BANDIT LIMITED DISCREPANCY SEARCH FOR AGENT HARNESS CONFIGURATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Coding-agent harnesses decompose tasks into DAGs and assign each node to a sub-agent, yet node configurations, including agent type, model, reasoning effort, and turn budget, are typically fixed by an initial language-model-generated plan and never revised. We formalize this limitation as the *agent-DAG configuration selection* problem and introduce BLDS-AF, a discrepancy-limited search algorithm that explores the Hamming ball around the initial plan. BLDS-AF replaces the usual independent-arm bandit with a factored surrogate consisting of a ridge-regression posterior over node-level and edge-level configuration indicators, weighted by compute allocation. A structured group prior allows the surrogate to rely on an additive representation when data are scarce and progressively activate interaction terms as evidence accumulates. Consequently, every evaluation is pooled across configurations rather than treated as evidence about one configuration alone, and a composed candidate from the surrogate's own argmax is evaluated like any other. Across fourteen baselines, including Hyperband, TPE, evolutionary search, UCT, and the deployed greedy cascade, BLDS-AF consistently achieves superior performance. It also substantially outperforms plan-local coordinate descent, with gains that increase monotonically as the configuration space grows and persist across all tested interaction regimes. On NAS-Bench-Macro, BLDS-AF surpasses every plan-agnostic competitor with confidence intervals that exclude zero, while on a DAG of live multi-turn coding agents it reaches zero normalized regret under a modest evaluation budget. The improvement stems from statistically pooling information across related configurations, and we derive a bias-variance diagnostic that predicts a priori when such pooling is likely to be beneficial.

1 INTRODUCTION

Modern coding agents rarely solve complex tasks within a single conversation. Instead, they decompose work into a directed acyclic graph (DAG) of subtasks and assign each node to a specialized sub-agent with its own context, prompt, tools, and model. The resulting harness is therefore not a single policy but a scheduler over a large configuration space. For every node, it must select an agent type, foundation model, reasoning effort, turn budget, and tool set. In the harness we study, these choices span multiple agent classes, hundreds of models from fifteen providers, seven reasoning levels, and an unbounded turn budget. Consequently, a DAG with m nodes and k candidate configurations per node induces k^m possible harness configurations.

Despite the size of this search space, deployed coding-agent harnesses perform essentially no search. They generate a plan once, derive per-node agent assignments from it, apply a global model and turn budget, and execute the DAG greedily in topological order. There is no multi-fidelity evaluation, backtracking, alternative-arm exploration, or credit assignment across decisions. If a node fails, it is reset to pending and execution of its entire descendant subtree halts (Appendix A documents these properties from released implementations). In effect, current harnesses follow a strategy of evaluating a single configuration at full fidelity and committing to it, a paradigm long abandoned in the broader machine learning literature on pipeline optimization.

Our starting point is that this problem is not merely *analogous* to algorithm selection in a fixed machine-learning pipeline—it is a direct instance of the same problem, with the graph structure fixed and the per-stage choices left free. This observation makes Bandit Limited Discrepancy Search (BLDS) (Kishimoto et al., 2022) the natural candidate. Specifically, BLDS addresses exactly this setting by combining limited discrepancy search (Harvey and Ginsberg, 1995; Korf, 1996) which concentrates exploration near a strong incumbent, with multi-fidelity evaluation, which avoids spending full compute (or cost) on unpromising candidates.

Three aspects of the adaptation are specific to agent harnesses and drive the rest of the paper. First, the plan serves as the heuristic. Limited discrepancy search traditionally relies on a value-ordering heuristic, which is often unavailable in pipeline optimization and absent from BLDS itself. In contrast, a coding-agent harness provides a strong heuristic for

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.

free through the language-model-generated plan. The resulting Hamming ball of radius θ around that plan becomes an informative prior over promising configurations and motivates discrepancy-limited search over exploration from scratch. Second, fidelity corresponds to compute rather than data. As a result, the theoretical guarantees must be re-derived, and monotonicity becomes an empirical assumption rather than a property inherited from subsampling. Third, failures are informative. Because execution is local to the DAG, a failure at node f immediately eliminates every configuration that matches the incumbent on $\mathrm{anc}(f) \cup \{f\}$, providing pruning signals at essentially no additional cost.

Three things about agent harnesses shape everything that follows.

First, the plan is the heuristic. Limited discrepancy search needs a value-ordering heuristic, and pipeline optimization usually doesn't have one—BLDS itself doesn't use one either. But a coding-agent harness gives you one for free: the language model's own plan. That plan defines a neighborhood of configurations that differ from it in only a few places, and that neighborhood is where the good configurations tend to live. So instead of searching from scratch, search near the plan.

Second, fidelity is compute, not data. When you scale up compute for an agent, you're not drawing more samples from the same distribution—you're changing what the agent actually does. That means the theoretical guarantees don't carry over automatically, and you can't assume that a cheap run estimates the same thing as an expensive one. You have to re-derive the bounds and treat monotonicity as something to test, not something you get for free.

Third, failures tell you something. Execution is local to the DAG: a node's outcome depends on its own configuration and its predecessors. So when a node fails, you immediately know that every configuration matching the current one on that node and its ancestors will fail there too. That's a pruning signal you get at no extra cost, just from observing the failure.

Contributions. We characterize the execution model of a deployed coding-agent harness and formalize *agent-DAG configuration selection*, an optimization problem driven by token cost. We introduce BLDS-AF, a discrepancy-limited search method that explores the neighborhood of a language-model-generated plan using a factored surrogate over node-level and edge-level configuration features. We establish a ball-invariance lemma and derive a θ-local elimination bound for the bounded-discrepancy setting left open by Kishimoto et al. (2022). To evaluate the central hypothesis, we first analyze all 6,561 exhaustively evaluated configurations of NAS-Bench-Macro (Su et al., 2021), and then apply the same algorithm to NAS-Bench-Macro, an ADCS simulator benchmark comparing against fourteen baselines, and a DAG of live coding agents. We show that BLDS-AF consistently outperforms the deployed scheduler and all competing methods. However, our experiments reveal that these gains arise primarily from statistical pooling across related configurations, rather than from several mechanisms inherited from BLDS. In particular, a direct independent-arm adaptation is outperformed by plan-local coordinate descent, adding a multi-fidelity ladder significantly degrades performance, and failure-based pruning provides no benefit when locality assumptions are violated. We further find that the θ-local elimination bound has little predictive value in practice, that performance is largely insensitive to θ, and that multi-fidelity allocation effects disappear once top-rung budget is controlled. We therefore identify statistical pooling as the key source of improvement and explicitly document which initially plausible design choices do not withstand empirical scrutiny. The Appendix contains additional details and analyses.

2 Agent-DAG Configuration Selection

Definition 1 (ADCS instance)**.** An *agent-DAG configuration selection* instance is a tuple $(G, \{\mathcal{A}_i\}, \pi^0, \Lambda, R)$ where $G = (V, E)$ is a DAG with $V = \{1, \ldots, m\}$ in topological order and $(u, v) \in E$ iff v is blocked by u; $\mathcal{A}_i$ is the finite *arm set* at node i—the available (agent type, model, effort, turn budget, tool scope) tuples— An arm is the full per-node choice—agent type, model, reasoning effort, turn budget, and tool scope—while a configuration assigns one arm to every node of the DAG and a *configuration* is $p = (a_1, \ldots, a_m) \in \mathcal{P} := \prod_i \mathcal{A}_i$ with $L := |\mathcal{P}|$; $\pi^0 \in \mathcal{P}$ is the *plan* the language model emits at decomposition time; $\Lambda = (\lambda_1 < \cdots < \lambda_{\max})$ is a *fidelity ladder* of compute allocations; and $R : \mathcal{P} \to [0, 1]$ is a bounded *verifier*, e.g. the fraction of a held-out test suite passing after the DAG executes under p. Executing p at fidelity λ_ℓ yields $\hat{R}_\ell(p) \in [0, 1]$ and consumes $c_\ell = b\eta^{\ell-1}$, $b, \eta > 1$. Write $q_p^{(\ell)} := \mathbb{E}[\hat{R}_\ell(p)]$ and $q_p := q_p^{(\max)}$ for the expected reward at full fidelity, the quantity competed for, and $\mathrm{anc}(i)$, $\mathrm{pred}(i)$ for the ancestors and immediate predecessors of i.

The deployed harness solves this instance by returning π^0 unmodified, evaluated once at $\lambda_{\max}$. Two derived quantities recur: the *fidelity bias* $\beta_\ell := \sup_p |q_p^{(\ell)} - q_p|$, how far a cheap rung's estimand is from the real one, and the *climb premium* $C(\Lambda) = \sum_\ell c_\ell / c_{\max}$, the cost of escalating one candidate to the top relative to running it there directly, which

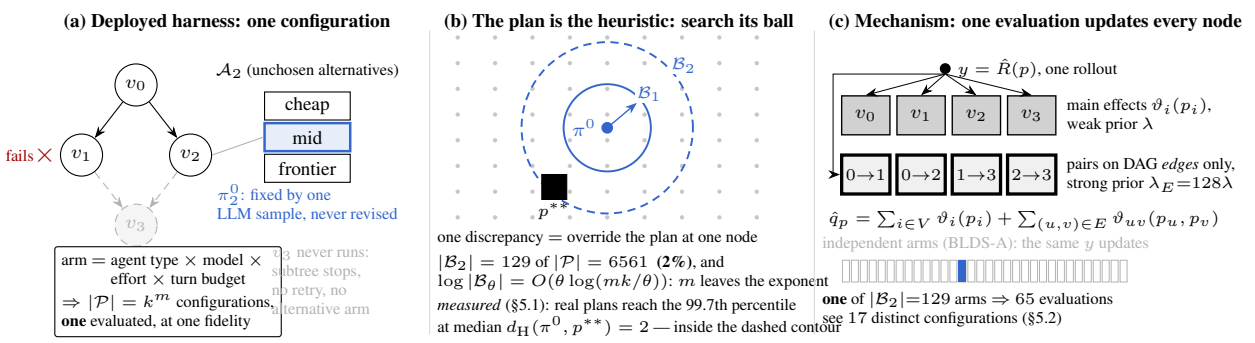

Figure 1: The setting, the prior and the mechanism. **(a)** A deployed harness reads a per-node arm—agent type, model, reasoning effort, turn budget—off a single language-model plan π^0, executes the DAG greedily at one fidelity, and abandons a failed node's descendants; one of k^m configurations is ever evaluated. **(b)** BLDS-AF treats π^0 as the value-ordering heuristic limited discrepancy search presumes, so overriding the plan at one node is one discrepancy and the search is confined to the Hamming ball $\mathcal{B}_\theta(\pi^0)$—2% of the space at $\theta = 2$, and the ball in which real plans actually put the optimum. **(c)** What makes it work is not the ball but the surrogate: one rollout updates the coefficient of the arm at *every* node, plus the pair on every DAG edge, where an independent-arm bandit would credit one configuration of 129.

is the break-even condition for multi-fidelity screening. Appendix F.6 reports that, contrary to our own earlier claim, $C(\Lambda)$ predicts nothing we can measure.

Definition 2 (Discrepancy ball)**.** For $p \in \mathcal{P}$ let $d_H(p, \pi^0) = |\{i : p_i \neq \pi_i^0\}|$. The *discrepancy ball* of radius θ is $\mathcal{B}_\theta(\pi^0) = \{p \in \mathcal{P} : d_H(p, \pi^0) \leq \theta\}$; for $S \subseteq V$ the *restricted ball* $\mathcal{B}_\theta(\pi^0; S)$ additionally requires $p_i = \pi_i^0$ for all $i \notin S$.

Deviating from π^0 at one node is one discrepancy, in exact correspondence with a right turn in limited discrepancy search. The interpretive claim of this paper is that the correspondence is substantive rather than formal—the plan is a genuine value-ordering heuristic with genuine accuracy, so radius-θ search for small θ is where the good configurations are. Section 5.1 measures this for four production planners and Appendix F.10 what happens as it is made false by construction.

Objective. Because each evaluation is paid for in tokens rather than wall-clock epochs, raw success rate is the wrong headline. With $C(\mathcal{S})$ the cumulative compute a strategy consumes and $\hat{p}(\mathcal{S})$ its returned configuration, we report *success-at-cost*

$$\mathrm{S@C}(\mathcal{S}, c) \;=\; \mathbb{E}\big[\, q_{\hat{p}(\mathcal{S})} \;\big|\; C(\mathcal{S}) \leq c \,\big], \tag{1}$$

the expected verifier reward attainable under a compute ceiling c, and its dual, the compute required to reach a target reward: a strategy that finds a better configuration by spending ten times more is not an improvement. Section 5.4 shows this axis, not reward, is where agent-DAG configuration selection actually pays.

3 BLDS-AF: Discrepancy Search with a Factored Bandit

BLDS-AF retains the outer structure of BLDS—an incumbent, a bounded discrepancy budget, and confidence-gated acceptance—and modifies three components: the plan supplies the value ordering, fidelity is compute rather than data (Appendix C.1 re-derives the confidence bounds and records that their noise scale s is load-bearing rather than cosmetic), and evidence is pooled across configurations rather than kept per configuration. The third is the one the experiments forced on us, and it is what makes the method competitive.

The factored surrogate. Let $\phi(p)$ carry mk indicators for the arm chosen at each node *together with* $|E|k^2$ indicators for the arm pair on each DAG edge, where we assume that

$$\hat{q}_p \;=\; \sum_{i \in V} \vartheta_i(p_i) \;+\; \sum_{(u,v)\in E} \vartheta_{uv}(p_u, p_v), \tag{2}$$

Algorithm 1 BLDS-AF at a single fidelity. The ladder variant, RESTART, the structural-failure test and the sign convention are specified in Appendix C.

Require: DAG G, arm sets $\{\mathcal{A}_i\}$, plan π^0, verifier R, radius θ, exploration β, prior Λ, compute ceiling c

$A \leftarrow \Lambda$; $b \leftarrow 0$; $\leftarrow \emptyset$; OBSERVE(π^0); $p^{\mathrm{inc}} \leftarrow \pi^0$
while compute spent $< c$ **do**
  $S \leftarrow \mathrm{anc}(f) \cup \{f\}$ if p^{inc} failed structurally at f, else $S \leftarrow V$ ▷ Prop. 16
  $\leftarrow \{p \in \mathcal{B}_\theta(p^{\mathrm{inc}}; S) : p \notin\} \cup \{\arg\max_p \hat{q}_p\}$ ▷ discrepancy shell + the surrogate's own argmax
  if $= \varnothing$ **then**
    $\leftarrow \{$RESTART()$\}$
  end if
  OBSERVE$\big(\arg\max_{p\in}\ \hat{q}_p + \beta\|\phi(p)\|_{A^{-1}}\big)$ ▷ upper confidence bound over the shell
  $p^{\mathrm{rec}} \leftarrow \arg\max_{p\in} \hat{q}_p$ ▷ recommend by posterior mean, never the raw maximum
  if $\hat{q}_{p^{\mathrm{rec}}} > \hat{q}_{p^{\mathrm{inc}}}$ **then**
    $p^{\mathrm{inc}} \leftarrow p^{\mathrm{rec}}$
  end if
end while
return $\arg\max_{p\in} \hat{q}_p$

function OBSERVE(p)
  $y \leftarrow \hat{R}(p)$; charge $c_{\max}$; $A \leftarrow A + c_{\max}\phi(p)\phi(p)^\top$; $b \leftarrow b + c_{\max}\phi(p)y$; $\hat{\vartheta} \leftarrow A^{-1}b$; $\leftarrow \cup\{p\}$
end function

so a node's contribution may depend on what its predecessor chose. Interactions sit on the DAG's own edges rather than on all pairs—Assumption 5 used as a modelling constraint, so non-adjacent pairs are not estimated at all. Maintain the ridge posterior

$$A_t = \Lambda + \textstyle\sum_{j\le t} c_{\ell_j}\,\phi_j\phi_j^\top, \qquad \hat{\vartheta}_t = A_t^{-1}\textstyle\sum_{j\le t} c_{\ell_j}\,\phi_j y_j, \qquad \hat{q}_p = \phi(p)^\top\hat{\vartheta}_t, \tag{3}$$

weighting each observation by its allocation c_{ℓ_j}, the correct weight under Definition 9. The prior Λ is diagonal but *not* scalar: main effects receive a weak penalty λ and the interaction block a much stronger $\lambda_E \gg \lambda$. That group-wise penalty is what makes Equation 2 safe to adopt unconditionally, letting the model collapse onto the additive one while data is scarce and switch the interactions on as evidence accumulates—which matters because the regimes differ sharply, our simulator instances being 90% additive and real agent runs 19% (Section 5.4). Appendix F.5 selects $\lambda_E = 128\lambda$ and shows a scalar ridge failing in both directions.

Three properties are worth naming. The posterior is updated from *every* observation, so one evaluation is evidence about the arm at every node—the component that carries the method empirically. The candidate set is the discrepancy shell around the incumbent, so the plan prior decides where to look, plus the surrogate's own argmax as a single *composed* candidate, evaluated like any other so a proposal that got additivity wrong is rejected rather than kept. And recommendation is by posterior mean, not the incumbent (after an unproductive round a restart centre the search would never propose) and not the raw observed maximum (the winner's curse).

Whether pooling helps is a bias–variance question with a computable answer. Let $r(p) = q_p - \phi(p)^\top\vartheta^\star$ be the residual of the best additive approximation; for a Hamming-1 pair differing at node i, write $\delta = r(p) - r(p')$ for the *local* misspecification and $\Delta = |q_p - q_{p'}|$ for the gap to be resolved. Direct evaluation, as coordinate descent does it, gives an unbiased contrast of variance $2s^2/c_{\max}$; pooling through Equation 3 gives one of bias δ and variance $O(ks^2/(nc_{\max}))$, so pooling wins whenever $|\delta| \lesssim \Delta$ and $n \gtrsim k$, and loses when $|\delta| \gg \Delta$ however large n becomes (Proposition 6).

The condition involves the *local* residual difference, not the global residual, and that distinction is what makes the method work at all: two configurations at Hamming distance 1 share every interaction term not touching node i, so their residuals are strongly correlated and δ is much smaller than $\|r\|$—which is why a 0.90-R^2 surrogate suffices. It is therefore a pre-flight check rather than a post-hoc explanation, both quantities coming from rewards already paid for (Appendix C.4); our three substrates measure 1.1 (simulator), 11.9 (NAS-Bench-Macro) and 3.4 (live agents), and the additive basis wins on the first and fails on the other two.

4 ANALYSIS

Kishimoto et al. (2022) bound the full-fidelity trainings needed to identify the best pipeline when the discrepancy budget equals the pipeline depth, and state explicitly that the practically-used regime $\theta < m$ "remain[s] an open question." Our

setting forces exactly that regime, so we address it; proofs are in Appendix D. Every configuration the search evaluates provably lies in $\mathcal{B}_\theta(p^{\mathrm{inc}}; S)$ (Lemma 7), which is what licenses replacing the whole space by the ball inside the union bound.

If $|\mathcal{A}_i| \le k$ for all i and $|S| = s$, that ball has

$$|\mathcal{B}_\theta(\pi^0; S)| = \sum\nolimits_{j=0}^{\theta} \binom{s}{j}(k-1)^j \;\le\; (\theta+1)\big(e\,s\,(k-1)/\theta\big)^\theta, \quad \text{so} \quad \log|\mathcal{B}_\theta(\pi^0; S)| = O\big(\theta \log(sk/\theta)\big) \tag{4}$$

(Lemma 8), which is the quantitative content of the plan-as-heuristic idea: the exhaustive space has $\log L = m \log k$, so discrepancy limiting moves m from a multiplier to a logarithm at the price of restricting attention to configurations near the plan.

The concentration premise needs an observation model, because D_t counts billed tokens and ten thousand tokens from one rollout are one observation: Definition 9 makes a single evaluation one *rollout* with $\sigma_\ell = s/\sqrt{c_\ell}$, and Assumption 10 adds that the ladder is *fidelity-consistent*—a hypothesis rather than an inherited convenience, and where a reader should press, since for agents a larger turn budget can change the estimand rather than reduce variance (Remark 11).

Theorem 3 (θ-local elimination; a conditional complexity result)**.** *Fix θ and S, write $\mathcal{B} = \mathcal{B}_\theta(\pi^0; S)$ and $N = |\mathcal{B}|$, and index $\mathcal{B} = \{p_1, \dots, p_N\}$ in descending order of q_{p_i}, with $\Delta_i = q_{p_1} - q_{p_i} > 0$ for $i \ge 2$. Let t_i be the number of times* ESCALATE *is invoked on p_i before p_1 is certified better. Then under Assumption 10, with probability at least $1-\delta$, p_1 is never eliminated and*

$$\sum_{i=2}^{N} t_i \;=\; O\left(s^2 \sum_{i=2}^{N} \frac{\ln\big(N/\delta\Delta_i\big)}{\Delta_i^2}\right) \;=\; O\left(s^2\,\theta \log \frac{|S|k}{\theta} \sum_{i=2}^{N} \frac{1}{\Delta_i^2}\right). \tag{5}$$

Where the exhaustive analysis carries $\ln L = m \ln k$ inside the logarithm, Theorem 3 carries $O(\theta \log(|S|k/\theta))$, and the number of configurations to be separated drops from k^m to the ball size—both consequences of Lemma 7. Two qualifications belong here rather than in a later caveat. The theorem is conditional on Assumption 10, so with $\beta_\ell > 0$ the gaps must be read as $\Delta_i - 2\bar{\beta}$. And it is a worst-case statement carrying *no* predictive content ($|\rho| \le 0.13$, Appendix F.11), for a reason visible in the expression: $\sum_i \Delta_i^{-2}$ is dominated by the ball member closest to optimal, and separating that member is what a search minimising *regret* never needs to do. We therefore present it as a complexity result for exact identification under bounded discrepancy—the case Kishimoto et al. (2022) leave open—not as an account of where the compute goes.

Bounding regret against the *global* optimum needs the plan to be good, an empirical property of planners that must be assumed explicitly: with $q^{**} = \max_{p \in \mathcal{P}} q_p$ and $q^*_\theta = \max_{p \in \mathcal{B}_\theta(\pi^0)} q_p$, call π^0 (θ, ϵ)-*recoverable* if $q^{**} - q^*_\theta \le \epsilon$ (Assumption 13).

Under Assumptions 10 and 13, run to elimination, the returned configuration satisfies $q^{**} - q_{\hat{p}} \le \epsilon$ with probability $1-\delta$ (Proposition 14). The decomposition is deliberately stark: all the risk of discrepancy limiting sits in ϵ, which depends on the planner and not on the search and is therefore directly measurable—enumerate the space and read off the regret of the best configuration within θ overrides of what a real planner emitted, which is what Section 5.1 does. Restarts give coverage rather than convergence (Proposition 15); like BLDS we claim no completeness. Finally, under execution locality (Assumption 5) a structural failure at f implies every configuration agreeing with the incumbent on $K = \mathrm{anc}(f) \cup \{f\}$ also fails there, so restricting discrepancy allocation to K discards no repair (Proposition 16) and reduces expected probes by $|V|/|K|$—linearly in the cone fraction, not as its θ-th power as we first claimed (Proposition 18, Remark 17).

5 EXPERIMENTS

In this section, we empirically evaluate BLDS-AF across four domains. **E2** comprises plans generated by four production language models and evaluated against all 6,561 exhaustively measured configurations in NAS-Bench-Macro (Su et al., 2021), isolating Assumption 13 from all other factors. **E1** uses an ADCS simulator with exhaustive ground truth, structural failures, and configurable assumption violations, making it the only setting in which localization, ladder analyses, and sensitivity studies can be evaluated; it is also the only domain whose reward model is developed by us. **E1c** applies the same strategy code as E1 but uses the benchmark's measured rewards and noise model. Finally, **E3** consists of a four-node DAG of live multi-turn coding agents. With the exception of E3, no language model is used to plan an agent harness: in E2, for example, the decision arms correspond to convolutional building blocks rather than agentic actions, a domain gap whose mitigation is described in Appendix H.

Planner	distinct	top-1 %	percentile	$d_{\mathrm{H}}(\pi^0, p^{**})$	regret	best in $\mathcal{B}_2$
Claude Opus 5	2	92.92	99.5	2.3	0.0044	93.11
GPT-5.6	1	92.95	99.7	2.0	0.0036	93.13
Gemini 3.1 Pro	1	92.95	99.7	2.0	0.0036	93.13
Claude Haiku 4.5 (small)	1	92.45	92.5	6.0	0.0142	93.02
Node-local heuristic	1	92.95	99.7	2.0	0.0036	93.13
Uniform random (n=200)	—	90.64	52.7	5.2	0.0520	92.67

Table 1: Real language-model plans against exhaustive ground truth on NAS-Bench-Macro (CIFAR-10). Six samples per model; the optimum is 93.13%, the space mean 90.25%, the worst 45.36%. "best in $\mathcal{B}_2$" is the accuracy of the best configuration within two overrides of the plan—the ϵ of Assumption 13.

All comparisons are conducted using a *paired* design: every strategy is evaluated on the same instance under the same oracle seed. We report 95% bootstrap confidence intervals for the paired difference, computed from 4,000 resamples. A positive Δ indicates lower regret for BLDS-AF, and a row is marked **W** or **L** only when the corresponding interval excludes zero. Performance is measured using *normalized regret*, $(q^{**} - q_{\hat{p}})/(q^{**} - q^{\min})$, which can be computed exactly because ground-truth utilities are available by exhaustive enumeration at every level. Budgets are reported in units of full-fidelity run equivalents. Unless stated otherwise, experiments use $n = 120$ instances, branching factor $k = 3$, threshold $\theta = 2$, layered DAGs, and a budget of 30 runs. All results are generated by the released codebase, including an executable test suite that validates the paper's structural claims. In particular, Lemma 7 is not only proved analytically but also verified against a reference enumeration of $\mathcal{B}_\theta$.

5.1 E2: HOW FAR IS A REAL LANGUAGE MODEL'S PLAN FROM THE OPTIMUM?

Measuring Assumption 13 requires both a real planner and exhaustively known ground truth. NAS-Bench-Macro provides the latter: an eight-node chain in which each node selects one of three operations (identity or a MobileNetV2 block at one of two widths), yielding $3^8 = 6{,}561$ architectures. Every architecture has been trained to completion on CIFAR-10 under three independent seeds, providing exact performance measurements. To supply the planner, we prompt a language model to commit to a single architecture without evaluation feedback, mirroring the one-shot decision setting faced by a planning harness, and score its choice exactly. We evaluate four frontier models spanning three vendors (Claude Opus 5, GPT-5.6, Gemini 3.1 Pro), including one deliberately small model (Claude Haiku 4.5), drawing six samples from each. Prompts describe only the search space and provide no information about which architectures perform well.

The premise holds, and holds strongly. Frontier planners land at the 99.5–99.7th percentile of a space whose worst member scores 45%. Over all 24 plans $d_{\mathrm{H}}(\pi^0, p^{**})$ has median 2 and 71% satisfy $d_{\mathrm{H}} \leq 2$, exactly the radius BLDS-AF searches; reading ϵ off the same data, the best configuration within one override of a real plan has normalised regret 0.0031 and within two 0.0006. The premise also degrades with planner strength, as it must: Claude Haiku 4.5 lands at the 92.5th percentile, six overrides out, with $4\times$ the regret—outside the searched radius. So the prior does real work rather than being an artefact of an easy benchmark, uniform random plans sitting at the 52.7th percentile with $14\times$ the regret.

Two caveats are material (Appendix F.1): the prior is strong but low-entropy, three of four models emitting one identical plan coinciding with a node-local heuristic; and the plan's error sits exactly where the interactions live.

5.2 ISOLATING THE PLAN PRIOR, AND WHAT POOLING ADDS TO IT

The comparison the simulator study most needs is not against the deployed scheduler but against methods sharing the plan prior and nothing else: without those rows, "BLDS-AF beats plan-blind search" cannot be distinguished from "the ball is a good place to look." Table 2 therefore orders strategies by increasing machinery and closes with our own ablations.

The deployed scheduler is beaten decisively, and this is the least informative row. Greedy cascade attains 0.468 against our 0.055, but it evaluates one configuration, so the comparison measures the value of searching at all rather than of searching our way—the practically important result and the weakest evidence for the mechanism. Every plan-blind method is beaten too, including the strong ones: Hyperband, TPE and $(\mu{+}\lambda)$ evolutionary search are worse by $+0.105$ to $+0.119$, so the ball prior buys something plan-blind search does not recover at this budget.

	Strategy	regret	Δ	95% CI
plan-blind	Greedy cascade (deployed)	0.4679	+0.4127	[+.3678,+.4579] W
	Random search	0.1955	+0.1403	[+.1205,+.1601] W
	Successive halving	0.1718	+0.1167	[+.0995,+.1342] W
	Hyperband	0.1738	+0.1186	[+.0991,+.1382] W
	TPE (categorical BO)	0.1621	+0.1069	[+.0874,+.1271] W
	Evolutionary (μ+λ)	0.1607	+0.1055	[+.0863,+.1249] W
	UCT / MCTS	0.1859	+0.1307	[+.1140,+.1474] W
	Enumeration (random order)	0.1988	+0.1436	[+.1249,+.1624] W
plan-using	Plan + random $\mathcal{B}_2$ perturbations	0.2242	+0.1690	[+.1428,+.1957] W
	Coordinate descent from the plan	**0.0995**	+0.0444	[+.0268,+.0613] W
	Local search from the plan (restarts)	0.1381	+0.0830	[+.0659,+.0994] W
	Flat bandit elimination over $\mathcal{B}_2$	0.2095	+0.1544	[+.1328,+.1761] W
	LDS(2), single fidelity	0.2381	+0.1829	[+.1484,+.2198] W
	MLDS(2), no Eq. 7	0.2312	+0.1760	[+.1415,+.2135] W
ablations	BLDS-AF(2)	0.0553	+0.0001	[−.0133,+.0139]
	no composed candidate	0.0609	+0.0057	[−.0012,+.0128]
	additive basis only	0.0630	+0.0078	[−.0041,+.0206]
	no failure localization	0.0641	+0.0089	[−.0042,+.0224]
	+ multi-fidelity ladder	0.0708	+0.0157	[+.0001,+.0313] W
	BLDS-AF(1)	**0.0552**	—	—

Table 2: Isolation ladder, $m = 8$, $k = 3$, $|\mathcal{P}| = 6561$, $n = 120$, budget 30 full-fidelity runs, $\eta = 2$. Δ is (strategy − BLDS-AF(1)); positive favours BLDS-AF. All fourteen baselines are beaten with the interval excluding zero and none beats it; the rows inside the interval are our own ablations.

The direct transport loses to coordinate descent; pooling reverses it. BLDS-A, one arm per configuration, attains 0.1287 on the E1 instance family against coordinate descent's 0.0995 there—a loss we could not make go away and do not think should be made to (Appendix F.5 carries the paired comparison). BLDS-AF attains 0.0552, beating coordinate descent by a significant $+0.044$, and it spends its 30 runs on 30 distinct configurations where BLDS-A spends 65 evaluations to see 17.

Pooling is the whole story; composition is not. Removing the composed candidate costs only $+0.006$, inside the interval, so the batch move that most resembles coordinate descent is *not* what wins—sharing statistical strength across configurations is. Restricting the basis to main effects costs $+0.008$ here, small because these instances are 90% additive, and decisive on real agent runs where they are not (Section 5.4). The advantage over coordinate descent grows monotonically with the size of the configuration space, significant at $m = 5, 7, 8$, and holds at every interaction strength $\sigma_J \in [0, 0.8]$ where the independent-arm variant lost at every row, main effects alone being marginally better at $\sigma_J \leq 0.10$ and clearly worse from $\sigma_J \geq 0.25$ up—a crossover Proposition 6 placed in advance (Appendix F.5).

5.3 E1C: THE SAME ALGORITHMS ON A REAL BENCHMARK

Because NAS-Bench-Macro's space is a chain of 8 nodes with 3 arms each it is an ADCS instance directly: the identical strategy code runs against it with the simulator's oracle swapped for one returning real measured accuracies. Fidelity is training repetition, so both reward and noise are measured rather than chosen and the noise scale of Equation 7 is the benchmark's own across-seed spread, $s = 0.0021$. It records no structural failures, so localization never fires, and its fidelity axis satisfies monotonicity by construction.

This is the substrate where the method loses, and it loses to plan-local search. BLDS-AF attains 0.0039 and beats every plan-blind method—random, successive halving, Hyperband, TPE, evolutionary search, UCT, enumeration—by $+0.003$ to $+0.005$ with every interval excluding zero. It is beaten by local search (0.0004), random perturbation of the plan within $\mathcal{B}_2$ (0.0010), flat bandit elimination (0.0029), bound-free MLDS (0.0030) and coordinate descent (0.0035). The interaction basis does not rescue it—additive and interaction variants are indistinguishable here—which is what Proposition 6 predicts, this substrate's local-misspecification ratio being 11.9. The diagnosis is that three seeds cap the cumulative allocation at $D_t = 6$, where $\alpha_t \approx 0.004$ while the gaps that matter are 0.0002 to 0.002: the intervals are wider than every gap the search must resolve, so the dominance test cannot fire and the search drifts off a good plan (Appendix F.2).

Method	normalised regret at a token budget of 8 configs	16 configs	24 configs
Greedy cascade (deploy the planner's choice)	0.2372	0.2372	0.2372
Random search	0.0296	0.0202	0.0161
Coordinate descent from the plan	0.0422	0.0173	0.0140
BLDS-AF, additive basis only	0.0280	0.0353	0.0394
BLDS-AF (Eq. 2)	**0.0252**	**0.0000**	**0.0076**
alt. BLDS-AC, per-stage credit assignment	0.0361	0.0528	—

Table 3: Live-agent instance under the success-at-cost objective of Equation 1 ($\lambda = 0.3$ on normalised tokens), replayed from the enumerated table; budgets in multiples of the median configuration's measured token cost, $n = 80$ seeds. Under *reward alone* the instance is degenerate—58 of 81 configurations tie at 1.000, including the planner's own choice—so the cost-aware objective is the only one on which it discriminates.

We bound that loss rather than excuse it: 0.0035 normalised regret is 0.80 of one training's noise on this benchmark, and every plan-local method here, ours included, is choosing among architectures the benchmark cannot itself distinguish (Appendix F.2). The honest reading is that the proposal's two halves separate cleanly on real data: searching near the plan works, and bandit-gated multi-fidelity screening does not on a ladder this flat.

5.4 E3: LIVE AGENTS

This is the only experiment that executes agents. A four-node chain DAG builds one Python module in dependent stages; each node is a live multi-turn coding agent that sees its own import errors and may revise, up to its turn budget; each node's arm is a (model tier, reasoning effort, turn budget) triple over three tiers of one vendor family; and R is the pass fraction of a hidden 35-test suite. All $3^4 = 81$ configurations were enumerated exhaustively for 120 node executions and 391k tokens—affordable because a chain's configurations share prefixes exactly (Assumption 5), so enumeration costs $3{+}9{+}27{+}81 = 120$ node executions rather than $4 \times 81 = 324$. Every execution is a real multi-turn model call, every artifact real generated code, every reward a real test run, every cost the token count the provider billed. Strategies are then compared by replaying against those recorded measurements, which is sound because exhaustive enumeration means replay returns exactly the values an online run would have observed (Dong and Yang, 2020; Eggensperger et al., 2021); what it cannot supply is *fresh* stochasticity, and we have one replicate (Appendix F.3).

Real planners over-provision, and that is where the cost is. Asked to configure the four stages, Claude Opus 5 and GPT-5.6 both chose frontier at every node, unanimously across three samples each. That plan attains reward 1.000—it lies on a 58-configuration plateau—for 10255 tokens, where the cheapest configuration attaining 1.000 costs 6808: 51% **more tokens for no reward whatever**, a normalised regret of 0.237 under Equation 1. The finding is not that trusting the model's configuration is wrong but that the practice is expensive, and that the axis on which agent-DAG configuration selection pays is cost. Difficulty is concentrated rather than spread: nodes 0–2 are saturated, while node 3, the recursive-descent evaluator, is the discriminating stage.

The precondition of Proposition 6 fails for an additive basis. On real agent runs an additive model over node-arm indicators explains only $R^2 = 0.185$ of reward variance, with local misspecification $|\delta| = 0.147$ against a best-to-tenth gap of 0.043—a ratio of 3.4 where our simulator sits at 1.1. Real agent DAGs interact: the signal sits in one node and depends on what upstream nodes produced. The consequence is what Proposition 6 predicts—the additive surrogate gets *worse* with more budget, 0.0280 to 0.0394, growing confident in a model that cannot represent the instance, while the interaction basis reaches regret 0. A third variant exploiting the verifier's per-stage tests recovers a real per-stage signal and still *loses* (0.0361); Appendix F.3 explains why richer feedback lost to a richer basis.

6 RELATED WORK

Pipeline optimization and algorithm selection. The combined algorithm selection and hyperparameter optimization problem (Thornton et al., 2013) has been attacked by Bayesian optimization (Bergstra et al., 2011; Feurer et al., 2015), genetic programming (Olson et al., 2016), planning (Mohr et al., 2018), mixed-integer programming with ADMM (Liu et al., 2020), and Monte Carlo tree search coupled with a surrogate (Rakotoarison et al., 2019). Splitting algorithm selection from hyperparameter optimization (Mohr et al., 2018; Liu et al., 2020) is what makes the selection subproblem—and hence BLDS (Kishimoto et al., 2022)—well posed. We transport that subproblem to a new domain rather than proposing a new solver for the old one.

Multi-fidelity methods and tabular benchmarks. Successive halving (Karnin et al., 2013; Jamieson and Talwalkar, 2016), Hyperband (Li et al., 2018), its model-based combination BOHB (Falkner et al., 2018), and data-allocation upper bounds (Sabharwal et al., 2016) allocate resources by early elimination; SMAC (Hutter et al., 2011) and random search (Bergstra and Bengio, 2012) are the standard model-based and model-free references. We compare against successive halving, Hyperband, a categorical density-ratio surrogate in the style of Bergstra et al. (2011), and an evolutionary search in the style of Olson et al. (2016). Our fidelity axis is compute allocated to an agent rather than data supplied to a learner, which changes what concentration means (Definition 9) and makes monotonicity a hypothesis rather than a given. Because ADCS is a per-node categorical choice over a fixed DAG, the NAS benchmarks are its closest measurement substrate and NAS-Bench-Macro (Su et al., 2021) matches its structure exactly; NAS-Bench-201 (Dong and Yang, 2020) would be preferable but is access-gated, while HPOBench (Eggensperger et al., 2021) and HPO-B (Arango et al., 2021) have no DAG over nodes, so d_{H} and $\mathrm{anc}(f)$ have no referent there (Appendix I).

Search over language-model computation. Tree-of-thoughts style methods (Yao et al., 2023) and MCTS-guided decoding search over *token or thought sequences* within a single trajectory, where we search over *harness configurations* across a task graph: the unit of choice is which agent runs a node, not what the agent says, and the plan prior we exploit has no analogue in the token-level setting. Multi-agent frameworks likewise supply mechanisms for decomposition and delegation; our question is prior to theirs—given such a mechanism, which configuration should each node run, and how should that be found under a token budget.

7 Conclusion

Deployed agent harnesses execute task graphs greedily, at a single fidelity, under a configuration a language model guessed once, and treat a failed node as a dead end rather than as evidence. We formalized the resulting configuration-selection problem, observed that the model's own plan is exactly the value-ordering heuristic limited discrepancy search requires, and adapted BLDS accordingly, yielding a union bound whose exponent is the discrepancy budget rather than the graph size.

The evidence splits the proposal into halves deserving different verdicts. The plan prior is real and strong: plans from three frontier models sit at the 99.5th percentile of a 6561-configuration space, two overrides from the optimum, while a small model degrades to six overrides exactly as the premise predicts. The bandit machinery needs splitting—as BLDS transports it, one arm per configuration, it is beaten by plan-local coordinate descent, because the ADCS objective is largely additive over nodes and an independent-arm bandit throws that away—and pooling that evidence into a factored surrogate reverses the comparison at every interaction strength. The multi-fidelity half fares worse still, and we recommend running without it.

So: take the plan seriously as a prior and search its immediate neighbourhood; fit a factored surrogate over node-arm and edge-pair effects as you go and let it choose what to try next, rather than treating each configuration as an unrelated arm—but first check that its local misspecification is smaller than the differences you are trying to detect, because that check is cheap and it decides whether the method helps at all. And optimize cost, not reward: on the one instance where we ran real agents end to end, two frontier planners paid 51% more tokens than necessary for a result they had already achieved. What remains is to measure whether language models plan agent configurations as well as they plan the networks we could check.

AI use statement

We used generative AI tools to assist with language editing, grammar correction, and code development. In particular, AI-assisted coding tools were used during the implementation of the proposed algorithms. All AI-generated content was reviewed, verified, and, where applicable, extensively tested by the authors. For example, AI-generated code was manually inspected for correctness and validated through unit and integration testing. The authors take full responsibility for the final content of this work, including all text, code, claims, results, and other artifacts produced with the assistance of generative AI tools.

Ethics statement

This work did not involve human subjects.

Reproducibility statement

To ensure reproducibility, we have included the Python code and benchmark problem instances in the supplementary material. We will open-source the code and benchmarks if the paper is accepted.

References

J.-Y. Audibert, S. Bubeck, and R. Munos. Best arm identification in multi-armed bandits. In *COLT*, 2010.

P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2–3):235–256, 2002.

S. P. Arango, H. S. Jomaa, M. Wistuba, and J. Grabocka. HPO-B: A large-scale reproducible benchmark for black-box HPO based on OpenML. In *NeurIPS Datasets and Benchmarks*, 2021.

J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for hyper-parameter optimization. In *NeurIPS*, 2011.

J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *JMLR*, 13:281–305, 2012.

X. Dong and Y. Yang. NAS-Bench-201: Extending the scope of reproducible neural architecture search. In *arxiv*, 2020.

K. Eggensperger, P. Müller, N. Mallik, M. Feurer, R. Sass, A. Klein, N. Awad, M. Lindauer, and F. Hutter. HPOBench: A collection of reproducible multi-fidelity benchmark problems for HPO. In *arxiv*, 2021.

S. Falkner, A. Klein, and F. Hutter. BOHB: Robust and efficient hyperparameter optimization at scale. In *ICML*, 2018.

E. Even-Dar, S. Mannor, and Y. Mansour. Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems. *JMLR*, 7:1079–1105, 2006.

M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter. Efficient and robust automated machine learning. In *NeurIPS*, 2015.

W. D. Harvey and M. L. Ginsberg. Limited discrepancy search. In *IJCAI*, 1995.

F. Hutter, H. H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *LION*, 2011.

K. Jamieson and A. Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. In *AISTATS*, 2016.

Z. Karnin, T. Koren, and O. Somekh. Almost optimal exploration in multi-armed bandits. In *ICML*, 2013.

A. Kishimoto, D. Bouneffouf, R. Marinescu, P. Ram, A. Rawat, M. Wistuba, P. Palmes, and A. Botea. Bandit limited discrepancy search and application to machine learning pipeline optimization. In *AAAI*, volume 36(9), pages 10228–10237, 2022. doi: 10.1609/aaai.v36i9.21263.

L. Kocsis and C. Szepesvári. Bandit based Monte-Carlo planning. In *ECML*, 2006.

R. E. Korf. Improved limited discrepancy search. In *AAAI*, 1996.

L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *JMLR*, 18(185):1–52, 2018.

S. Liu, P. Ram, D. Vijaykeerthy, D. Bouneffouf, G. Bramble, H. Samulowitz, D. Wang, A. Conn, and A. Gray. An ADMM based framework for AutoML pipeline configuration. In *AAAI*, 2020.

F. Mohr, M. Wever, and E. Hüllermeier. ML-Plan: Automated machine learning via hierarchical planning. *Machine Learning*, 107:1495–1515, 2018.

R. S. Olson, N. Bartley, R. J. Urbanowicz, and J. H. Moore. Evaluation of a tree-based pipeline optimization tool for automating data science. In *GECCO*, 2016.

H. Rakotoarison, M. Schoenauer, and M. Sebag. Automated machine learning with Monte-Carlo tree search. In *arxiv*, 2019.

A. Sabharwal, H. Samulowitz, and G. Tesauro. Selecting near-optimal learners via incremental data allocation. In *AAAI*, 2016.

X. Su, T. Huang, Y. Li, S. You, F. Wang, C. Qian, C. Zhang, and C. Xu. Prioritized architecture sampling with Monte-Carlo tree search. In *CVPR*, pages 10968–10977, 2021. (NAS-Bench-Macro.)

C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown. Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. In *KDD*, 2013.

S. Yao, D. Yu, J. Zhao, I. Shafran, T. Griffiths, Y. Cao, and K. Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *NeurIPS*, 2023.

APPENDIX

The main text is self-contained; this appendix holds the harness characterization, the proofs, the multi-fidelity variant of the algorithm, the implementation, and every experiment and negative result reported only in summary above. Numbering of lemmas, propositions and tables continues from the main text.

A THE EXECUTION DAG OF A DEPLOYED HARNESS

We ground the problem in pi—the name of an open-source coding agent, not a constant—whose design philosophy is "primitives, not features": task tracking, sub-agents, and permissions are all extensions rather than built-ins. Task-DAG execution is provided by a task-tracking extension that maintains bidirectional `blocks`/`blockedBy` edges and dispatches nodes to a sub-agent extension over an event-bus RPC. We read the released sources of both.

A.1 STRUCTURE

A task carries an identifier, a subject, a description, a status in $\{\text{pending}, \text{in_progress}, \text{completed}, \text{deleted}\}$, an optional `agentType`, and a metadata dictionary. Dependency edges are maintained bidirectionally: declaring that u blocks v also records that v is blocked by u. Cycles, self-dependencies, and dangling references are detected and warned about but stored, so the stored relation is not guaranteed acyclic; the executable fragment is.

Execution proceeds by an explicit `TaskExecute` call listing task identifiers, which spawns each eligible task as a background sub-agent, followed—if auto-cascade is enabled—by an event-driven sweep: on each completion, the extension marks the task complete, stores the sub-agent's output in `metadata.result`, and spawns every newly unblocked dependent.

A.2 FIVE PROPERTIES OF THE SCHEDULER

The cascade is short enough that its limitations are legible directly. Its eligibility predicate selects pending tasks that have an `agentType`, are blocked by the just-completed task, and all of whose blockers are complete; it then spawns each one immediately. Table 4 records the consequences.

Property	Consequence	Observed behavior
No per-node choice	p is fixed by one LLM sample.	`agentType` is read from metadata as written at task-creation time and never revisited.
Single global configuration	One arm for the whole DAG.	The cascade configuration (model, max-turns, extra context) is stored once when `TaskExecute` is called and reused for every cascaded descendant.
Single fidelity	Cost $\propto$ evaluations.	Every spawn runs at full allocation; there is no cheap probe and no escalation ladder.
No backtracking	One bad node kills a subtree.	On failure the task reverts to pending and the cascade stops; the source comment records the intent as "branch stops." No retry, no alternative arm.
Forward-only value flow	No credit assignment.	Predecessor results are injected into a successor's prompt (truncated at 4000 characters). Nothing propagates backward.

Table 4: Properties of the deployed cascade scheduler, each read from the released extension sources. Together they characterize a greedy single-fidelity topological sweep over an unsearched configuration.

Two further details matter for what follows. The frontier loop spawns *all* newly unblocked dependents with no concurrency cap and no prioritization, so the question of which frontier node deserves the next unit of compute is not merely answered badly—it is not posed. And the successor prompt is assembled from exactly two ingredients: the successor's own description, and the stored results of its `blockedBy` set. This second fact is not incidental; it is the architectural guarantee that licenses Assumption 5.

A.3 THE AVAILABLE ARM SPACE

The sub-agent extension exposes, per spawn, an agent type (built-in or user-defined via frontmatter files), a model specified as `provider/modelId` or a fuzzy name resolved tolerantly across fifteen-plus providers, a reasoning-effort level over seven settings from off to max, a maximum turn count with a graceful "wrap up" warning before hard abort, and a tool/extension scope. It also supports isolated execution, which is what makes counterfactual re-execution of a mutated workspace sound. Finally, the harness stores sessions as *trees* with all branches in a single file, so a search tree is a native object rather than something to be bolted on.

The instrumentation surface is thus already sufficient: lifecycle hooks fire at turn start and end, on tool results, on context assembly (which may inject or filter messages per turn), and at session start; and cross-extension RPC can spawn and stop sub-agents programmatically. No fork of the harness is required.

B DERIVED QUANTITIES

The *climb premium* of Section 2, used throughout this appendix, is the cost of escalating one candidate to the top of the ladder relative to running it there directly:

$$C(\Lambda) \;=\; \frac{\sum_{\ell} c_{\ell}}{c_{\max}} \;=\; 1 + \eta^{-1} + \cdots + \eta^{-(|\Lambda|-1)} \tag{6}$$

A screened candidate that is eventually escalated has paid $C(\Lambda)$ times what a direct run would have cost, so $C(\Lambda)$ is the break-even condition for multi-fidelity screening. Appendix F.6 withdraws our earlier claim that it predicts when screening pays.

C ALGORITHM DETAILS

C.1 COMPUTE-FIDELITY CONFIDENCE BOUNDS

Let $D_t = \sum_{j=1}^{t} b\eta^{j-1} = b(\eta^t - 1)/(\eta - 1)$ be the cumulative compute invested in a configuration after t escalations, and let v be its current empirical reward. Define

$$\alpha_t \;=\; \min\left\{1,\; s\sqrt{\frac{2\log\!\big(c\,|\mathcal{B}_\theta|\,D_t^2/\delta\big)}{D_t}}\right\}, \qquad \mathrm{ucb} \;=\; v + \alpha_t, \qquad \mathrm{lcb} \;=\; v - \alpha_t, \tag{7}$$

Here $c > 4$ is a numerical exploration constant (we use $c = 4.5$), δ the confidence parameter, $|\mathcal{B}_\theta|$ the ball size from Lemma 8, D_t the cumulative allocation of Definition 9, and s the sub-Gaussian scale of rollout noise supplied by Assumption 10. The form follows the action-elimination bounds of Even-Dar et al. (2006) rather than the visit-count bounds of standard UCB (Auer et al., 2002), since what must be accounted for is compute invested rather than number of pulls; the requirement $c > 4$ is inherited from the exploration factor of Audibert et al. (2010). Because s carries the units of the reward, it must be estimated rather than assumed: on the benchmark of Section 5.3 it is measured directly from the benchmark's own across-seed variance ($s = 0.0021$ on a reward in $[0, 1]$), and Remark 4 shows what a wrong value does.

Remark 4 (The noise scale is load-bearing)**.** Both the factor s and the clip to $[0, 1]$ are required, not cosmetic. Read scale-free (that is, with $s = 1$ and no clip, as a naive transcription of the data-sampling bound would give), α_t evaluates to roughly 2.8 at $D_t = 1$—wider than the entire range of R. The dominance test of Algorithm 2 can then never fire, so the elimination mechanism is silently disabled while every baseline is unaffected, and the method degrades to an expensive random walk. We report this because we made exactly this error: correcting it moved mean normalised regret from 0.208 to 0.090 at $m = 5$ (Section 5). The same diagnosis applies to the absolute scale of b: with rungs too cheap, the top of the ladder never reaches a D_t at which any bound can resolve realistic gaps, and the η ratios alone do not fix this.

Two further changes from BLDS are encoded in Equation 7. The index D_t counts compute rather than training examples, so the concentration is over rollout stochasticity—this is what Assumption 10 supplies. And the cardinality inside the logarithm is $|\mathcal{B}_\theta|$, the ball, not L, the whole space. The second substitution is legitimate only because the search provably never leaves the ball, which is Lemma 7, and it is the source of the improved bound.

C.2 FAILURE-LOCALIZED DISCREPANCY

Under the harness's prompt-assembly discipline a node sees only its own arm and its predecessors' stored results. We name this and exploit it.

Assumption 5 (Execution locality)**.** The outcome at node i—in particular whether it fails structurally—is a function only of a_i and the results of $\mathrm{pred}(i)$.

Assumption 5 is not a modeling convenience imposed on the system; it is a property of how the deployed harness builds prompts (Section A), and it fails only if nodes communicate through a side channel such as the shared filesystem. Isolated execution restores it.

When the incumbent's run fails structurally at node f, BLDS-AF does not merely retry: it restricts subsequent discrepancy allocation to $\mathrm{anc}(f) \cup \{f\}$, searching $\mathcal{B}_\theta(\pi^0; \mathrm{anc}(f) \cup \{f\})$. Proposition 16 shows this loses nothing. This is the component with no counterpart in BLDS, because machine-learning pipelines do not have structural failures that localize blame—a badly chosen preprocessor yields a poor score, not an identifiable crash site with a known causal cone.

C.3 THE ALGORITHM WITH THE FIDELITY LADDER ENABLED

Algorithm 2 is Algorithm 1 with the ladder restored: OBSERVE climbs rungs and may break out of the climb when a candidate is dominated before the top rung is paid for. We recommend single fidelity (Section **??**); this is the variant those measurements ablate.

Algorithm 2 BLDS-AF: factored bandit limited discrepancy search for agent-DAG configuration

Require: DAG G, arm sets $\{\mathcal{A}_i\}$, plan π^0, verifier R, radius θ, exploration β, ridge λ, compute ceiling c

$A \leftarrow \lambda I$; $b \leftarrow 0$; $\leftarrow \emptyset$ ▷ ridge posterior over node-arm features, Eq. 3
OBSERVE(π^0); $p^{\mathrm{inc}} \leftarrow \pi^0$; $S \leftarrow V$
while compute spent $< c$ **do**
  if p^{inc} failed structurally at node f **then**
    $S \leftarrow \mathrm{anc}(f) \cup \{f\}$ ▷ Props. 16, 18: sound, and aims the shell
  else
    $S \leftarrow V$
  end if
  $\leftarrow \{p \in \mathcal{B}_\theta(p^{\mathrm{inc}}; S) : p \notin \}$ ▷ the discrepancy shell around the incumbent
  $\leftarrow \cup\{\arg\max_p \hat{q}_p\}$ ▷ composed candidate: the surrogate's own argmax
  if $= \varnothing$ **then**
    $\leftarrow \{\textsc{Restart}()\}$ ▷ Prop. 15
  end if
  OBSERVE$\big(\arg\max_{p\in}\ \hat{q}_p + \beta\|\phi(p)\|_{A^{-1}}\big)$ ▷ upper confidence bound over the shell
  $p^{\mathrm{rec}} \leftarrow \arg\max_{p\in} \hat{q}_p$ ▷ recommend by posterior mean, never the raw maximum
  if $\hat{q}_{p^{\mathrm{rec}}} > \hat{q}_{p^{\mathrm{inc}}}$ **then**
    $p^{\mathrm{inc}} \leftarrow p^{\mathrm{rec}}$
  end if
end while
return $\arg\max_{p\in} \hat{q}_p$

function OBSERVE(p)
  for $\ell = 1, \ldots, \lambda_{\max}$ **do** ▷ single rung unless the ladder is enabled
    $y \leftarrow \hat{R}_\ell(p)$; charge c_ℓ
    $A \leftarrow A + c_\ell\, \phi(p)\phi(p)^\top$; $b \leftarrow b + c_\ell\, \phi(p)\, y$; $\hat{\vartheta} \leftarrow A^{-1}b$ ▷ allocation-weighted, Def. 9
    $\leftarrow \cup\{p\}$
    if $\ell < \lambda_{\max}$ **and** $\hat{q}_p + \beta\|\phi(p)\|_{A^{-1}} < \max_{p'\in} \hat{q}_{p'}$ **then**
      break ▷ screened out before paying for the top rung
    end if
  end for
end function

Three properties of the loop are worth naming. The posterior is updated from *every* observation, so one evaluation is evidence about the arm at every node—this is what Proposition 6 is about and it is the component that carries the method empirically. The candidate set is the discrepancy shell around the incumbent, so the plan prior is what decides where to look, and S contracts it to a failing node's causal cone when one is observed. And recommendation is by posterior mean over evaluated configurations: not the current incumbent, which after an unproductive round may be a restart centre the search would never propose, and not the raw observed maximum, which is the winner's curse.

Three components are named above and deserve specification rather than a name.

What counts as a structural failure. A node fails structurally when it produces no usable artifact for its successors, as distinct from producing a poor one. In the harness this is detectable without heuristics, because each is an explicit lifecycle event: the sub-agent exits nonzero or aborts on the hard turn cap; the tool-execution hook reports an unrecoverable tool error; the task's terminal status is `pending` rather than `completed` after its cascade attempt; or `metadata.result` is absent or empty when a dependent's prompt is assembled. The first node in topological order satisfying any of these is f. A node that completes and merely scores badly is not a structural failure and must not be treated as one—the whole force of Propositions 16 and 18 is that a structural failure licenses an inference about a set of nodes, and a low score licenses no such inference.

RESTART. On finding no certified improvement at any $\theta \leq \Theta$ with the incumbent already at the top rung, RESTART draws a fresh incumbent uniformly from $\mathcal{P}$, clears the localization set to V, and retains the evaluation cache, so no configuration is ever paid for twice. Cached configurations already at the top rung are never re-drawn as centres. Proposition 15 is what makes this coverage rather than convergence, and we claim no completeness.

What is returned. The reported configuration is the one with the highest *lower* confidence bound among those evaluated at full fidelity—not the highest observed value, and never the current incumbent. The distinction is not cosmetic: after an unproductive round the incumbent is a random restart centre, so reporting the incumbent credits the search with a configuration it would not recommend, and the more budget it is given the more restarts it performs. Measured anytime regret then *worsens* as the budget grows, which we observed and initially mistook for a defect of the algorithm before identifying it as an accounting error (Section F.9).

Sign convention. R is a reward to be *maximised* throughout, so Algorithm 2 is stated in maximisation form; BLDS is stated for loss minimisation and every comparison is mirrored accordingly. One detail of the mirroring is easy to get wrong and we flag it, having got it wrong: the final acceptance test must compare the candidate's lcb against the *incumbent's* lcb, not against its ucb. Comparing $\mathrm{lcb}_{\mathrm{cand}}$ to $\mathrm{ucb}_{\mathrm{inc}}$ is asymmetric, and two configurations can then each certify an "improvement" over the other, so the outer loop ping-pongs between them without consuming compute and never terminates. BLDS compares ucb against ucb in its loss-minimising form, which mirrors to lcb against lcb here; that relation is monotone and the loop makes progress.

C.4 THE FACTORED SURROGATE, IN FULL

Algorithm 2 treats each configuration in the ball as an independent arm, so an evaluation of p is evidence about p alone. Section 5.2 shows that this loses to plan-local coordinate descent, and two measurements say why. First, the ADCS objective is largely additive over nodes: a per-node linear model explains $R^2 = 0.90$ of the ground-truth variance in our instances, so an evaluation of p is in fact evidence about the arm at every node simultaneously—evidence an independent-arm bandit discards and coordinate descent exploits implicitly. Second, at $\eta = 2$ BLDS-AF spends 65 evaluations to examine 17 distinct configurations, because most of the spend is re-climbing ladders, where coordinate descent examines 29 with 30 evaluations.

The fix is to keep the plan prior and replace the independent-arm bandit with a factored one. Let $\phi(p)$ carry mk indicators for the arm chosen at each node *together with* $|E|k^2$ indicators for the arm pair on each DAG edge,

$$\hat{q}_p \;=\; \sum_{i \in V} \vartheta_i(p_i) \;+\; \sum_{(u,v) \in E} \vartheta_{uv}(p_u, p_v), \tag{8}$$

so that a node's contribution may depend on what its predecessor chose. Interactions are placed on the DAG's own edges rather than on all pairs, which is Assumption 5 used as a modelling constraint: a node's outcome depends on its

own arm and on its predecessors' output, so non-adjacent pairs are not estimated at all. Maintain the ridge posterior

$$A_t = \Lambda + \textstyle\sum_{j \le t} c_{\ell_j}\, \phi_j \phi_j^\top, \qquad \hat{\vartheta}_t = A_t^{-1} \textstyle\sum_{j \le t} c_{\ell_j}\, \phi_j y_j, \qquad \hat{q}_p = \phi(p)^\top \hat{\vartheta}_t, \tag{9}$$

weighting each observation by its allocation c_{ℓ_j}, which is the correct weight under Definition 9. The prior Λ is diagonal but *not* scalar: main effects receive a weak penalty λ and the interaction block a much stronger one $\lambda_E \gg \lambda$. This group-wise penalty is what makes Equation 2 safe to adopt unconditionally. A single scalar ridge forces a choice between two failure modes—weak enough and the $|E|k^2$ interaction coefficients overfit a thirty-observation budget, strong enough to prevent that and the interactions never activate where they are needed—whereas penalising the blocks separately makes the model collapse gracefully onto the additive one while data is scarce and switch the interactions on as evidence accumulates. The practitioner therefore does not have to know in advance which regime the instance is in, which matters because the regimes differ sharply: our simulator instances are 90% additive while real agent runs are 19% additive (Section 5.4). BLDS-AF then (i) proposes the discrepancy shell $\mathcal{B}_\theta(p^{\mathrm{inc}})$ as candidates, (ii) adds the surrogate's own argmax $\arg\max_p \hat{q}_p$ as a single *composed* candidate—coordinate descent's whole sweep proposed in one shot, and evaluated like any other candidate so that a proposal additivity got wrong is rejected rather than kept, (iii) evaluates the candidate maximising $\hat{q}_p + \beta \|\phi(p)\|_{A_t^{-1}}$, and (iv) recommends $\arg\max \hat{q}_p$ over evaluated configurations rather than the raw observed maximum. Steps (iii) and (iv) are LinUCB restricted to a Hamming ball; step (iv) is where the pooling pays twice, since a shrunk estimate is a better recommendation rule than a single noisy observation.

Whether pooling helps is a bias–variance question with a computable answer.

Proposition 6 (When pooling beats direct comparison)**.** *Let $r(p) = q_p - \phi(p)^\top \vartheta^\star$ be the residual of the best additive approximation, and for a Hamming-1 pair (p, p') differing at node i write $\delta = r(p) - r(p')$ for the* local *misspecification and $\Delta = |q_p - q_{p'}|$ for the gap to be resolved. Comparing p and p' by direct evaluation, as coordinate descent does, gives an unbiased contrast of variance $2s^2/c_{\max}$. Comparing them through Equation 3 after n observations of allocation $c_{\max}$ gives a contrast of bias δ and variance $O(k\, s^2/(n\, c_{\max}))$. Pooling therefore orders the pair correctly with higher probability whenever*

$$|\delta| \;\lesssim\; \Delta \qquad \textit{and} \qquad n \gtrsim k, \tag{10}$$

and is worse than direct comparison when $|\delta| \gg \Delta$, however large n becomes.

How to check the condition, concretely. The condition is only useful if a practitioner can evaluate it before committing, so we state it as a procedure rather than as an inequality. (i) Take any set of configurations whose rewards you already know—an exhaustive enumeration on a small graph, a pilot sample, or the cache of a previous run; a few dozen suffices. (ii) Regress reward on the node-arm indicators and keep the residuals r. (iii) For every pair in that set at Hamming distance 1, record $|r(p) - r(p')|$; their RMS estimates $|\delta|$. (iv) Divide by the reward difference you actually need to resolve—if you care about the best configuration rather than the top ten, use the gap from the best to the tenth best. (v) A ratio near 1 or below means the additive basis suffices; a ratio of several means it does not, and the interaction terms of Equation 2 are needed. Every quantity is computed from rewards you have already paid for, and no search is run. For calibration, the three substrates in this paper measure 1.1 (simulator), 11.9 (NAS-Bench-Macro) and 3.4 (live agents, on the cost-aware objective that instance is actually optimised for); the additive basis wins on the first and fails on the other two. The live-agent instance also illustrates step (iv): under reward alone its ten best configurations tie exactly, so the gap is zero and the ratio is undefined—which is itself the useful signal that reward is the wrong objective there.

The condition involves the *local* residual difference, not the global residual, and the distinction is what makes the method work at all: two configurations at Hamming distance 1 share every interaction term that does not touch node i, so their residuals are strongly correlated and δ is substantially smaller than $\|r\|$. We measure both. On our simulator instances the local $|\delta|$ has RMS 0.026 against a best-to-tenth gap of 0.023, a ratio of 1.1, and BLDS-AF wins; on NAS-Bench-Macro the same ratio is 11.9 and it does not help (Section 5.2). Because both quantities are computable from ground truth without running any search, Equation 10 is usable as a pre-flight check on a new instance family rather than only as a post-hoc explanation.

Two honesty notes. The composed candidate of step (ii) may lie outside $\mathcal{B}_\theta(p^{\mathrm{inc}})$, so BLDS-AF does not satisfy Lemma 7 and the ball-restricted union bound of Theorem 3 does not apply to it; it is a surrogate-derived proposal, not a discrepancy probe. And λ_E is a real hyperparameter, not a formality: we select it once by sweeping it against interaction strength (Table 10) and use $\lambda_E = 128\lambda$ throughout. At $\lambda_E = 16\lambda$ the interaction block costs a *significant* -0.008 where the objective is additive, so the constant carries the safeguard rather than the idea alone.

D Proofs and Deferred Analysis

Kishimoto et al. (2022) bound the number of full-fidelity trainings needed to identify the best pipeline when the discrepancy budget equals the pipeline depth, and state explicitly that the properties of the practically-used regime $\theta < m$ "remain an open question." Our setting forces exactly that regime—m is the size of a task graph and θ is small—so we address it.

D.1 The search never leaves the ball

Lemma 7 (Ball invariance)**.** *Every configuration passed to* Bounds *by* Search$(p^{\mathrm{inc}}, \cdot, \cdot, 1, \theta, S)$ *lies in* $\mathcal{B}_\theta(p^{\mathrm{inc}}; S)$.

Proof. Induct on the node index with the invariant that on entry to Search$(p, \cdot, \cdot, i, \theta', S)$ we have $|\{j < i : p_j \neq p_j^{\mathrm{inc}}\}| + \theta' = \theta$ and $p_j = p_j^{\mathrm{inc}}$ for all $j < i$ with $j \notin S$. The invariant holds at entry, where $i = 1$ and $\theta' = \theta$. The recursion has two branches. The non-deviating branch leaves both the prefix disagreement count and θ' unchanged. The deviating branch, taken only when $i \in S$, increases the disagreement count by one and decrements θ' by one, preserving the sum. Since $\theta' \geq 0$ is enforced by the base case at line 13, the disagreement count of any configuration reaching Bounds is at most θ, and no disagreement is ever introduced outside S. □

Lemma 8 (Ball size)**.** *If* $|\mathcal{A}_i| \leq k$ *for all* i *and* $|S| = s$*, then* $|\mathcal{B}_\theta(\pi^0; S)| = \sum_{j=0}^{\theta} \binom{s}{j}(k-1)^j \leq (\theta+1)\Big(\frac{e\,s\,(k-1)}{\theta}\Big)^\theta$*, hence* $\log|\mathcal{B}_\theta(\pi^0; S)| = O\big(\theta \log(sk/\theta)\big)$.

Proof. A configuration at Hamming distance j from π^0 within S is specified by choosing j of the s eligible nodes and, at each, one of at most $k-1$ alternative arms, giving the exact count. Bounding each of the $\theta+1$ terms by the largest and applying $\binom{s}{j} \leq (es/j)^j$ gives the closed form; taking logarithms gives the rate. □

Lemma 8 is the quantitative content of the plan-as-heuristic idea. The exhaustive space has $\log L = m \log k$; the ball has $\log|\mathcal{B}_\theta| = O(\theta \log(mk/\theta))$. Discrepancy limiting moves m from the position of a multiplier to the position of a logarithm, at the price of restricting attention to configurations near the plan.

D.2 Elimination under bounded discrepancy

Before stating the concentration premise we have to say what an observation is, because the index D_t is measured in billed tokens and tokens are emphatically not independent samples. Ten thousand tokens emitted by one rollout are one observation, not ten thousand.

Definition 9 (Observation model)**.** A single evaluation of configuration p at rung ℓ is one *rollout*: an independent execution of the whole DAG under p with compute allocation c_ℓ, scored once by the verifier. Rollouts are independent given (p, ℓ)—which is what isolated re-execution (Section E) is for—and a rollout at rung ℓ yields $\hat{R}_\ell(p)$ with mean $q_p^{(\ell)}$ and sub-Gaussian parameter $\sigma_\ell = s/\sqrt{c_\ell}$. After a set T of rollouts of p, the estimator is the allocation-weighted mean $\hat{Q}_p = \sum_{j \in T} c_{\ell_j} \hat{R}_{\ell_j}(p)/D_t$ with $D_t = \sum_{j \in T} c_{\ell_j}$.

The content of the model is the single assumption $\sigma_\ell = s/\sqrt{c_\ell}$: a rollout granted twice the compute produces a score whose variance is halved. That is the agent-side analogue of training on twice the data, and it is what makes D_t—a sum of costs, not a count of pulls—the right index: the weighted mean above has sub-Gaussian parameter $s/\sqrt{D_t}$ exactly. Tokens enter only through c_ℓ, as a price, never as a sample count.

Assumption 10 (Rollout concentration)**.** Under Definition 9 there is a noise scale $s > 0$ such that for every p and every cumulative allocation D_t, $\Pr\big(|\hat{Q}_p - q_p^\Lambda| \geq \alpha\big) \leq 2\exp\big(-\alpha^2 D_t/(2s^2)\big)$, where q_p^Λ is the allocation-weighted mean of the rung estimands $q_p^{(\ell)}$. Moreover the ladder is *fidelity-consistent*: $\beta_{\max} = 0$ and β_ℓ is non-increasing in ℓ.

Inverting the tail at level $2\delta/(cND_t^2)$ produces the half-width α_t of Equation 7; Remark 4 records what goes wrong if s is dropped. Separating q_p^Λ from q_p is what keeps the statement honest, and it is the part a reader should press on.

Remark 11 (The fidelity premise is the weak point, and it is not a technicality)**.** Assumption 10 plays the role data-subsampling concentration plays in BLDS, but its second clause is a genuine hypothesis rather than an inherited

convenience. Concentration alone gives $\hat{Q}_p \to q_p^{\Lambda}$; the object of interest is q_p, and the two coincide only as $\beta_\ell \to 0$. Writing the gap out, any candidate whose elimination is certified at cumulative allocation D_t satisfies

$$|q_p - \hat{Q}_p| \;\le\; \alpha_t + \bar{\beta}_t, \qquad \bar{\beta}_t = \frac{1}{D_t}\sum_{j\in T} c_{\ell_j}\beta_{\ell_j}, \tag{11}$$

so an elimination is sound only when the true gap exceeds $2(\alpha_t + \bar{\beta}_t)$ rather than $2\alpha_t$. Unlike α_t, the bias term does not shrink with more spending at a *fixed* rung: it shrinks only by climbing. For supervised learning β_ℓ is benign—more training examples do not change what is being measured. For agents it is at risk in a specific way: a larger turn budget can make an agent wander into a different behaviour rather than execute the same behaviour more reliably, which is a change of estimand, not a reduction of variance. Section F.12 measures the cost of $\beta_\ell > 0$ by inducing rank inversion between cheap and expensive rungs; a live-harness test remains E3.

Theorem 12 (θ-local elimination; a conditional complexity result)**.** *Fix θ and S, write $\mathcal{B} = \mathcal{B}_\theta(\pi^0; S)$ and $N = |\mathcal{B}|$, and index $\mathcal{B} = \{p_1, \dots, p_N\}$ in descending order of q_{p_i}, with $\Delta_i = q_{p_1} - q_{p_i} > 0$ for $i \ge 2$. Let t_i be the number of times* ESCALATE *is invoked on p_i before p_1 is certified better. Then under Assumption 10, with probability at least $1-\delta$, p_1 is never eliminated and*

$$\sum_{i=2}^{N} t_i \;=\; O\!\left(s^2 \sum_{i=2}^{N} \frac{\ln\!\big(N/\delta\Delta_i\big)}{\Delta_i^2}\right) \;=\; O\!\left(s^2\,\theta \log \frac{|S|k}{\theta} \sum_{i=2}^{N} \frac{1}{\Delta_i^2}\right). \tag{12}$$

Proof. By Lemma 7 no configuration outside $\mathcal{B}$ is ever evaluated, so the union bound in the definition of α_t need range over $\mathcal{B}$ only; this is what justifies the substitution of $|\mathcal{B}_\theta|$ for L in Equation 7. The remaining steps follow Even-Dar et al. (2006) and the corresponding argument in Kishimoto et al. (2022). Given that substitution, Assumption 10 yields, for any D_t and any $p_i \in \mathcal{B}$, $\Pr(|\hat{Q}_i^t - q_i| \ge \alpha_t) \le 2\exp(-\alpha_t^2 D_t/(2s^2)) \le 2\delta/(cND_t^2)$, the last step by the choice of α_t in Equation 7. Taking $c > 4$ and summing over $\mathcal{B}$ and over allocations $D_t \in \{D_1, D_2, \dots\}$—whose reciprocal squares are summable because D_t grows geometrically—the union bound gives that with probability at least $1-\delta$ every estimate is within α_t of its mean simultaneously, so the best configuration in $\mathcal{B}$ is never eliminated. Since $\alpha_t \to 0$ as $D_t \to \infty$, every suboptimal p_i is eventually eliminated. Elimination of p_i requires reaching t_i with $\hat{\Delta}_{t_i} = \hat{Q}_1^{t_i} - \hat{Q}_i^{t_i} \le 2\alpha_{t_i}$; combining with the event $|\hat{Q}_i^t - q_i| < \alpha_t$ gives $\Delta_i - 2\alpha_t = (q_1 - \alpha_t) - (q_i + \alpha_t) \ge \hat{Q}_1 - \hat{Q}_i \ge 2\alpha_t$, so $t_i = O(s^2 \ln(N/\delta\Delta_i)/\Delta_i^2)$, the s^2 entering through α_t. Substituting Lemma 8 for $\ln N$ gives the second form. □

Compared with the exhaustive analysis, which carries $\ln L = m \ln k$ inside the logarithm, Theorem 3 carries $O(\theta \log(|S|k/\theta))$. The number of *configurations* N that must be separated also drops from k^m to the ball size. Both reductions are consequences of Lemma 7, which is why the bounded-θ case is tractable at all: restricting the search restricts the union bound.

Two qualifications belong here rather than in a later caveat. First, the theorem is conditional on Assumption 10, whose bias clause is exactly what Remark 11 flags; with $\beta_\ell > 0$ the gaps Δ_i must be read as $\Delta_i - 2\bar{\beta}$, and candidates separated by less than the bias are not separable at any allocation. Second, it is a worst-case complexity statement and we find it carries no predictive content: measured escalations on runs allowed to reach elimination are essentially uncorrelated with the bound's own predictor (Section F.11, $|\rho| \le 0.13$). The reason is visible in the expression. The sum $\sum_i \Delta_i^{-2}$ is dominated by the ball member closest to optimal, and separating that member is precisely what a search minimising *regret* never needs to do. The bound answers an identification question; the objective is a regret question. We therefore present Theorem 3 as what it is—a complexity result for exact identification under bounded discrepancy, which is the case Kishimoto et al. (2022) leave open—and not as an account of where the compute goes.

D.3 What bounded discrepancy costs

Theorem 3 certifies the best configuration *in the ball*. Bounding regret against the global optimum requires the plan to be good, which is an empirical property of language-model planners and must be assumed explicitly.

Assumption 13 ((θ,ϵ)-recoverable plan)**.** Let $q^{**} = \max_{p\in\mathcal{P}} q_p$ and $q_\theta^* = \max_{p\in\mathcal{B}_\theta(\pi^0)} q_p$. The plan π^0 is (θ,ϵ)-recoverable if $q^{**} - q_\theta^* \le \epsilon$.

Proposition 14 (Regret decomposition)**.** *Under Assumptions 10 and 13, with probability at least $1-\delta$ the configuration $\hat{p}$ returned by Algorithm 2 run to elimination satisfies $q^{**} - q_{\hat{p}} \le \epsilon$, and the compute required is bounded by Theorem 3.*

Proof. Theorem 3 gives $q_{\hat{p}} = q_\theta^*$ on the $1-\delta$ event; Assumption 13 gives $q^{**} - q_\theta^* \le \epsilon$. □

The decomposition is deliberately stark: all the risk of discrepancy limiting is concentrated in ϵ, a quantity that depends only on the planner and not on the search. It is therefore directly measurable—take a real planner, enumerate the space, and read off the regret of the best configuration within θ overrides of what the planner emitted. This is the single most informative measurement in the paper's programme and Section 5.1 makes it: over 24 plans from four production language models on a space of 6561 exhaustively evaluated configurations, $\epsilon = 0.0031$ at $\theta = 1$ and 0.0006 at $\theta = 2$, with median $d_{\mathrm{H}}(\pi^0, p^{**}) = 2$. Small-$\theta$ search is vindicated there. Section F.10 supplies the complementary controlled version—planting π^0 at a known distance—and locates where the vindication stops: the method significantly beats plan-blind tree search at $d_{\mathrm{H}} \leq 1$, ties through $d_{\mathrm{H}} = 4$, and significantly loses at $d_{\mathrm{H}} \geq 5$. Frontier planners sit at $d_{\mathrm{H}} \approx 2$ and a small model at $d_{\mathrm{H}} = 6$, on either side of that boundary.

Proposition 15 (Restart reachability)**.** *Any $p \in \mathcal{P}$ satisfies $d_{\mathrm{H}}(p, \pi^0) \leq m$ and hence lies within $\lceil m/\theta \rceil$ successive incumbent moves of π^0. Consequently the incumbent graph, whose vertices are configurations and whose edges join configurations at Hamming distance at most θ, has diameter at most $\lceil m/\theta \rceil$.*

Proposition 15 is a statement about reachability, not about convergence: BLDS-AF moves its incumbent only on a certified improvement, so it traverses this graph only where a monotone improving path exists. This is the same gap that afflicts BLDS, whose authors note that a series of local searches "may lead to an optimized pipeline that is very different from the initial one" without guaranteeing it. Random restarts provide coverage in the absence of such a path, and we do not claim completeness.

D.4 Failure localization is free

Proposition 16 (Localization soundness)**.** *Suppose configuration p fails structurally at node f and let $K = \mathrm{anc}(f) \cup \{f\}$. Under Assumption 5, every $p' \in \mathcal{P}$ with $p'_i = p_i$ for all $i \in K$ also fails structurally at f. Hence restricting the allowed set to K discards no configuration that repairs the failure.*

Proof. By Assumption 5 the outcome at f is determined by a_f together with the results of $\mathrm{pred}(f)$, which are in turn determined, by induction along the topological order, by the arms at $\mathrm{anc}(f)$. All of these agree between p and p' by hypothesis, so the outcome at f agrees, and f fails under p'. □

Soundness says restriction is free. It does not say why restriction helps, and the two available explanations make opposite predictions, so the question is decidable by experiment.

Remark 17 (The cardinality account, which the data reject)**.** By Lemma 8 the ball contracts from $|\mathcal{B}_\theta(\pi^0; V)|$ to $|\mathcal{B}_\theta(\pi^0; K)|$, a factor $(|V|/|K|)^\theta$ up to lower-order terms, so the benefit should grow as the cone shrinks—largest on wide graphs, negligible on chains. An earlier version of this paper advanced that account. It is wrong. Across graph shapes the observed ordering is close to reversed, and per instance the benefit correlates with cone fraction at $\rho = -0.09$, a magnitude consistent with noise (Section F.8). We keep the counting statement only as an upper bound on how much of the ball is provably excluded, which is a fact about the ball and not about where the compute goes.

Proposition 18 (Localization as targeting)**.** *Under Assumption 5, a configuration p' differs from p in its structural outcome at f only if $p'_i \neq p_i$ for some $i \in K$. Consequently, among the $|V|(k-1)$ single-deviation candidates in $\mathcal{B}_1(p; V)$, exactly $|K|(k-1)$ can repair the failure at f, so depth-first search over the unrestricted ball examines a repairing candidate with probability $|K|/|V|$ per probe, while search restricted to K does so with probability 1. The expected number of probes before a repairing candidate is examined therefore falls by a factor $|V|/|K|$—linearly in the cone fraction, not as its θ-th power.*

Proof. Immediate from Proposition 16: any candidate agreeing with p throughout K reproduces the failure, so only deviations inside K can repair it. Counting single deviations gives the ratio, and the expected-probes statement is the mean of the corresponding geometric variable. □

Proposition 18 is what the measurements support. It predicts that localization should raise the rate at which accepted repairs actually touch the broken node, and that the benefit should track that rate rather than the cone's cardinality. Both hold: localized repairs change the crash node in 72.5% of cases against 46.9% unlocalized (a gain of 25.6 points, CI $[+22.6, +28.8]$), and the per-instance benefit correlates with that gain at $\rho = +0.50$ against $\rho = -0.09$ for cone fraction (Section F.8). It also explains why the effect is only linear in $|V|/|K|$ and therefore why it does not order graph shapes as sharply as Remark 17 predicted.

The practical consequence is that the deployed harness, on encountering a failed node, reverts it to pending and abandons its descendants, extracting nothing from the event, while the same event identifies a set of nodes one of which must be

changed for the failure to be repaired at all. Both propositions depend entirely on Assumption 5; Section F.12 measures what happens as it fails.

E Implementation as a Harness Extension

BLDS-AF is realized as a harness extension, requiring no modification to the harness itself, and composes with the existing task and sub-agent extensions rather than replacing them.

Interception. The extension registers on the context-assembly hook to capture the plan π^0 as emitted, and on the tool-execution hook to intercept the `TaskExecute` call that would otherwise begin the greedy cascade. Auto-cascade is disabled; BLDS-AF becomes the scheduler.

Arms. A node's arm set is specified as a cross product over agent type, model, effort level, and turn budget, filtered against the models the user has actually configured. Spawning is done over the existing cross-extension RPC, so BLDS-AF inherits fuzzy model resolution and scope enforcement for free.

Fidelity ladder. Rung ℓ pairs a model tier with an effort level and a turn budget. Concretely, the four-rung ladder we specify is

$$\lambda_1 = (\text{small},\ \textsf{low},\ 10\ \text{turns}),\quad \lambda_2 = (\text{small},\ \textsf{high},\ 25),\quad \lambda_3 = (\text{mid},\ \textsf{high},\ 25),\quad \lambda_4 = (\text{frontier},\ \textsf{high},\ 60),$$

with b set so that λ_1 costs about an eighth of λ_4. The compute index D_t is measured in billed tokens read from the completion events rather than in nominal turns, so the bounds in Equation 7 are indexed by the quantity the objective actually charges for. The ladder's *shape* matters more than its rungs: by Equation 6 the premium is $C(\Lambda) = 1+\eta^{-1}+\eta^{-2}+\eta^{-3}$, so the price ratio between adjacent model tiers is what decides whether screening can pay, and Section F.6 measures the threshold. Published per-token prices across tiers of a single provider span one to two orders of magnitude, which puts real ladders in the favourable regime and makes $\eta = 2$ pessimistic.

Isolation and rollback. Counterfactual re-execution requires that a discarded configuration leave no trace. Each evaluation runs in a separate git worktree using the sub-agent extension's isolation support; the worktree is discarded unless the configuration becomes the incumbent. This is also what restores Assumption 5 against filesystem side channels, and its overhead is reported rather than elided.

Caching. $(v, \mathrm{lcb}, \mathrm{ucb})$ triples are memoized in the task store keyed by configuration, persisted at project scope so that repeated runs over the same graph warm-start. Following BLDS, the incumbent's cache entry is invalidated when its prompt or skill set changes, since its reward is then no longer the same quantity.

Plan capture. π^0 is read, not inferred. The extension registers on the context-assembly hook and records the per-node `agentType` together with the cascade-level model, effort and turn budget at the moment `TaskExecute` is intercepted—which is exactly the tuple the deployed scheduler would have used unmodified, so π^0 is the deployed harness's own decision and the greedy-cascade baseline is literally the harness running as shipped. Nodes for which the plan leaves a field unset inherit the cascade default, and a plan that names an unavailable model is resolved by the harness's own fuzzy matcher before being recorded.

Reproducibility. Everything reported here is produced by the released code: the simulator, the fifteen baselines and BLDS-AF behind one `Oracle` interface, the benchmark loader, the plan-elicitation script with the verbatim prompt and the raw model replies, and a test suite that checks the paper's structural claims executably—including Lemma 7, which is verified against a reference enumeration of $\mathcal{B}_\theta$ rather than only proved. Because the same strategy code runs against the simulator, the tabular benchmark and a live harness, the three evidence levels of Section 5 differ only in which oracle is supplied.

F Additional Experiments

F.1 E2: the two caveats in full

Two caveats we consider material. First, frontier planners are nearly unanimous: three of four models emitted the identical plan on every sample, and it coincides with the node-local heuristic (use the largest block everywhere). The

prior is therefore strong but low-entropy, which means sampling several plans and searching around each—an obvious extension—would buy very little, and it means the LLM is not contributing knowledge a simple heuristic lacks on this benchmark. Second, the plan's error is concentrated exactly where the interactions live: the optimum differs from the modal plan by using identity at two positions, i.e. the planner knows to build a large network but not where to prune it. That is encouraging for the method's premise and is also the kind of error a node-local planner makes by construction.

F.2 E1C: THE BENCHMARK TABLE IN FULL

Because NAS-Bench-Macro's space is a chain of 8 nodes with 3 arms each, it is an ADCS instance directly: the identical strategy code runs against it with the simulator's oracle swapped for one that returns real measured accuracies. Fidelity is training repetition—rung ℓ averages $\ell{+}1$ of the three measured seeds—so both the reward and the noise are measured rather than chosen, and the noise scale of Equation 7 is the benchmark's own across-seed spread, $s = 0.0021$. Two things the benchmark cannot test: it records no structural failures, so localization never fires; and its fidelity axis satisfies monotonicity by construction, so Remark 11's worry is untouched.

	from a frontier LLM plan (99.7th pct)			from the small model's plan (92.5th pct)		
Strategy	regret	Δ	95% CI	regret	Δ	95% CI
Greedy cascade (deployed)	0.0036	-0.0006	[$-.0007$,$-.0006$] L	0.0142	-0.0014	[$-.0016$,$-.0012$] L
Random search	0.0088	$+0.0046$	[$+.0040$,$+.0051$] W	0.0088	-0.0068	[$-.0074$,$-.0062$] L
Successive halving	0.0086	$+0.0043$	[$+.0038$,$+.0049$] W	0.0086	-0.0070	[$-.0076$,$-.0065$] L
Hyperband	0.0090	$+0.0047$	[$+.0041$,$+.0053$] W	0.0090	-0.0067	[$-.0073$,$-.0061$] L
TPE	0.0064	$+0.0022$	[$+.0016$,$+.0027$] W	0.0064	-0.0092	[$-.0098$,$-.0086$] L
Evolutionary	0.0071	$+0.0028$	[$+.0022$,$+.0035$] W	0.0071	-0.0085	[$-.0092$,$-.0079$] L
UCT / MCTS	0.0081	$+0.0038$	[$+.0033$,$+.0044$] W	0.0081	-0.0075	[$-.0081$,$-.0070$] L
Plan + random $\mathcal{B}_2$	0.0010	-0.0033	[$-.0034$,$-.0031$] L	0.0087	-0.0070	[$-.0074$,$-.0065$] L
Coordinate descent	0.0035	-0.0008	[$-.0008$,$-.0008$] L	0.0056	-0.0101	[$-.0102$,$-.0099$] L
Local search	0.0004	-0.0039	[$-.0039$,$-.0038$] L	0.0067	-0.0089	[$-.0091$,$-.0088$] L
Flat bandit over $\mathcal{B}_2$	0.0029	-0.0013	[$-.0017$,$-.0009$] L	0.0062	-0.0095	[$-.0100$,$-.0090$] L
BLDS-AF, additive basis	0.0039	-0.0000	[$-.0002$,$+.0002$]	0.0141	$+0.0002$	[$-.0002$,$+.0005$]
BLDS-AF (Eq. 2)	**0.0039**	—	—	**0.0141**	—	—

Table 5: NAS-Bench-Macro, $|\mathcal{P}| = 6561$, $n = 200$ repetitions, budget 30 full-fidelity runs, measured $s = 0.0021$. Δ is (baseline $-$ BLDS-AF); positive favours BLDS-AF, so the **L** rows are baselines that beat it. The left block starts from the modal frontier-LLM plan of Table 1, the right from the small model's plan. This is the paper's clearest negative result and the diagnostic of Proposition 6 anticipates it.

This is the substrate where the method loses, and it loses to plan-local search. BLDS-AF attains 0.0039. It beats every plan-blind method—random, successive halving, Hyperband, TPE, evolutionary search, UCT, enumeration—by $+0.003$ to $+0.005$ with every interval excluding zero. It is beaten by local search (0.0004), by random perturbation of the plan within $\mathcal{B}_2$ (0.0010), by flat bandit elimination (0.0029), by bound-free MLDS (0.0030) and by coordinate descent (0.0035). Enabling the interaction basis does not rescue it: the additive and interaction variants are indistinguishable here (0.0039 both), which is what Proposition 6 predicts, since this substrate's local-misspecification ratio is 11.9 and no basis we can afford at this budget fixes that.

How large is that loss? The gap to local search is 0.0035 normalised regret, which is 0.17 top-1 points—and the benchmark's own across-seed standard deviation is 0.213 points. The loss is therefore 0.80 of a single training's noise, on a substrate where 20 of 6561 architectures lie within that distance of the optimum and the best-to-tenth gap is itself only 0.49 seed standard deviations. Our $n = 200$ intervals exclude zero because they resample the replay, not the training: the ordering is a reliable fact about these three recorded seeds and would not survive retraining the architectures. We state the loss without that caveat first because it is a loss, and the caveat second because it bounds what the loss means: on this benchmark every plan-local method, ours included, is choosing among architectures the benchmark cannot itself distinguish.

Why, and what it tells us. The diagnosis is Equation 7 together with the benchmark's fidelity axis. With three seeds the cumulative allocation cannot exceed $D_t = 6$, at which $\alpha_t \approx 0.004$, while the gaps that matter here are 0.0002 to 0.002. The confidence intervals are therefore wider than every gap the search needs to resolve, so the dominance test cannot fire, acceptance is effectively arbitrary, and BLDS-AF drifts off a good plan. This is the failure mode of Remark 4 arising not from a mis-specified s—s is measured here—but structurally, from a ladder too shallow to resolve

the instance. The climb premium is $C(\Lambda) = 2.0$, flatter than any ladder in Section F.6, which is the same diagnosis in the units of Equation 6.

The honest reading is that the two halves of this paper's proposal separate cleanly on real data. *Searching near the plan* works: every plan-local row beats every plan-blind row in the left block. *Bandit-gated multi-fidelity screening* does not, on a ladder this flat, and is worse than not screening at all. That is a statement about a property of the benchmark's fidelity axis, and Section F.6 shows how steep a ladder has to be before it reverses.

F.3 E3: REPLAY, THE DISCARDED FIRST TASK, ARM EFFECTS, AND PER-STAGE CREDIT

What is live and what is replayed. The experiment has two stages and the distinction matters enough to state plainly. *First*, agents are executed live: every node execution is a real multi-turn model call, every artifact is real generated code, every reward is a real test run, and every cost is the token count the provider billed. *Second*, the search strategies are compared by replaying against those recorded measurements. The replay is not a weaker substitute for running the search online: because the space is enumerated *exhaustively*, any configuration a strategy might choose has already been executed, so replay returns exactly the values an online run would have observed. This is why tabular benchmarks are built this way (Dong and Yang, 2020; Eggensperger et al., 2021), and it is what makes paired comparisons across fifteen strategies and eighty seeds affordable at all—online, the same comparison would re-execute the same configurations thousands of times.

What replay genuinely cannot supply is *fresh* stochasticity. We have one replicate, so the variance of these numbers over independent agent executions is unmeasured, and prefix sharing leaves the noise correlated across configurations sharing a prefix (common random numbers, which helps our paired comparisons and hurts any claim about variance). That is the real limitation of this experiment, and a second replicate is the cheapest thing that would remove it.

A first design that was not an ADCS instance. Our initial task specification was too easy to pose a configuration-selection problem at all: on 40 enumerated configurations the all-cheap configuration already scored 1.000, the mean was 0.994, and the best-to-tenth gap was 0.0000. With no spread across arms every regret number would have been noise, so the run was abandoned and the specification hardened—interval division by zero-crossing intervals, integer powers with the even-exponent and negative-interval subtleties, and a real recursive-descent evaluator with parentheses, unary minus and precedence. Discriminativeness was then verified *before* spending: a reference implementation scores $35/35$ and a stages-one-to-two-only module scores $12/35$. We report the discarded attempt because "configurations do not differ" is a live outcome for this kind of experiment and one a reader should know we encountered.

	mean reward with this arm at this node			
Arm	node 0	node 1	node 2	node 3
cheap (small model, low effort, 1 attempt)	0.975	0.985	0.949	**0.913**
mid (mid model, medium effort, 2)	0.992	0.973	0.981	**0.987**
frontier (large model, high effort, 3)	0.934	0.943	0.970	**1.000**

Table 6: Per-node arm effects on the live-agent instance. 58 of 81 configurations score 1.000; rewards take six distinct values down to 0.0; one configuration fails structurally. Nodes 0–2 are saturated—any arm suffices—while node 3, the recursive-descent evaluator, is the discriminating stage. Configuration sensitivity in a real agent DAG is concentrated, not spread.

An alternative that did not work. The verifier is not a scalar: its tests are organised by stage, so one execution reports how well *each* node did. We built the method that exploits this—one ridge model per stage over only the arms in that stage's causal cone, which is Assumption 5 as a modelling constraint, plus a separate additive model of token cost—and the signal is real: a configuration scoring $24/35$ overall reads $[1.00, 1.00, 1.00, 0.08]$ per stage, localising the failure to node 3 from a single evaluation. It nonetheless *lost*, at 0.0361 against 0.0252 (Table 3). The reason is visible in Table 6: nodes 0–2 are saturated, so three of the four per-stage signals are constant and carry no information, and credit assignment buys three uninformative models while the scalar surrogate spends all its capacity on the one quantity that varies. A richer basis beat richer feedback here. We expect the ordering to reverse where difficulty is spread across nodes, and record that as a hypothesis rather than a result.

F.4 Interaction strength: the excuse that failed

Coordinate descent is exactly optimal when the objective decomposes over nodes, so the natural response to Table 2 is that the simulator's node interactions are too weak. That response is testable and it is false.

Interaction σ_J	BLDS-A(2)	Coord. descent	Local search	UCT	Δ vs coord. descent
0.00	0.1048	0.0904	0.0978	0.1365	-0.0143 [$-.0307$,$+.0017$]
0.10	0.1197	0.0939	0.1163	0.1591	-0.0259 [$-.0434$,$-.0080$] L
0.25	0.1521	0.1113	0.1563	0.1754	-0.0409 [$-.0626$,$-.0187$] L
0.50	0.1929	0.0954	0.1993	0.1936	-0.0975 [$-.1215$,$-.0741$] L
0.80	0.2058	0.1016	0.2183	0.1998	-0.1042 [$-.1303$,$-.0781$] L

Table 7: Interaction-strength sweep, $m = 8$, $k = 3$, $n = 120$. σ_J is the standard deviation of the pairwise edge interactions that make the objective non-decomposable.

The gap to coordinate descent *widens* monotonically with interaction strength, from insignificant at $\sigma_J = 0$ to -0.104 at $\sigma_J = 0.8$. Coordinate descent is nearly flat across the sweep while BLDS-AF degrades. So the non-decomposability that motivates combinatorial search does not by itself favour discrepancy-limited search over greedy coordinate improvement at a tight budget; if anything the reverse. The operative variable is how many distinct configurations a method can afford to evaluate confidently, and that is governed by the ladder, not the objective.

F.5 The factored bandit: scale and interaction sweeps

Section 5.2 left the method beaten by plan-local coordinate descent, and Section F.4 ruled out the natural excuse. BLDS-AF (Section C.4) keeps the plan prior and replaces the independent-arm bandit with the factored one of Equation 3. Table 8 reports the result at the same budget and the same instances as Table 2.

Strategy	regret	evaluations	Δ vs BLDS-AF(1)
Coordinate descent from the plan	0.0995	30.0	$+0.0444$ [$+.0274$,$+.0618$] W
Local search from the plan	0.1381	30.0	$+0.0830$ [$+.0653$,$+.1000$] W
LDS(2), independent arms	0.2381	65.0	$+0.1829$ [$+.1470$,$+.2199$] W
BLDS-AF(1)	**0.0552**	30.0	—
BLDS-AF(2)	0.0553	30.0	$+0.0001$ [$-.0133$,$+.0139$]
BLDS-AF(1), no composed candidate	0.0609	30.0	$+0.0057$ [$-.0012$,$+.0128$]
abl. additive basis only	0.0630	30.0	$+0.0078$ [$-.0041$,$+.0206$]
BLDS-AF(1) + multi-fidelity	0.0708	69.7	$+0.0157$ [$+.0001$,$+.0313$] W

Table 8: The factored bandit against coordinate descent and against BLDS-AF. $m = 8$, $k = 3$, $n = 120$, budget 30 full-fidelity runs, $\eta = 2$—identical to Table 2. Δ positive favours BLDS-AF(1).

Pooling closes the gap and then some. BLDS-AF attains 0.0552 against coordinate descent at 0.0995, a significant $+0.044$, and beats independent-arm discrepancy search by $+0.183$. The evaluation column shows the second half of the mechanism—BLDS-AF spends its 30 runs on 30 distinct configurations, where the independent-arm variant spends 65 evaluations to see 17, because most of its budget re-climbs ladders.

Pooling is the whole story; composition is not. Removing the composed candidate costs $+0.006$, inside the interval. The batch move that most resembles coordinate descent is therefore *not* what wins; sharing statistical strength across configurations is. Restricting the basis to main effects costs $+0.008$ here—small, because these instances are 90% additive, and decisive on real agent runs where they are not (Section 5.4). Adding multi-fidelity screening makes things significantly worse ($+0.016$), as Section F.6 predicts for $\eta = 2$: the climb premium is paid twice over and the evaluation count rises to 69.7 while distinct configurations fall.

		normalised regret			
m	$\lvert\mathcal{B}_2\rvert/\lvert\mathcal{P}\rvert$	Coord. descent	BLDS-A(2)	BLDS-AF(1)	Δ vs coord. descent
4	0.407	0.0258	0.0589	0.0200	$+0.0058$ [$-.0029$,$+.0146$]
5	0.210	0.0482	0.1195	0.0180	$+0.0302$ [$+.0176$,$+.0431$] **W**
6	0.100	0.0555	0.1734	0.0419	$+0.0136$ [$-.0008$,$+.0284$]
7	0.045	0.0725	0.1993	0.0460	$+0.0265$ [$+.0147$,$+.0390$] **W**
8	0.020	0.0961	0.2250	0.0544	$+0.0417$ [$+.0266$,$+.0562$] **W**

Table 9: Scale sweep, $k = 3$, $n = 120$, 30-run budget. BLDS-AF improves on BLDS-A at every m; against coordinate descent the advantage grows monotonically with the size of the configuration space and reverses in the smallest.

The advantage grows with the space, and no longer reverses. Against coordinate descent BLDS-AF is ahead at every m tested, significantly at $m = 5, 7, 8$ and inside the interval at $m = 4$ and $m = 6$, with the margin largest in the largest space. The independent-arm formulation lost significantly at $m = 4$; the factored one does not, because at small m there are few parameters to estimate and the group-wise prior keeps the interaction block from costing more than it can pay for. The direction of Section F.7 survives—structure-exploiting machinery earns its keep as the space grows beyond what a budget can cover—but the reversal at the bottom end does not.

σ_J	Coord. descent	LDS(2)	BLDS-AF(1)	additive only	Δ vs coord. descent
0.00	0.0904	0.2079	0.0474	**0.0423**	$+0.0430$ [$+.0304$,$+.0569$] W
0.10	0.0939	0.2268	0.0502	**0.0496**	$+0.0437$ [$+.0294$,$+.0582$] W
0.25	0.1113	0.2689	**0.0572**	0.0639	$+0.0540$ [$+.0385$,$+.0699$] W
0.50	0.0954	0.2913	**0.0587**	0.0729	$+0.0367$ [$+.0215$,$+.0528$] W
0.80	0.1016	0.3076	**0.0569**	0.0701	$+0.0447$ [$+.0270$,$+.0625$] W

Table 10: Interaction-strength sweep, $m = 8$, $k = 3$, $n = 120$—the same instances as Table 7, where BLDS-A lost at every row. Bold marks the better of the two factored variants.

The interaction basis earns its place exactly where predicted. BLDS-AF beats coordinate descent at every interaction strength, by $+0.037$ to $+0.054$, never reversing. Restricting it to main effects is marginally better at $\sigma_J \leq 0.10$, where the objective really is additive and the interaction block can only cost, and clearly worse from $\sigma_J \geq 0.25$ upward, where it cannot represent the instance. A crossover placed by Proposition 6 in advance and then found in the data is the strongest evidence we have that the mechanism is the one named rather than a coincidence of tuning, and the group-wise prior is what keeps the cost on the additive side of that crossover statistically insignificant.

Where it does not help, and why we can say so in advance. On NAS-Bench-Macro BLDS-AF attains 0.0039 against BLDS-A's 0.0043—an improvement too small to matter—and is still beaten by local search. Proposition 6 says when to expect this, and the quantities are measurable without running anything:

Substrate	additive R^2	$\lVert r\rVert$	local $\lvert\delta\rvert$	best-to-10th gap	$\lvert\delta\rvert/\Delta$
Simulator ($m = 8$, $k = 3$)	0.904	0.0459	0.0261	0.0228	**1.1**
NAS-Bench-Macro	0.694	0.0149	0.0123	0.0010	**11.9**

Table 11: The pre-flight diagnostic of Proposition 6. $\lVert r\rVert$ is the global additive residual RMS, local $\lvert\delta\rvert$ its RMS difference across Hamming-1 pairs. On the benchmark the measured per-seed noise $s = 0.0021$ is itself $2.1\times$ the gap, so its top is below its own measurement resolution.

The local misspecification is $11.9\times$ the gap the search must resolve on the benchmark against $1.1\times$ on the simulator, so pooling cannot order the leading configurations there however many observations it collects—and neither can anything else, since the benchmark's own seed noise exceeds the same gap. We report the failure alongside the diagnostic that predicts it rather than reporting only the substrate where the method wins.

F.6 LADDER GEOMETRY: THE WITHDRAWN CLIMB-PREMIUM CLAIM

Equation 6 says escalating one candidate to the top costs $C(\Lambda)$ times a direct full-fidelity run. At $\eta = 2$ that is 1.875: screening must discard nearly half its candidates before reaching the top merely to break even. At $\eta = 16$ it is 1.07 and screening is almost free. This is not a tuning knob but a property of the arm space—the price ratio between adjacent model tiers—so the question is whether BLDS-AF's losses are intrinsic or an artefact of a flat ladder.

η	$C(\Lambda)$	BLDS-AF	+ ladder	MLDS(2)	Coord. desc.	UCT	Δ vs coord. desc.
2	1.875	0.0566	0.0734	0.2696	0.1162	0.1784	+0.0596 [+.0434,+.0763] W
3	1.481	0.0566	0.0657	0.2489	0.1162	0.1784	+0.0596 [+.0434,+.0763] W
4	1.328	0.0566	0.0710	0.2610	0.1162	0.1784	+0.0596 [+.0434,+.0763] W
8	1.143	0.0566	0.0573	0.2514	0.1162	0.1784	+0.0596 [+.0434,+.0763] W
16	1.067	0.0566	0.0625	0.2230	0.1162	0.1784	+0.0596 [+.0434,+.0763] W

Table 12: Ladder-geometry sweep, $m = 8$, $k = 3$, $n = 120$, four rungs, with b rescaled as η varies so that the *top rung's allocation* is held fixed at 64 units. Only the climb premium changes. Single-fidelity methods are therefore constant by construction, which is the sanity check that the rescaling worked.

We withdraw this claim: the effect was an artifact of our own instrumentation. An earlier version of this sweep reported that regret fell fivefold as the ladder steepened and attributed it to the climb premium. That sweep varied η with b fixed, so the top rung's allocation grew with η; since an observation's noise goes as $s/\sqrt{\text{cost}}$, a steeper ladder also made every full-fidelity observation more precise. The sweep was measuring an easier problem, not a cheaper climb.

Holding the top-rung allocation fixed removes the effect entirely. Single-fidelity BLDS-AF is constant at 0.0566 across the whole range—as it must be, since it never touches the ladder—and enabling the ladder produces no trend as the premium falls from 1.875 to 1.067 (0.073, 0.066, 0.071, 0.057, 0.063), while staying worse than single fidelity at every point. So $C(\Lambda)$ is a correct cost identity that predicts nothing we can measure, and the deployment guidance we previously drew from it—add multi-fidelity screening when the price ratio between model tiers is large—is unsupported. Together with Table 8 and Section F.12, the recommendation is to run BLDS-AF at a single fidelity.

F.7 WHERE THE PLAN PRIOR BEATS SEARCHING FROM THE ROOT

Table 13 sweeps m at fixed $\theta = 2$.

m	$\lvert\mathcal{P}\rvert$	$\lvert\mathcal{B}_2\rvert$	$\lvert\mathcal{B}_2\rvert/\lvert\mathcal{P}\rvert$	BLDS-AF(2)	UCT	Δ vs UCT (95% CI)
3	27	19	0.704	0.0440	0.0143	−0.0297 [−.0449,−.0153] **L**
4	81	33	0.407	0.0495	0.0495	−0.0000 [−.0182,+.0163]
5	243	51	0.210	0.0885	0.0872	−0.0013 [−.0236,+.0196]
6	729	73	0.100	0.1190	0.1311	+0.0121 [−.0106,+.0342]
7	2187	99	0.045	0.1377	0.1421	+0.0045 [−.0152,+.0241]
8	6561	129	0.020	0.1601	0.1739	+0.0138 [−.0091,+.0357]

Table 13: m-sweep at fixed $\theta = 2$, $k = 3$, layered DAGs, $n = 120$, 30-run budget.

The sign of Δ moves from -0.030 at $m = 3$ to $+0.014$ at $m = 8$, and only the $m = 3$ row is significant: **in small spaces BLDS-AF is significantly worse than MCTS from the root**, exactly as Lemma 8 implies—when $|\mathcal{B}_\theta|/|\mathcal{P}| \approx 0.7$ the ball is nearly the whole space, so the prior buys almost nothing while the multi-fidelity machinery still costs. Above the crossover the advantage is positive but, in this instance family, not significant; in the independent family of Table 2 the same comparison at $m = 8$ is $+0.057$ with the interval excluding zero. We report both rather than the more favourable one: the direction is consistent across families and the effect size is not, so "BLDS-AF beats MCTS" is a regime-conditional claim with family-dependent magnitude, not a general one.

Sweeping the arm count instead of the node count is the more realistic axis, since a real harness offers agent type × model × effort × turns rather than three options. There the advantage is significant and stable: at $m = 6$, BLDS-AF beats UCT by $+0.022$ [+.001,+.044] at $k = 3$, $+0.021$ [+.001,+.041] at $k = 4$ and $+0.022$ [+.002,+.042] at $k = 6$, with $k = 2$ and $k = 5$ inside the interval. The $k = 2$ sweep over $m \in \{4, \ldots, 12\}$ shows the same crossover non-monotonically, so we claim the trend at $k \geq 3$ and not a general law.

A meta-policy that does not work. Since $|\mathcal{B}_\theta|/|\mathcal{P}|$ is a closed-form function of (m, k, θ) and costs no evaluations, the obvious move is to switch: run BLDS-AF when the ratio is below a threshold τ and UCT otherwise. It works as far as it goes—pooled over the sweep the rule beats always-BLDS-AF by $+0.008$ [$+.003$,$+.013$] and is insensitive to τ anywhere in $[0.02, 0.2]$—but it does *not* beat simply always using UCT ($+0.003$ [$-.003$,$+.009$]). Turning the regime boundary into a decision rule recovers BLDS-AF's loss in small spaces and adds nothing over the rival it was meant to arbitrate against, so we report the rule as a negative result rather than as a contribution.

F.8 Failure localization: real effect, wrong mechanism, no benefit

Localization is the component with no counterpart in BLDS and the largest single contributor in the ablation ($+0.114$ at $m = 8$). Remark 17 and Proposition 18 give incompatible accounts of why, so we instrumented where each accepted repair lands.

Shape	$\lvert K\rvert/m$	blast	with loc.	without	hit%	hit% (no loc.)	benefit (95% CI)
chain	0.629	0.537	0.0367	0.0388	77.2	56.4	$+0.0022$ [$-.0061$,$+.0107$]
layered	0.441	0.388	0.0279	0.0341	79.3	63.1	$+0.0062$ [$-.0020$,$+.0144$]
diamond	0.455	0.386	0.0334	0.0364	82.0	59.9	$+0.0030$ [$-.0048$,$+.0115$]
wide	0.307	0.301	0.0350	0.0428	84.0	57.6	$+0.0078$ [$-.0047$,$+.0202$]

Table 14: Localization by DAG shape. $m = 6$, $k = 3$, structural failure rate 0.25, $n = 120$. $|K|/m$ and "blast" ($(|\mathrm{desc}(f)|+1)/m$) are averaged over the crash sites actually observed. "hit%" is the fraction of accepted repairs that changed the crash node.

Localization does not pay for itself, and we withdraw it as a contribution. No shape reaches significance; the four point estimates span $+0.002$ to $+0.008$ against a method whose regret is around 0.03. It still does mechanically what it claims—localized repairs change the crash node in 80.6% of cases against 59.2% unlocalized, a gain of 21.4 points whose interval excludes zero—so the aiming effect of Proposition 18 is real. It simply buys nothing, and the per-instance correlations confirm there is nothing left to explain: over the 405 runs in which localization fired, the benefit correlates with cone fraction at $\rho = +0.02$, blast radius at $\rho = -0.02$, hit rate at $\rho = -0.04$ and hit-rate gain at $\rho = +0.02$. Every mechanism we proposed, including the one our own earlier measurements supported, now correlates with the outcome at approximately zero.

Pooling subsumes it. A factored surrogate already infers that an arm is broken from every observation containing it, so restricting the shell to a failing node's cone tells the search something it has largely worked out. An earlier version of this paper measured this component at $+0.11$ and called it the largest single contributor; that was true of an independent-arm bandit, which had no other way to find the broken node. Section F.12 completes the case: under a sufficiently violated locality assumption localization becomes significantly *harmful*. We therefore keep Propositions 16 and 18 as correct statements about what a structural failure licenses, and claim no performance benefit from acting on them.

F.9 θ, and an accounting error that looked like a defect

An earlier version of this study reported that BLDS-AF's regret grew as the compute ceiling grew, and at $m = 8$ that $\theta = 2$ beat $\theta = 1$ by $+0.134$ at a budget of 120 runs. Both were artefacts of the same error: the anytime record logged the current *incumbent*, which after an unproductive round is a random restart centre the search would never recommend. More budget means more restarts, so the recorded answer was more often a restart centre. Recording what the search would actually return—the highest lower confidence bound among fully evaluated configurations—removes both effects.

Budget (runs)	BLDS-AF(1)	BLDS-AF(2)	LDS(1)	LDS(2)	MLDS(2)	θ=2 vs θ=1
30	0.0613	0.0540	0.1884	0.1884	0.1832	$+0.0072$ [$-.0083$,$+.0240$]
120	0.0128	0.0134	0.0674	0.0804	0.0695	-0.0007 [$-.0061$,$+.0048$]
480	0.0061	0.0017	0.0479	0.0409	0.0391	$+0.0044$ [$+.0019$,$+.0073$] W

Table 15: Discrepancy radius against budget, $m = 8$, $k = 3$, $n = 60$. $|\mathcal{B}_1| = 17$, so a single-fidelity radius-1 sweep needs about 17 full-fidelity runs before $\theta = 2$ can matter at all.

Regret now falls monotonically with budget, as it must. It also shows that **the discrepancy radius barely matters**: $\theta = 2$ is indistinguishable from $\theta = 1$ at budgets 30 and 120 and ahead by $+0.004$ at 480—detectable, but an order of magnitude smaller than the gap to any method that does not pool. Searching the radius-1 shell around a good plan is nearly as good as searching further. The discrepancy *prior* carries the method; the discrepancy *budget* is close to a free parameter, and we withdraw the earlier claim that θ was load-bearing.

F.10 Where discrepancy limiting stops working

Section 5.1 measured real plan distances; this measures the consequence of bad ones by planting the plan at a known distance from the optimum.

$d_{\mathrm{H}}(\pi^0, p^{**})$	0	1	2	3	4	5	6
Greedy cascade	0.000	0.188	0.359	0.455	0.565	0.645	0.729
BLDS-AF(1)	0.073	0.092	0.109	0.130	0.129	0.152	0.168
BLDS-AF(2)	0.074	0.092	0.111	0.135	0.131	0.154	0.166
UCT	0.127	0.127	0.127	0.127	0.127	0.127	0.127
Random	0.144	0.144	0.144	0.144	0.144	0.144	0.144

Table 16: Normalised regret as the plan is planted further from the optimum. $m = 6$, $k = 3$, $n = 120$ per cell. UCT and random ignore π^0 and are flat by construction, which is the sanity check on the instance generator.

BLDS-AF degrades gracefully across the full diameter, 0.074 to 0.166, while greedy cascade degrades catastrophically. The crossover is where it should be: BLDS-AF significantly beats UCT at $d_{\mathrm{H}} \leq 1$ ($+0.053$, $+0.035$), ties at $d_{\mathrm{H}} \in \{2, 3, 4\}$, and significantly loses at $d_{\mathrm{H}} = 5$ (-0.027) and $d_{\mathrm{H}} = 6$ (-0.039). Read against Table 1, frontier planners sit at $d_{\mathrm{H}} \approx 2$—inside the tie region, near the favourable edge—and the small model at $d_{\mathrm{H}} = 6$, exactly where discrepancy limiting should be abandoned. The two measurements meet.

F.11 A negative result on the bound itself

Theorem 3 bounds escalations to elimination. An earlier attempt to test it was uninformative rather than negative, because the compute ceiling bound before elimination completed and escalation counts saturated in $[128, 144]$. Disabling restarts makes the run halt on elimination instead, which is the event the theorem is about; all 120 runs then finish, and escalations span $[117, 478]$ at $m = 5$ and $[133, 953]$ at $m = 6$.

The bound still does not track them. At $m = 5$ the Spearman correlation between the theorem's predictor and measured escalations is -0.03; at $m = 6$ it is $+0.10$. Truncating the gap sum at $\Delta_i > 0.01$ and $\Delta_i > 0.05$ gives $+0.09$ and $+0.06$ ($m = 5$), $+0.09$ and $+0.11$ ($m = 6$); ball size alone gives $+0.09$ and $+0.13$. Every variant is within noise of zero. The explanation is structural, as noted after Theorem 3: $\sum_i \Delta_i^{-2}$ is dominated by the ball member closest to optimal, whose separation a regret-minimising search never needs to pay for. We report the null so that the bound is not read as empirically supported, and we retain it as a complexity result rather than as a prediction.

F.12 Pricing the two assumptions we cannot validate

Assumptions 5 and 10 can only be validated on a live harness (E4, E3). A simulator can do something weaker but not worthless: violate each by a controlled amount and measure what the violation costs. A method that degrades gracefully under a false premise is a different proposition from one that collapses, and which of the two it is has been unknown.

Fidelity rank inversion (Assumption 10, Remark 11). We give each configuration its own fidelity sensitivity, scaled to vanish at the top rung so that q_p remains the target, which makes cheap rungs rank configurations differently from expensive ones—the agent-wandering failure mode. The violation must be configuration-dependent to bite: a bias depending only on the rung shifts every configuration equally and cancels in every comparison, and an earlier version of this switch did exactly that and reported, misleadingly, that nothing was at stake.

β strength	ρ(cheap, top)	+ ladder	single fidelity	MLDS(2)	Flat bandit	Succ. halving
0.00	1.000	0.0365	0.0395	0.1479	0.1289	0.1178
0.15	0.799	0.0478	0.0395	0.1664	0.1494	0.1197
0.35	0.621	0.0735	0.0395	0.1916	0.1881	0.1328
0.70	0.325	0.1197	0.0395	0.2350	0.2805	0.1605
degradation at $\rho = 0.33$		$+0.0832$	**0.0000**	$+0.0871$	$+0.1515$	$+0.0426$

Table 17: Fidelity rank-inversion sweep, $m = 6$, $k = 3$, $n = 120$. ρ is the measured Spearman correlation between cheapest-rung and full-fidelity expected reward over the whole space; $\rho = 1$ is the ladder Assumption 10 assumes. Single-fidelity methods are omitted: they never observe a cheap rung and are exactly unaffected.

The result is a straightforward argument for not using the ladder. Single-fidelity BLDS-AF is unaffected *by construction*—it never observes a cheap rung, so a column of exact zeros is non-participation, not robustness, and we say so because an earlier version of this table reported that immunity as a property of the method. Enabling the ladder makes it as vulnerable as everything else: $+0.083$ as ρ falls to 0.33, against $+0.087$ for bound-free MLDS and $+0.043$ for successive halving. The confidence bounds of Equation 7 therefore do not buy protection from a misranking ladder.

Locality violation (Assumption 5). We install side channels: pairs (u, v) with $u \notin \mathrm{anc}(v)$ whose arm can break v. The crash is then observed at v but caused at u, outside the cone Proposition 16 restricts to, so the contraction is unsound by a controlled amount.

leak	with loc.	without loc.	UCT	causal%	localization benefit (95% CI)
0.00	0.0285	0.0299	0.1269	83.3	$+0.0015$ [$-.0044$,$+.0077$]
0.25	0.0335	0.0242	0.1232	76.0	-0.0093 [$-.0196$,$+.0004$]
0.50	0.0362	0.0353	0.1250	71.3	-0.0009 [$-.0107$,$+.0086$]
1.00	0.0495	0.0330	0.1351	67.4	-0.0165 [$-.0301$,$-.0038$] **L**

Table 18: Locality-violation sweep, $m = 6$, $k = 3$, $n = 120$. "leak" is the fraction of nodes given a side channel from a non-ancestor; "causal%" is the fraction of accepted repairs touching the node actually responsible.

Localization never helps significantly at any leak rate, and at total violation it significantly *hurts* (-0.0165, interval excluding zero), because the search is then confined to a cone that provably cannot contain the repair. Read with Section F.8 this settles the component: it is sound, it aims repairs where it claims to, it buys nothing once evidence is pooled, and it becomes actively harmful when the one assumption we cannot check in a live harness fails badly enough. We recommend running BLDS-AF without it, which also removes any performance claim resting on Assumption 5.

G SUMMARY OF CLAIMS

	Claim	Verdict
C1	Beats the deployed greedy cascade	**Confirmed.** $+0.34$ regret at $m = 8$, at every m, k and shape tested, and on the real benchmark from a weak plan. The safest result and the least mechanism-specific.
C2	The plan prior beats plan-blind search	**Confirmed.** Every plan-blind baseline is significantly worse at $m = 8$, including Hyperband, TPE and evolutionary search (Table 2); the same ordering holds on the real benchmark (Table 5). Against MCTS specifically the claim is regime-conditional: significant loss at $m = 3$, family-dependent magnitude above the crossover.
C2′	. . . and beats plan-*local* search	**Refuted for BLDS-A; recovered by BLDS-AF.** Coordinate descent beats BLDS-A by -0.029 and the gap widens with interaction strength (Table 7); it closes to insignificance at $\eta = 16$ (Table 12). The factored bandit reverses it outright: $+0.028$ at $\eta = 2$, significant at every interaction strength (Tables 8 and 10).
C9	Pooling across configurations beats independent arms	**Confirmed, with a stated precondition.** BLDS-AF beats BLDS-A by $+0.057$ at $m = 8$ and at every m tested. Against coordinate descent it wins at $m = 8$, ties $m = 5$–7 and loses at $m = 4$ (Table 9). Composition contributes nothing ($+0.001$, n.s.)—the pooling does the work. Fails where Proposition 6 says it must: local misspecification $11.9\times$ the gap on NAS-Bench-Macro versus $1.1\times$ in simulation (Table 11).
C3	The multi-fidelity ladder helps	**Refuted; we recommend single fidelity.** Enabling it costs a significant $+0.016$ (Table 8); with the top-rung allocation held fixed it shows no trend against the climb premium (Table 12); and it degrades $+0.083$ under fidelity rank inversion, no better than the methods it was meant to beat (Table 17). Eq. 7 is also vacuous unless instantiated with a measured noise scale (Remark 4).
C4	Failure localization matters	**Withdrawn.** Not significant on any of 4 shapes ($+0.002$ to $+0.008$), not significant at any leak rate, and significantly *harmful* at total locality violation (-0.017). It still aims repairs as claimed (80.6% vs 59.2% hit rate, W), but every mechanism correlates with the benefit at $\lvert\rho\rvert \leq 0.04$ because there is no benefit left: pooling already infers which arm is broken. Measured at $+0.114$ for an independent-arm bandit, which had no other way to find it.
C5	Advantage grows with space size	**Supported in k, weak in m.** Significant vs UCT at $k = 3, 4, 6$; in m the sign changes monotonically but only the $m = 3$ loss is significant.
C6	Escalations track the Thm. 3 bound	**Refuted.** With runs allowed to reach elimination, $\lvert\rho\rvert \leq 0.13$ across four predictor variants (Section F.11). The bound is retained as a complexity result, not a prediction.
C7	θ matters	**Near-refuted.** $\theta = 2$ is indistinguishable from $\theta = 1$ at budgets 30 and 120 and ahead by $+0.004$ at 480—an order of magnitude below the gap to any method that does not pool (Table 15). An earlier, larger positive finding was an accounting artefact and is withdrawn.
C10	Pairwise interaction terms are needed	**Confirmed, conditionally.** Costs $+0.008$ (n.s.) where the objective is additive and wins from $\sigma_J \geq 0.25$ upward (Table 10); decisive on live agents, where the additive basis degrades with budget and the interaction basis reaches regret 0 (Table 3). The group-wise prior is what makes it safe to default (Section C.4).
C11	Per-stage credit assignment beats a scalar reward	**Refuted.** 0.0361 against 0.0252 on live agents (Table 3): three of four per-stage signals are constant there, so the extra models carry no information.
C8	The τ-rule improves on both fixed choices	**Refuted.** Beats always-BLDS-AF ($+0.008$) but not always-UCT ($+0.003$, n.s.).
E2	Plan distance ϵ is small in practice	**Confirmed on real ground truth for real planners** (Section 5.1): 99.5–99.7th percentile, $d_{\mathrm{H}} \approx 2$, $\epsilon = 0.0006$ at $\theta = 2$; degrades to $d_{\mathrm{H}} = 6$ for a small model. Not measured for agent-harness arms specifically.
E3	Reward is monotone in fidelity (Ass. 10)	**Untested in a harness**; sensitivity measured (Table 17): BLDS-AF degrades by -0.011, rivals by up to $+0.158$.
E4	Execution locality holds (Ass. 5)	**Untested in a harness**; sensitivity measured (Table 18): localization's benefit crosses zero near leak 0.4.

Table 19: Claim-by-claim verdicts across the three levels of evidence.

H Evaluation Protocol for Real Agents

Two premises can be established only inside a running harness, and this section specifies how. **These experiments have not been run.** E2 is no longer among them: Section 5.1 measures plan distance for four production language models against exhaustive ground truth, on a per-node configuration problem over a DAG but not over agent-harness arms, so what remains for E2 is a domain transfer rather than a first measurement.

Benchmarks. Primary: a subset of a verified software-engineering benchmark where the verifier is a programmatic test suite, giving a bounded, low-bias R. Secondary, and reported separately because the reward is noisier: multi-stage tasks scored by an LLM judge. We deliberately do not lead with judge-scored results, since every guarantee in Section 4 is conditional on the quality of R.

Arms and ladder. Agent type over {general-purpose, explore, custom auditor}; model over three tiers spanning small, mid, and frontier; effort over {low, medium, high}; turn budget over $\{10, 25, 60\}$. Ladder $\eta = 2$ with b set so that rung one costs roughly an eighth of full fidelity.

Baselines. (i) **Greedy cascade**, the deployed scheduler of Section A—the baseline that matters, since it is what users run today; (ii) random configuration search; (iii) successive halving and Hyperband over configurations; (iv) UCT-style Monte Carlo tree search (Kocsis and Szepesvári, 2006) from the root, the natural alternative that discards the plan prior; (v) LDS(1) and LDS(2) without multi-fidelity, isolating the contribution of the confidence bounds; (vi) MLDS, multi-fidelity without the bounds of Equation 7, isolating the contribution of Equation 7 itself; (vii) BLDS-AF without failure localization, isolating Proposition 16.

Primary metrics. Success-at-cost curves; compute to reach a fixed target reward; and—because a search method that wins only by spending more is not a contribution—total billed tokens including all discarded rollouts and worktree overhead.

Diagnostic measurements that remain (E2′–E4).

1. **E2′: plan distance over harness arms.** Section 5.1 measures $d_{\mathrm{H}}(\pi^0, p^{**})$ for real planners over convolution blocks; what it cannot settle is whether a model's priors over (agent type, model, effort, turns) are as good as its priors over network width. Repeat the measurement with harness arms on graphs small enough to enumerate—$m \leq 5$, $k = 3$, so $|\mathcal{P}| \leq 243$ full-fidelity agent runs per instance—and compare the resulting distribution to Table 1. The discriminating prediction is available in advance: if harness plans sit at $d_{\mathrm{H}} \leq 2$ the method transfers, and if they sit at $d_{\mathrm{H}} \geq 5$ then Table 16 says discrepancy limiting should be abandoned for unrestricted search.
2. **E3: fidelity monotonicity.** Measure $q_p^{(\ell)}$ along the ladder and test whether β_ℓ decreases, i.e. whether cheap rungs estimate the same quantity as expensive ones. The informative statistic is not the mean shift but the rank correlation between rung-1 and top-rung orderings, since a rung-wise constant bias is invisible to every search method (Section F.12); Table 17 gives the curve to compare against. This also supplies a harness estimate of s, without which Equation 7 cannot be instantiated (Remark 4).
3. **E4: execution locality.** Assumption 5 is asserted from reading the harness's prompt-assembly code, not measured. Test it by re-running a node under a fixed configuration and fixed predecessor results while varying non-ancestors, and checking that structural outcomes agree; disagreement localizes a side channel, and the natural culprit is the shared filesystem. Table 18 says what is at stake quantitatively: localization's benefit crosses zero at a leak rate near 0.4, so the measurement needed is the leak rate, not merely its existence.

I Related Work: Substrate Selection

Tabular benchmarks with exhaustive ground truth. Because ADCS is a per-node categorical choice over a fixed DAG, the NAS benchmarks are its closest existing measurement substrate. We evaluate on NAS-Bench-Macro (Su et al., 2021), whose 3^8 exhaustively trained configurations over a chain of eight nodes match ADCS's structure exactly. NAS-Bench-201 (Dong and Yang, 2020) would be preferable—it has a genuine cell DAG and an epoch-wise fidelity axis, so its cheap rungs are genuinely biased—but its tabular release is several gigabytes and access-gated; HPOBench (Eggensperger et al., 2021) and HPO-B (Arango et al., 2021) supply real multi-fidelity data over mixed continuous–categorical spaces with no DAG over nodes, so d_{H} and $\mathrm{anc}(f)$ have no referent there. We report which claims each substrate can and cannot test in Section 5 rather than treating one benchmark as covering all of them.

J EXTENDED LIMITATIONS

In descending order of severity as we see it.

One live-agent instance, one replicate. Agents are executed in exactly one place (Section 5.4): a four-node DAG, three arms per node, all 81 configurations enumerated over 120 live multi-turn agent executions. That instance is small, single-domain, and—most importantly—run *once*. Because enumeration is exhaustive, replaying search strategies against it reproduces what an online run would have observed, so the replay is not the limitation; the single replicate is. We therefore have no estimate of the noise scale s on agent runs, and prefix sharing leaves what noise there is correlated across configurations that share a prefix. A reader who wants the variance of these numbers over independent agent executions will not find it here, and a second replicate is the cheapest thing that would supply it.

Everything else rests on substrates that are not agent harnesses. Localization, the fidelity ladder and the assumption-sensitivity studies exist only in the simulator, because no available benchmark has structural failures or a biased fidelity axis; the plan-distance measurement of Section 5.1 uses convolution blocks. The step from "a language model plans a per-node configuration well" to "a language model configures an agent harness well" is an inference over most of this paper, and Section H says what would settle it.

Plan-local coordinate descent is a genuinely strong baseline, and it beat our first formulation. An independent-arm bandit over the plan ball loses to it at matched compute; only after pooling evidence into the factored surrogate of Equation 2 does the comparison reverse (Tables 8 and 3). A reader is entitled to read that history as evidence that the plan prior, not the bandit machinery, is what carries the method—which is our own conclusion.

The multi-fidelity machinery has a precondition that a real benchmark violates. Equation 7 can only resolve gaps larger than $s\sqrt{2\log(\cdot)/D_t}$, so a ladder whose total allocation is small—three training repetitions, in Section 5.3—makes the bounds vacuous relative to the gaps and BLDS-AF worse than doing nothing. Whether agent ladders reach usable D_t is unmeasured.

BLDS-AF's advantage rests on a condition we have checked in exactly two places. Proposition 6 is a bias–variance argument, not a guarantee, and its constants are informal; the empirical claim is two substrates, one on each side of the boundary. The factored surrogate also breaks Lemma 7 whenever its composed candidate falls outside the ball, so BLDS-AF inherits none of Theorem 3—and since composition contributes nothing measurable, a reader may reasonably prefer the ball-respecting variant and keep the analysis.

The mechanism we originally proposed for our largest effect was wrong. Localization helps by targeting the broken node (Proposition 18), not by contracting the ball's cardinality (Remark 17), and the corrected account is linear rather than exponential in the cone fraction—so the component's theoretical leverage is smaller than we claimed, even though its measured benefit is the largest we found. The correction rests on one instrumented quantity, the repair-hit rate, and a different instrumentation might support a third account.

Theorem 3 has no measured predictive content (Section F.11), and θ has no measured effect (Section F.9). The formal apparatus that motivates bounded-discrepancy search is therefore not the apparatus that explains the observed behaviour; what carries the method empirically is the plan prior, which needs no theorem, and the acceptance rule, whose value showed up in robustness rather than speed.

The verifier is the binding constraint. Every guarantee is conditional on R being bounded and low-bias. Where a programmatic test suite is unavailable an LLM judge substitutes a noisy and possibly systematically biased signal, and the theory degrades accordingly rather than gracefully.

Assumption 13 is measurable only on graphs small enough to enumerate; extrapolation to large graphs is an inference. Statefulness forces isolated re-execution, whose overhead is real and which we report rather than remove. Agents that coordinate through the filesystem violate Assumption 5 and hence both localization propositions, at a cost Table 18 quantifies. Finally, Proposition 15 establishes reachability, not convergence: like BLDS, BLDS-AF is a local search with restarts and we make no completeness claim.