1%	5.69%	63.3%
256	9.53%	86.1%	7.13%	79.2%
512	10.06%	90.8%	8.11%	90.1%
1024	10.35%	93.5%	8.94%	99.3%
2048	11.08%	100.0%	9.00%	100.0%

The queries are taken *in order*, not sampled. They are a contiguous decode over one passage, so a random subset would let the small-N rows draw on the whole of it and would make a short calibration look easier than the situation it stands for. Taken in order, 512 queries — a quarter of the capture — retain 90.8% and 90.1% of the ceiling on the two models, and the final four-fold is worth about a point on each.

Two limits. This does not say 512 is enough on a model or corpus not measured here: two models agreeing on the shape of a curve is two points. And the quantity held out is the reduction in aggregate tail risk on the calibration corpus, which is not the task-level gain on another one; §6.6 measures that, and it is a different question.

G.12 Scoring policies

Table 7 scores every allocation on one selection policy's curves, SnapKV's, which leaves open whether the ordering is a property of the allocations or of SnapKV. Swapping the scoring rule at the same operating budget answers that.

The one non-negative cell is the reading most favourable to the method that produced it — a layer-wise allocation judged by its own selection rule's curves — and even there it only draws. Because it is an upper bound the data does not permit "it wins" and does not permit "it loses" either, so we report it as parity and no more. The fourteen-cell result of Appendix G.10 is unaffected: those cells are all scored on SnapKV's curves, so this varies a different axis.

Table 8: Reduction in aggregate tail risk against a uniform budget at the same total, $b_0 = 2048$, $\alpha = 0.99$, per-cell normaliser, by the policy whose curves the allocations are scored on. † marks an upper bound. Seventeen of the eighteen published cells are negative and ours is positive in all six; the exception is layer-DefensiveKV scored against DefensiveKV's own curves, which is a bound at $+0.54\%$ and so reaches at most parity.

Model	Scored on	CAKE	layer-DKV	Ada-KV	Ours
Qwen3-8B	SnapKV	$-29.33\%^\dagger$	$-3.80\%^\dagger$	-16.07%	$+13.29\%$
	DefensiveKV	$-18.15\%^\dagger$	$+0.54\%^\dagger$	-14.75%	$+23.91\%$
	CriticalKV	$-22.02\%^\dagger$	$-0.71\%^\dagger$	-14.69%	$+14.30\%$
Llama-3.1-8B	SnapKV	$-10.71\%^\dagger$	$-6.73\%^\dagger$	-5.28%	$+10.69\%$
	DefensiveKV	$-6.78\%^\dagger$	$-4.80\%^\dagger$	-3.57%	$+12.97\%$
	CriticalKV	$-7.66\%^\dagger$	$-5.65\%^\dagger$	-4.60%	$+10.32\%$

G.13 Necessity of the envelope

The argument has two halves: β says which cells matter to the aggregate tail, and the lower convex envelope says how fast each cell's risk falls per token. A reviewer is entitled to ask whether the second half earns its place, so we removed it. Same greedy, same envelope for scoring, same operating budget and grid; the only change is the priority signal, which becomes a ranking of cells by β filled in order.

Model	Envelope greedy	β ranking	Cells agreeing
Qwen3-8B	$+11.08\%$	-20.96%	31.9%
Llama-3.1-8B	$+9.00\%$	-20.69%	30.1%

Ranking by β alone is not merely worse than the full method: it is about twenty points worse than giving every cell the same budget. It lifts high-β cells to the top of the ladder whether or not their curves are steep there, and starves the rest at the floor — which is the Jensen mechanism of §6.2 applied to our own signal. The criticism this paper makes of the field lands on its own contribution signal if the signal is used without the curve.

Scope. This tests *one* way of using β — rank the cells and fill them in order. A gentler rule, β-proportional rather than β-ordered, might do better and nothing here speaks to it. What the result does say is that the contribution signal alone does not license an allocation.

G.14 Comparison with distributionally robust optimisation

The obvious move for an estimate resting on five tail points is to replace the empirical law with the worst case in a divergence ball whose radius shrinks with the sample, and relative-entropy DRO has a large-deviations optimality argument behind it in general. It is the wrong move here, and the evidence is in hand rather than hypothetical. Robustifying CVaR against sampling error is, by CVaR's dual representation as a worst case over a likelihood-ratio ball (Artzner et al., 1999; Rockafellar & Uryasev, 2000), equivalent to evaluating at a *higher* effective tail level. Our own sweep over the fitting level runs the other way: the configuration chosen on validation lowers the fitting level below the scoring level in 56 of the 92 rows of the four captures this sweep was run on, and raises it in none. It has not been re-run on the two captures added later, so the count is scoped to the rows it covers rather than quietly restated over a row set it never saw. Robustness and estimation variance pull in opposite directions in this problem, and variance wins — so a method whose whole content is extra conservatism cannot help. What helps is either more information per sample, which is what the regression family buys, or an honest statement of what the sample supports, which is §6.4.

G.15 Interval construction

An i.i.d. bootstrap over queries is void here: the tail arrives in episodes (§5), so a resample that treats queries as exchangeable resamples episodes several times over and understates the spread — which is exactly the failure that started that investigation. The instrument that replaces it is a moving block

bootstrap on the test window, and the block length is not a free parameter: it is $\lceil 3/\hat{\theta} \rceil$, three mean cluster lengths at the θ measured on that same series (§5), which comes out between 3 and 15 queries across rows. The rule of thumb $\lceil Q^{1/3} \rceil = 11$ is carried alongside it as a check, as a guard: a finding that changed with the choice between two equally defensible rules would not be a finding. Across the 140 rows the two rules disagree on one. The statistic bootstrapped is the paper's own claim: the extra reduction the contribution-aware allocation achieves over the separable optimum, under the fixed recipe of §6.4 rather than a per-row selection, so these are a recipe's error bars and not a search's.

At 90% and 2000 resamples the point estimates run from -21.55 to $+7.44$ pp with median $+1.92$. The interval lies entirely *above* zero in 104 of 140 rows, entirely *below* it in 6, and straddles it in 30.

That three-way split replaces a two-way one, and the reason is worth stating. Through four captures this appendix reported "the interval excludes zero in 78 of 92 rows". The count was right and the description was not: the flag it came from is $\mathrm{lo} > 0$, a one-sided test, and it read as a two-sided one only because no row in those four captures had an interval lying wholly below zero. Six rows now do, so the two readings have come apart and the sentence has to say which it means. A description that is accidentally true of the data in hand is the failure mode this paper keeps finding in its own drafts.

The six negative rows are not scattered. Every one of them is on `s0004` and every one is at $\alpha = 0.99$: at that tail level the capture gives 2 rows above zero, 6 below and a median of -2.26 pp, while at $\alpha = 0.95$ the same capture gives 6 above, *none* below and $+0.57$. Pooled, $\alpha = 0.99$ is $46/70$ above and 6 below against $58/70$ and 0 at $\alpha = 0.95$. The largest single loss, StreamingLLM at -21.55 pp, is the same arm that gains $+1.6$ to $+6.3$ pp on the five other captures and is null rather than catastrophic at $\alpha = 0.95$ on this one ($+0.28$).

This is the paper's own thesis arriving as a cost. §5 argues that the $\alpha = 0.99$ tail is a five-sample estimation problem; $\alpha = 0.99$ is accordingly where the method has the most to win — a larger median gain — and the only place it is ever reliably wrong. The honest summary is that the extreme tail is where this allocation pays and where it can also fail, and that one capture in six is on the wrong side of that trade. Data: `data/blockci_*.json`.

The distribution-free alternative is worth stating because its cost has no free parameters. A Dvoretzky–Kiefer–Wolfowitz bound on the tail integral is vacuous unless the held-out sample exceeds $\ln(2/\delta)/2(1-\alpha)^2$ queries, which at $\delta = 0.1$ is 599 at $\alpha = 0.95$ and 14,980 at $\alpha = 0.99$. Our validation window holds 512. Widening the split would buy the bound at 0.95; at 0.99 it is two orders of magnitude of decode steps away, and no resampling scheme recovers what that inequality says is not in the sample.

G.16 The extremal index

We found the clustering by chasing a failure. A 90% bootstrap upper bound on the aggregate tail risk, built on validation, failed to cover the realised test value in *every* row, by roughly a factor of two. A bound that never covers is not a noisy bound; it is a bound whose exchangeability assumption is false.

Episodes. The three windows are consecutive segments of one generation, so the first candidate was a trend. Measured at a fixed uniform allocation over eight consecutive windows, that is not what is happening: the median aggregate error is flat — 127.6 at the first and 127.2 at the eighth on `snapkv` — while one window of the eight carries a $\mathrm{CVaR}_{0.95}$ twice its neighbours'. The level does not move; the tail arrives in *episodes*, and whether a window contains one decides its tail estimate.

Two estimators. We report two together, because a single one is a choice: the runs estimator of Smith & Weissman (1994), which declares a cluster over after eight consecutive non-exceedances, and the intervals estimator of Ferro & Segers (2003), which needs no run length; a more recent estimator of the same quantity, which we did not run, is Northrop (2015)'s. At $\alpha = 0.95$, where the estimate rests on 102–130 exceedances of the full 2048-step decode — not of the 512-query calibration window, which is why these counts are four times what that window would give — the four policies that score by attention have per-capture medians of 0.234–0.450 and the two that ignore attention — K-Norm and StreamingLLM — have 0.372–0.545; the ordering holds on all four captures this survey covers and under both estimators, which differ in level but not in order. The intervals estimator sits about 0.18 higher in level than the runs estimator (medians 0.439 against

0.668 for the attention-scored and non-attention policies respectively), so we report the level as a range across the two rules and treat only the ordering as a finding. For the attention-scored policies this leaves 23–55 effective independent tail events out of 102–128 nominal exceedances.

The unquoted level. At $\alpha = 0.99$ the estimate rests on 20–32 exceedances and is too noisy to carry a claim: the separation survives on the two Qwen captures (0.31–0.35 against 0.63–0.78) and reverses or vanishes on Llama and Mistral (0.87 and 0.94 for the attention-scored policies). That is the tail level where a clean separation would have been most useful, which is exactly why we decline to quote it as one. We also do not have a sharper statement than the direction of the effect on the estimator: we have not estimated the effective sample size of β_i itself, only argued that clustering inflates its variance without collapsing its effective sample to the number of episodes, as it does for an order statistic. Data: `data/extremal_*.json`.

G.17 Unchanged quantities

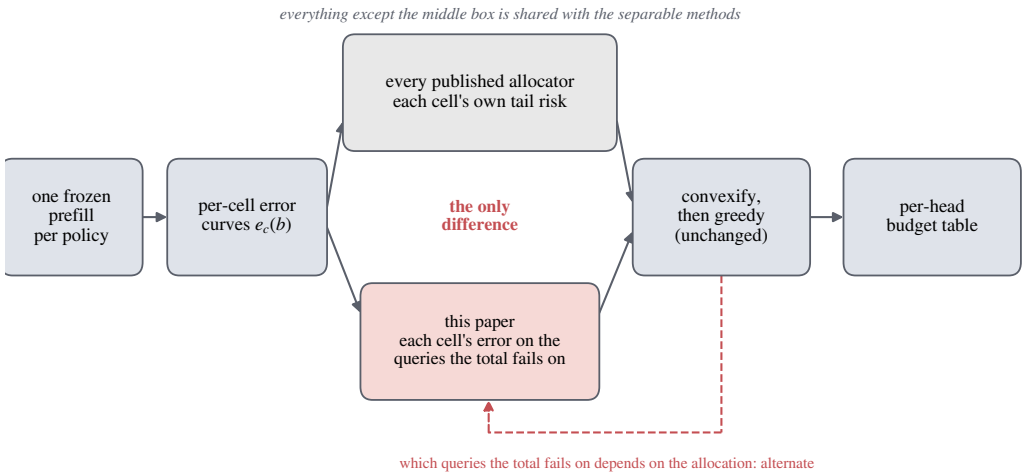

Figure 6: **What changes, and what does not.** A schematic of the pipeline, left to right. The grey boxes — the frozen prefill, the per-cell error curves, the convexification and the greedy — are shared with the separable methods. Only the middle branch differs: the upper alternative scores a cell by its own tail risk, which is what every published allocator does, and the lower one, ours, scores it by its error *on the queries where the total is failing*, its Euler contribution (4). The dashed arrow is the loop that branch forces: which queries those are depends on the allocation, so estimate and solve alternate (§5.1). The take-away is the size of the change — one substituted curve, and no new solver.

G.18 The asymptotic regime

§4.1 reports the conclusion; these are the measurements behind it, over 48 capture–policy–normaliser combinations. The Hill index at $k = 64$ sits well below 0.5 on the aggregate and at the median cell alike, so the variance is finite: mildly heavy-tailed, not Pareto-type. The index itself is the companion measurement submission's result and is not restated here. No single cell ever crosses the aggregate's 99th percentile — zero of 256–288 cells, in all 48 combinations. The largest single cell holds 2.1–10.7% of the tail total at $\alpha = 0.99$ and the largest ten hold 14.5–40.8%, against 0.37% and 3.7% if the tail were spread flat. Feller's single-big-jump principle, $P(\sum_i X_i > x) \sim \sum_i P(X_i > x)$, is what would have made the aggregate tail separable again over a common absolute threshold; none of the above is consistent with it.

G.19 Model versus corpus in the allocation

Every allocation in this paper is calibrated on one corpus and applied elsewhere, so it is worth measuring what it is a property of. Two captures share a model and differ only in the book —

Qwen3-8B on Moby-Dick and on War and Peace, same 36 layers, 8 KV heads, ladder and recipe — so their two allocations can be compared cell by cell, with a random permutation of one as a negative control. What the comparison finds is a middle: an allocation is only about half a property of the model, and half a property of the text it was fitted on, so neither reading on its own is safe.

Comparison	Spearman	Pearson	TV distance
Same model, different book	$+0.573$	$+0.522$	0.235
Against a random permutation	-0.140	-0.090	0.412
Layer totals only	$+0.663$	$+0.709$	—

The allocation agrees across corpora far more than chance and nothing like exactly: roughly half a property of the model and half of the text it was calibrated on. We therefore do not claim the allocation transfers as a property of the model, and no claim of that kind appears anywhere in this paper. What the number supports instead is a robustness statement — the same allocation moves RULER-4096 on two model families with both intervals clearing while being only half determined by the model, so the task-level gain does not depend on getting the allocation right in detail.

Two things follow that we did not pursue. A corpus-matched allocation might beat everything measured here, and we do not take that direction; the gap is real and unexplored. And the axis that transfers better is the one worth less: layer totals agree at Spearman 0.663 against 0.573 for the full cell-level allocation, while §6 prices the layer axis at a fifth to a quarter of the joint reduction and the head axis at most of the rest. We have no explanation for that and prefer to record it rather than invent one.

G.20 Hedge provenance

Partly each. Across the six eviction policies the β vectors agree at Spearman 0.51 in the median (range 0.31–0.95) and the hedge sets overlap at Jaccard 0.17–0.30 against 0.10–0.11 expected by chance — above chance, far from identical. So β must be measured per policy and cannot be characterised once and reused. What is stable is depth: hedges cluster in the first third of layers ($31.8/17.2/9.5$ and $40.3/11.8/8.0$ of 36), on every policy and both captures where we measured it.

G.21 The surrogate result

The per-row and per-capture view of the objective-side result is Figure 2 in the main text; the numbers behind each marker are in `rcshrink_*.json`, one file per capture.

G.22 Estimator selection

Of the 140 validation-selected configurations, all six regression variants are chosen somewhere (12–40 times each), the bagged tail mean twelve times, and the cross-validated blend ten. The cross-validated weight on the linear estimator has median 0.50 and spans the full $[0, 1]$. The bias–variance trade the family of §5.1 is built around is therefore visible in the selections rather than asserted in the text.

G.23 Published allocations as matched arms

Both vectors are the ones the allocators themselves produced at this operating budget, read off their retained caches by `mkpub.py` and written into the press configuration our own arms use, so the arms differ from `uniform_control` in the budget vector and in nothing else. The script was validated before use by rebuilding Ada-KV's allocation and reproducing, exactly, the press digest of the arm already reported in §6.6; an instrument that cannot reproduce a published number is not ready to produce a new one.

allocation	layer spread	per head	RULER-4096 vs uniform
layer-DefensiveKV	$8.86\times$	125–28595	$+2.07$ [+1.37, +2.80]
CAKE	$61.52\times$	157–9659	no score: $1300/1300$ refusals

CAKE's is not a bad score, it is the absence of one, and the distinction is the reason the harness carries a failure gate that runs before any mean is taken. At a layer spread of $61.5\times$ the thinnest layers keep under 1% of their tokens, every one of the 1300 questions came back as a refusal, and the gate refused the arm. Reporting the arm's mean of 0.48 as CAKE's RULER score would have been the single most misleading number this project could have published, and it is exactly the number a pipeline without that gate would have printed. What the run licenses is narrow and we state only that: at this operating point, in this harness, CAKE's own vector does not execute.

G.24 Non-allocators

Three neighbouring literatures compress a KV cache and are sometimes described in the same breath as budget allocation. They are outside §4's scope, and it is worth saying which and why, because the argument turns on *what* is being divided.

Quantization allocates precision rather than capacity. KIVI (Liu et al., 2024) quantizes keys per channel and values per token; KVQuant (Hooper et al., 2024) derives non-uniform datatypes per layer. That per-layer step is an allocation, but of a datatype, not of how many tokens a layer keeps, so a cell's cost curve over budget — the object every claim here is about — does not exist for it.

Query-aware selection allocates per query rather than per cell. Quest (Tang et al., 2024) scores cache pages against the current query and loads the top k; InfLLM (Xiao et al., 2024a) looks up context blocks from an external memory. Both choose *which* tokens to attend to at each step, not a standing per-head budget, so there is no fixed allocation to score. MInference (Jiang et al., 2024) is the closest of the three: it does assign something per head, offline — but it assigns a sparse-attention pattern for prefill compute, not a decode-time cache budget.

One method in the merge family does fall inside the scope. D2O (Wan et al., 2025) carries a layer-level component that varies how aggressively each layer evicts, which is a per-layer budget allocation of exactly the shape (1) describes, alongside a token-level merge that is not. We did not score it: our harvest records the budget each retained cache actually used, and D2O is not among the six policies of §6.1. Its layer-level half belongs in Table 9's class; whether it behaves like the rows we did score is untested here.

H Extended related work

H.1 Per method

Risk attribution. None of the machinery in §4 is ours: coherence and the comonotone equality condition are Artzner et al. (1999)'s (see also McNeil et al., 2015; Föllmer & Schied, 2016), the uniqueness result this paper leans on hardest is Tasche (1999)'s with Denault (2001)'s game-theoretic reading, and the optimisation form of CVaR behind §5.1 is Rockafellar & Uryasev (2000)'s.

Table 9: Every published budget allocator we were able to read optimises a sum over cells of a per-cell functional of that cell's own error. *Axis* is the level the method divides a budget over; f is the per-cell statistic it sums; *Objective* is what that sum comes to. X_i is cell i's error and $Z = \sum_i X_i$ the aggregate. Only the last row's objective depends on the joint law of the X_i — it conditions on the aggregate exceeding its own α-quantile — and that is the whole difference this paper is about.

Method	Axis	Per-cell functional f	Objective
Ada-KV	head, within layer	attention-output loss bound	$\sum_i f(X_i)$
CAKE	layer	dispersion, temporal variance	$\sum_i f(X_i)$
DefensiveKV	head or layer	worst case over a window	$\sum_i f(X_i)$
LU-KV	head	convexified cost curve	$\sum_i f(X_i)$
This paper	head	—	$\sum_i \mathbb{E}[X_i \mid Z > \mathrm{VaR}_\alpha(Z)]$

Ada-KV (Feng et al., 2025) introduced head-wise allocation and set the pattern the rest of this literature follows: it minimises an upper bound on the per-layer attention-output eviction loss, pooling head scores within a layer, and its own paper concedes that a lower upper bound does not strictly guarantee reduced loss. The objective is a sum over heads of each head's own loss.

DefensiveKV (Feng et al., 2026a) argues that mean aggregation is fragile in extreme cases and replaces it with a worst-case risk over an observation window, in a head-wise and a layer-wise variant. That worst case is taken *per cell* and then summed, which is exactly the right-hand side of (3) — the comonotone corner. The better the per-cell tail statistic, the tighter that corner is estimated and the further it still sits from the aggregate it bounds.

CAKE (Qin et al., 2025) allocates across layers by a preference score built from attention dispersion and temporal variance, again a sum over layers of a per-layer statistic, and closes by proposing head-level allocation as future work. It is also purely layer-wise, so the ceiling of Appendix G.9 applies to it: on this paper's objective the layer axis alone reaches a fifth to a quarter of what the two axes reach together.

LU-KV (Tang et al., 2026) relaxes the head-budget problem onto the convex hull by isotonic regression and solves it with a marginal-gain greedy, which is exactly optimal for its own objective under Fox's condition (Fox, 1966; Federgruen & Groenevelt, 1986; Katoh et al., 2013). Convexifying each cell's curve buys that condition's antecedent, which is what makes an exactly-solved separable optimum available as a baseline at all. Federgruen & Groenevelt (1986) give necessary and sufficient conditions for that greedy to be optimal. The literature adopted the sum for this tractability rather than for robustness, and §4's point is not that the tractability is illusory but that the quantity being made tractable is the wrong one.

Four further allocators appeared on this axis while this work was in progress — LKV (Zhou et al., 2026), which learns head budgets end to end; ReasonAlloc (Liu et al., 2026), which sets them hierarchically at decoding time; HeteroCache (Shi et al., 2026), which allocates on a stability statistic; and LAVa (Shen et al., 2025), layer-wise with dynamic head budgets — together with CONF-KV (Li & Miao, 2026). We name them and stop there rather than characterise an objective we have not read. What can be said without reading them is conditional and worth stating in that form: §4 applies to any allocator whose objective is a sum over cells of a per-cell functional of that cell's own error, whatever the functional, and does not apply to one whose objective is not. Whether each of these five falls inside that class is unverified here.

A companion measurement submission reports the wider programme these captures came from, including the distribution of realised per-query error across policies and budgets that the eviction literature does not report; nothing in the argument of this paper depends on it. The two share captures, and what else they share is measured rather than asserted: `overlap.py` intersects the two sources on every build and reports the shared word n-grams and the shared numeric tokens. An earlier draft of this sentence claimed the two shared no reported result, and that was false — a fourteen-line table stood in both — which is why the claim is now a measurement with a gate behind it and not a promise. What is shared today is the condition measurement of Table 5: the range of β_i and the minimum of the per-row Spearman are the same numbers in both, because both are extremes over row sets that overlap. Everything else here appears only here. In particular the published-allocator comparison of §6.2 — every figure in it — appears only here: an earlier draft of the companion carried the same table, and it was removed from that paper rather than from this one, because it is a contribution here and was a supporting caveat there. The overlap was found by intersecting the numeric tokens of the two compiled documents rather than by recollection, and the companion's build now refuses any draft that reintroduces one of those figures.

H.2 The estimator family

Two knobs span the space. The degree of m_i is 1 or 2, where degree 2 adds a quadratic term needing two conditional moments of Z — scalars estimated from Z's own tail, one series rather than n of them. The locality is a quantile $q \in \{0, 0.5, 0.8\}$: m_i is fitted on the queries whose aggregate lies above the q-quantile of Z, so raising q buys locality and spends sample. The family also contains the bagged tail mean and a convex blend of the linear estimator with it, whose weight is chosen by K-fold cross-validation *inside the calibration window* and is therefore derived rather than tuned. Each inner solve is exact for its tail set; the outer loop is the sample-average form of the Rockafellar–Uryasev program (Rockafellar & Uryasev, 2000), whose objective decreases monotonically in the population limit, and we keep the best iterate rather than the last, stopping on a repeat.