When Does Learned Search Guidance Transfer and Pay for Itself?

Noah Mitchell
University of Ottawa

Preprint, November 2026

Abstract

A recurring goal in reasoning research is a small, reusable controller that decides which branch of a search to expand next, generalizes to new problem families, and saves more computation than it costs. We separate this goal into five claims of increasing strength (generalization to new instances, extrapolation to larger instances, generalization to new underlying functions, transfer across domains or rapid online amortization, and savings in total compute) and test them in two regimes. In exact combinatorial search (SAT, Boxoban, Blocksworld and arithmetic program synthesis), learned guidance raises solve rate at a fixed number of expansions by 4.5 to 54.8 percentage points and extrapolates to larger instances. The stronger claims fail there. A 49.1-point synthesis gain comes entirely from functions already present in training and vanishes on new ones, a scheduler pretrained on SAT and Boxoban does not transfer to synthesis, and the uncached SAT guidance is 11.6 to 80.2 times slower than the baseline. Over a frozen 4B language model, which offers a shared representation across domains, three external controllers also fail to save total compute. A fine-tuned 0.6B controller does not beat a fixed careful chain of thought, even when trained in-domain. Pruning branches by the model's hidden states or confidence never beats unpruned sampling at equal token cost. Online amortization across a stream of code-repair trees gains under a point over a non-amortized tree search, although the hidden states do predict which failed attempts can be repaired. We also show that training a controller to imitate hindsight shortest paths is inconsistent even with unlimited data, and we observe the predicted failure in practice. The regimes differ in step granularity, in how much a single state reveals about distance to a solution, and in the cost of guidance relative to the work it saves; we offer these as testable conditions for where learned search control should pay.

1 Introduction

Search procedures spend most of their effort deciding which open partial state to expand next. When a solver meets many related problems, earlier searches should inform later ones: a model fitted to past search outcomes could recognize dead ends, prefer productive branches and solve more problems within a fixed budget [2, 4]. This idea runs from learned guidance for combinatorial search [12, 13, 16, 17] to recent controllers that allocate test-time compute for language models [1, 6, 18].

The idea bundles several claims that are easy to conflate. We separate five:

C1 *Instance generalization.* Guidance fitted on a training split improves solve rate at a fixed number of expansions on unseen instances from the same generator.

C2 *Extrapolation.* Guidance fitted on smaller or simpler instances keeps its gain on larger or structurally shifted instances of the same domain.

C3 *Function novelty.* Guidance keeps helping when test problems require genuinely new underlying functions, not new observations of functions seen in training.

C4 *Transfer and online amortization.* Experience from other domains, or from earlier problems in a stream, improves search on a new domain without offline training on it.

C5 *Total-compute savings.* The computation saved by better allocation exceeds the cost of computing and updating the guidance.

Each claim is stronger than the one before it, and evidence for one says little about the next. A large result on C1 does not imply C3 to C5, and a failure on a small test does not show which obstacle was binding: a missing shared representation, a misaligned training objective, noise in the search substrate or the controller's own overhead.

We test the five claims in two regimes under one accounting framework (Section 2). Regime I (Section 3) is exact combinatorial search in SAT, Boxoban, Blocksworld and arithmetic program synthesis, where each domain has its own learned model. Regime II (Section 4) asks whether a pretrained language model, which gives one representation for all domains, removes the obstacles found in Regime I; we test three external controllers over a frozen Qwen3-4B model [14]. Table 1 summarizes the outcome.

Our contributions are:

- A decomposition of the learned-search-control goal into five claims, and a test of each in two regimes under a common accounting of expansions and controller cost.
- Evidence that in exact search, learned guidance satisfies C1 and C2 with large margins, together with three audits that locate where it stops: an exact canonical-form audit showing that the program-synthesis gain comes from functions already seen in training, a transfer test of a domain-agnostic scheduler, and a wall-clock measurement.
- Three controller designs over a frozen language model, with confidence bounds on the largest gains the data allow, showing that a shared pretrained representation does not rescue C4 and C5 for lightweight external controllers. Two analyses explain part of the failure: an apparent cross-domain alignment of verification signals comes from truncation, and a real repairability signal in the hidden states has too little room to become a search gain.
- A proposition showing that imitating hindsight shortest paths is inconsistent for search control, as an instance of the imitation gap for privileged teachers [20], and an observation of the predicted failure in a language-model controller.

2 Setup

Persistent-frontier search. Let D_t be the set of discovered states after t executed computations. The persistent frontier is the set of untried actions at discovered states, $\mathscr{F}_t = \{(s,a) : s \in D_t,\ a \text{ available and untried}\}$. At each step a controller π chooses an element of $\mathscr{F}_t$ using only information observed so far, executes it with the exact environment or a frozen language model, observes the outcome, and adds any newly revealed actions to the frontier. Failed attempts and duplicate states consume budget, and untried alternatives stay available, so the controller can return to an earlier branch at any time.

Capability and cost. For a domain d with budget grid $\mathscr{B}_d$, let $A_d^\pi(B)$ be the probability that π finds a verified solution within budget B, and let the primary score be the average $P_d(\pi) = |\mathscr{B}_d|^{-1} \sum_{B \in \mathscr{B}_d} A_d^\pi(B)$. If N_π counts executed expansions (or generated tokens) at unit cost c, and G_π counts all controller inference, feature extraction and online updates, the total cost is

$$C_\pi = N_\pi c + G_\pi. \tag{1}$$

Table 1: The five claims in the two regimes. Gains are solve-rate differences in percentage points against the strongest non-learned control at matched budgets.

Claim	Regime I: exact combinatorial search	Regime II: controllers over a frozen LLM
C1 Instance generalization	Holds: $+28.1$ SAT, $+4.5$ Boxoban, $+54.8$ Blocksworld	Not shown: a controller trained in-domain is -0.2 [-3.2, $+2.8$] against a fixed schedule
C2 Extrapolation	Holds: $+30.0$ on 48-variable and $+26.7$ on community-structured SAT; Blocksworld from 8 to 10 blocks	Not tested
C3 Function novelty	Fails: $+49.1$ on familiar functions, -0.4 on new ones (size also changes)	Not tested
C4 Transfer and online amortization	Fails: a scheduler pretrained on SAT and Boxoban adds $+0.04$ to $+0.42$ on synthesis	Fails: probe directions are orthogonal across domains; online amortization adds $+0.7$ [-1.2, $+2.7$]
C5 Total-compute savings	Fails as implemented: uncached SAT guidance is 11.6 to 80.2$\times$ slower in wall-clock time	Fails: every pruning policy is 0.8 to 3.0 below unpruned sampling at equal tokens; scoring costs erase expansion gains

C1 to C4 are claims about P_d at fixed N_π. C5 is a claim about P_d at fixed C_π: a controller improves the capability–compute frontier only if it beats non-learned controls when its own cost is counted.

Hindsight supervision. Offline datasets of search graphs make it tempting to label, for each recorded state, the action on the cheapest path to a solution in that recorded episode. The quantity a controller should estimate is instead the value of an action given only what has been observed,

$$Q^*(h,e,b) = \mathbb{E}\big[V^*(h \oplus (e,o), b - c(e,o)) \,\big|\, h, e\big],$$

for public history h, candidate e, outcome o and remaining budget b. A hindsight teacher sees the realized outcomes, which the controller cannot. This is the setting of imitation from a teacher with privileged information, where the student's policy marginalizes over what the teacher knew and can be arbitrarily worse than the best policy available to the student [19, 20]. In search, the gap takes a specific form: the teacher never labels information-gathering actions as optimal.

Proposition 1 (Hindsight shortest-path imitation is inconsistent)**.** *Consider a task with four equally likely hidden answers $y \in \{1,2,3,4\}$, unit-cost guesses $g_1, \ldots, g_4$, a unit-cost diagnostic action d that reveals y without solving the task, and budget $b = 2$. A teacher that labels the first action on the cheapest realized solution path always labels g_y and never labels d. A controller that imitates these labels perfectly, with unlimited data, solves the task with probability $1/2$. The optimal controller given public information executes d and then g_y, and solves it with probability 1.*

Proof. In every episode the cheapest solution path is g_y alone, at cost 1, while any path through d costs 2, so the teacher's root label is g_y. Since y is uniform and unobserved at the root, the Bayes-optimal imitator of these labels assigns probability $1/4$ to each guess and never chooses d. After a failed guess, the teacher's label is again the correct guess, so the imitator chooses uniformly among the three untried guesses. It succeeds with probability $1/4 + (3/4)(1/3) = 1/2$. Executing d reveals y, after which g_y succeeds with probability 1 within the budget. □

An objective that rewards immediate success fails in the same way, since d earns no immediate reward while each guess earns $1/4$. In stochastic language-model search the analogue is that whenever a cheap,

Table 2: Primary solve rate (%, averaged over each domain's primary budget grid) for the non-learned control, three random-weight initializations of the learned model (mean, with the range across seeds) and the learned model. Differences in percentage points with paired 95% bootstrap intervals over tasks.

Domain	n	Control	Random weights	Learned	Difference (pp)
SAT, 36 variables	1,500	56.27	24.60 (9.7–48.6)	84.40	+28.13 [+26.54, +29.79]
SAT, 48 variables (larger)	1,000	35.29	–	65.25	+29.96 [+27.62, +32.26]
SAT, 36 variables, community	1,000	47.12	–	73.79	+26.66 [+24.79, +28.43]
Boxoban (hard)	600	15.25	13.35 (11.2–14.9)	19.79	+4.54 [+2.17, +6.83]
Blocksworld, 8 → 10 blocks	120	44.38	42.08 (37.9–47.7)	99.17	+54.79 [+48.33, +62.29]

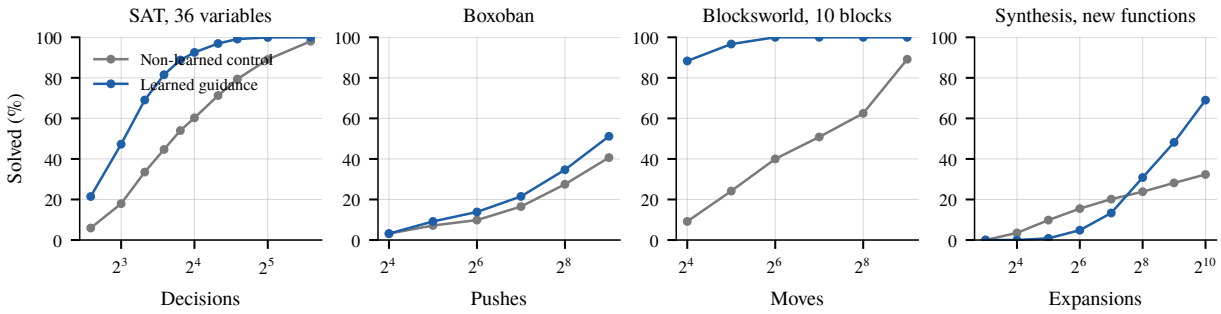

Figure 1: Solve rate against budget for the learned guidance and the non-learned control. In SAT, Boxoban and Blocksworld the learned guidance is ahead at almost every budget. On synthesis targets that compute functions absent from training, it is behind at small budgets and ahead at large ones, and the two average out.

unreliable draw happens to succeed on a training trace, the hindsight teacher marks the reliable action as wrong and trains the controller to gamble. Section 4.1 shows this happening.

Statistics. Solve-rate differences are paired across tasks and reported with 95% bootstrap intervals from 2,000 resamples of tasks. For null results the relevant quantity is the largest gain the data allow, so we also give one-sided 95% upper bounds.

3 Regime I: exact combinatorial search

We study four exact domains: finding satisfying assignments for random 3-SAT formulas, Boxoban levels [3], goal-conditioned Blocksworld and arithmetic program synthesis from input–output examples. Each domain has its own learned model inside the same persistent-frontier search [11]. In SAT, a recurrent graph network trained on solver traces predicts the consequences of a branching decision. In Boxoban, a convolutional model corrects a matching-distance heuristic. In Blocksworld, a Deep Sets model over block-support relations predicts excess remaining cost. In synthesis, an encoder over the input–output examples and a recurrent model over grammar productions score partial programs, and the score is added to a non-learned heuristic. Each domain also has a calibrated non-learned control and three random-weight initializations of its learned model. Models and deployment settings were frozen before the evaluations reported here (Appendix A).

Table 3: Three audits in exact search. (A) Program synthesis split by whether the target's canonical polynomial occurs in training. (B) Offline top-1 ranking accuracy on synthesis for a scheduler pretrained on SAT and Boxoban, against the same architecture trained from scratch, averaged over three paired seeds. (C) Wall-clock time summed over the SAT budget grid, with model inference uncached.

Audit	Cohort	Control	Learned	Difference
A. Synthesis	New examples (800/800 familiar)	35.05%	84.16%	+49.11 [+46.43, +51.52] pp
	Operator shift (595/600 familiar)	23.43%	78.67%	+55.24 [+52.81, +57.71] pp
	New functions, size 9 (0/600 familiar)	14.43%	14.00%	−0.43 [−2.69, +1.90] pp
B. Scheduler transfer	25 target tasks	60.99%	61.41%	+0.42 pp (seeds: +0.04, +0.56, +0.65)
	100 target tasks	61.87%	61.92%	+0.04 pp (seeds: −0.39, +0.52, +0.00)
	400 target tasks	61.94%	62.09%	+0.14 pp (seeds: +0.86, −0.52, +0.09)
C. SAT wall-clock	36 variables	74.5 s	863.4 s	11.6× total, 15.9× per decision
	Community structure	49.0 s	732.9 s	15.0× total, 19.0× per decision
	48 variables	78.4 s	6,284.5 s	80.2× total, 103.1× per decision

3.1 Instance generalization and extrapolation hold

Learned guidance improves the primary solve rate in SAT, Boxoban and Blocksworld (Table 2), and its advantage is positive at 21 of 22 evaluated budgets and tied at the remaining one (Figure 1). At 16 SAT decisions the solve rate rises from 60.27% to 92.53%; at 512 Boxoban pushes, from 40.67% to 51.17%. In ten-block Blocksworld, value guidance alone reaches 98.96%. Each learned model beats every random initialization of its architecture, so the gain comes from training rather than from the architecture or the search procedure. The SAT model keeps its gain on larger and structurally shifted formulas without retraining, and the Blocksworld model trained on eight blocks keeps it on ten. C1 and C2 hold.

3.2 Function novelty fails

The synthesis model's gain of 49.1 points on held-out tasks looks like C1. To ask whether it reflects C3, we expanded every expression into its canonical polynomial in the inputs (x, y). The 6,000 training tasks contain only 349 distinct functions, and every one of the 800 primary test targets, and 595 of the 600 operator-shift targets, computes one of them. The test tasks present new input–output examples of familiar functions. On 600 targets that compute functions absent from training, the gain disappears (-0.43 points; Table 3A). The average hides a crossover (Figure 1): the learned model is worse at small budgets (4.83% against 15.50% at 64 expansions) and better at large ones (69.00% against 32.33% at 1,024).

The new-function cohort also uses larger expressions (nine nodes), so this audit cannot separate novelty from size. It does show that the synthesis gain on the standard split is not evidence for C3, and that a split defined by held-out examples can hide that every test target was seen in training.

3.3 Transfer fails, and guidance is expensive

The domains in Regime I use incompatible encoders, so transfer needs a representation that does not depend on the domain. We pretrained a Deep Sets frontier ranker on 30,000 snapshots from SAT and Boxoban, using 21 per-candidate features that describe predictions and search context rather than domain content, and adapted it to synthesis with 25 to 400 target tasks. Against the same architecture trained from scratch on the same target data, pretraining changes offline top-1 accuracy by $+0.04$ to $+0.42$ points, with mixed signs across seeds at 100 and 400 tasks (Table 3B). A related test on Boxoban and Blocksworld, with a scalar risk rule $q + p/(1-p)$ combining predicted value q and failure probability p, beat value-only search by 2.28 points on 540 unseen tasks but tied the native novelty rule (-0.03 points), and optimizing the rule offline

Table 4: Study 1: primary solve rate over the three budgets, and the paired difference from the fixed long chain of thought on the same tasks, with 95% bootstrap intervals over tasks. Controllers trained on other categories are evaluated on the held-out category (all 192 tasks across the four folds) or on held-out tasks from their training categories. †Not part of the released per-task results.

Policy	Evaluated on	Primary (%)	Difference vs. long only (pp)	$B = 512$	$B = 2{,}048$
Long chain of thought only (fixed)	all 192	45.14	–	25.52	64.06
One fast attempt, then long (fixed)	all 192	45.31	+0.17 [−3.12, +3.65]	25.52	62.50
Repair after a fast attempt, then long (fixed)	all 192	43.92	−1.22 [−4.87, +2.78]	25.52	61.46
Best of n fast attempts (fixed)	all 192	22.92	−22.22 [−28.65, −15.80]	15.62	27.08
Broad training (96 tasks)	excluded category	42.71	−2.43 [−4.51, −0.17]	21.88	61.20
with prefill charged	excluded category	39.32	−5.82 [−8.16, −3.56]	19.01	59.38
Broad training (96 tasks)	held-out source tasks	45.57	−0.78 [−4.17, +2.86]	26.30	65.62
Narrow training (16 tasks)	excluded category	40.19	−4.95 [−7.03, −2.95]	20.57	60.16
Trained in-domain (reference)	own category (CV)	44.97	−0.17 [−3.21, +2.78]	21.88	64.58
with prefill charged	own category (CV)	41.49	−3.65 [−7.12, −0.35]	18.23	63.54
Label-permuted training (96 tasks)	excluded category	35.16	−9.98 [−13.19, −6.86]	23.44	50.78
Untrained controller†	all 192	37.67	–	13.54	63.02
Hindsight tape oracle† (privileged)	all 192	56.77	–	32.81	80.73

drove Blocksworld to 0% by selecting frontier items outside the support of the training data. C4 fails in exact search.

C5 also fails as implemented. The SAT guidance executes fewer decisions than the control (133,993 against 183,446 on the in-distribution cohort) but takes 11.6 to 80.2 times longer, or 15.9 to 103 times longer per decision, because the graph network is re-run on every frontier evaluation without caching (Table 3C). This is partly an implementation choice; Section 7 discusses what caching could recover.

4 Regime II: external controllers over a frozen language model

The transfer failure in Regime I has an obvious candidate cause: SAT, Boxoban and Blocksworld share no representation. A pretrained language model offers one representation for many domains, and its expansions (generations of hundreds of tokens) are expensive enough that a small controller might pay for itself. We test three controllers over a frozen Qwen3-4B-Instruct-2507, each removing an obstacle exposed by the previous one.

4.1 Study 1: a fine-tuned text controller

We use 192 procedural reasoning tasks from 16 generators in four categories (numeric, strings, spatial and structured), at proposer-token budgets of 512, 1,024 and 2,048. A persistent frontier exposes six macro-operations: a concise answer (fast), an alternative answer (diverse), working notes without an answer (work), an answer conditioned on those notes (final), a revision conditioned on verifier feedback (repair) and a careful chain of thought using the remaining budget (long). A Qwen3-0.6B controller with a rank-8 LoRA reads the task, the history, the feedback and the legal operations, and outputs one operation. It is trained for 128 updates on 1,943 snapshots labelled by the cheapest successful continuation in recorded search graphs. To test transfer, each of the four categories is held out in turn: the controller is trained on 96 ("broad") or 16 ("narrow") tasks from the other three categories and evaluated on the held-out category, and separately on held-out tasks from the training categories. An in-domain reference controller is trained and evaluated by three-fold cross-validation within each category. In proxy mode the controller's own tokens are charged against the budget at 0.19 proposer tokens each. We completed 64 of 108 planned cells before stopping the study; the uncompleted cells are additional seeds of conditions reported here.

Table 5: Study 2: cosine similarity between the correctness directions of probes trained on different domains, at layer 36 (layer 27 for finished traces). The self-check question aligns math and logic on the full set of traces but not on finished traces.

Traces and representation	Math–Code	Code–Logic	Math–Logic
All traces, 25% prefix, generation state	−0.003	−0.002	+0.109
All traces, 25% prefix, after self-check question	−0.028	−0.038	+0.628
All traces, 50% prefix, after self-check question	−0.013	+0.016	+0.599
Finished traces, 50% prefix, generation state	+0.001	+0.022	+0.038
Finished traces, 50% prefix, after self-check question	−0.002	+0.018	−0.022

No trained controller beats the fixed long chain of thought (Table 4). The in-domain reference controller scores −0.17 points relative to it [−3.21, +2.78], so the data rule out a gain larger than +2.34 points (one-sided 95% bound), and charging the controller's own prefill makes it significantly worse. Controllers trained on other categories are worse still on the held-out category. The privileged hindsight oracle reaches 56.77%, so better allocation is possible in principle.

Proposition 1 predicts one reason the controllers do not find it. At $B = 2{,}048$ the hindsight teacher labelled the fast operation optimal 58 times and the long chain of thought 64 times, although repeated fast attempts solve 27.08% of tasks and the long chain solves 64.06%. Lucky fast successes on training traces taught the controller to gamble. A second cost is prefill: re-reading the serialized history through a separate model costs 3.4 to 3.5 points under the proxy charge. In a 16-task code-revision reference, a 4B controller used 7,927 prefill tokens to steer 2,865 proposer tokens and scored 3.1 points below a fixed alternation.

4.2 Study 2: probes on the model's own hidden states

Study 2 removes the external controller's prefill by reading signals from the frozen model's own computation, and asks whether a verification signal learned in one domain carries over to another. We sampled 8 solutions (temperature 0.7, at most 1,280 tokens) for 80 tasks in each of three verifiable domains, for 1,920 trajectories: MATH-500 levels 4 and 5 [5, 8], MBPP+ [10] and ZebraLogic grid puzzles of size 3×3 to 4×3 [9]. At 25, 50, 75 and 100% of each trajectory we recorded token log-probabilities and entropies, the hidden state at layers 18, 27 and 36, and, by appending a short self-check question to the existing KV cache, the hidden state and the yes–no logit margin after the question (38 tokens per check). We trained ridge probes and a set model that compares the 8 branches of a task, all evaluated leave-one-domain-out, and compared them with DeepConf's training-free confidence scores [1].

A shared backbone does not produce a shared correctness direction. Probe directions from different domains are orthogonal ($|\cos| < 0.04$) except between math and logic after the self-check question (Table 5). That alignment comes from truncation: half of math trajectories and 72% of logic trajectories hit the length limit, truncated trajectories are almost never correct (3.4% and 0%) and finished ones almost always are (99.4% in both), and the alignment vanishes when only finished trajectories are used. On code, where almost every trajectory finishes and failures are real bugs, the learned probes rank solutions at chance (AUROC 0.47 to 0.51 at the 50% prefix). There, DeepConf's training-free lowest-group confidence does better (0.687), as it does on math (0.921 against at most 0.56 for the learned probes); on logic, the probe after the self-check question is the stronger signal (0.932 against 0.881).

Under total-token accounting no pruning policy helps (Figure 2 and Table 6). Every policy is below unpruned sampling at equal cost, by 0.8 to 3.0 points. The most favourable, the set model on hidden states, is −0.8 [−2.7, +0.9] points.

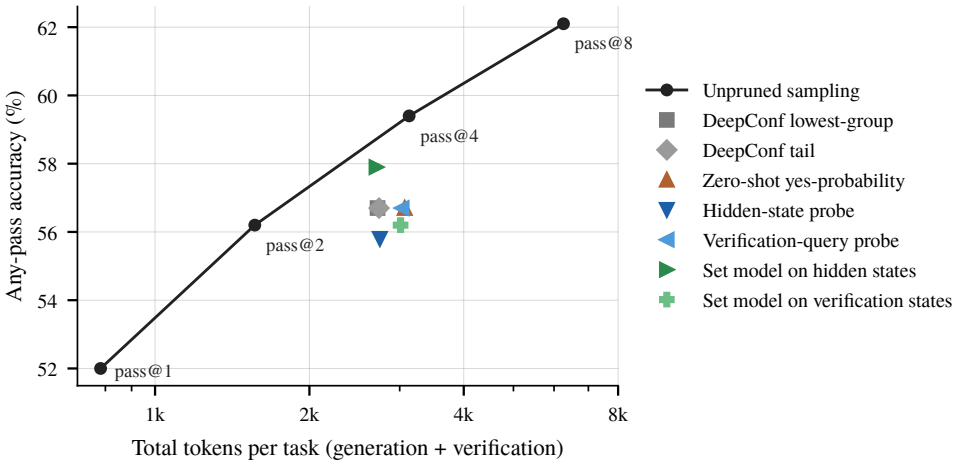

Figure 2: Study 2: accuracy against total tokens per task. Pruning policies sample 8 branches, keep 2 after 25% of generation and count a task solved if a kept branch is correct; verification tokens are charged. Every policy lies below unpruned sampling.

Table 6: Study 2: pruning 8 branches to 2 after 25% of generation, against unpruned sampling at the same token cost (interpolated in log tokens between pass@2 and pass@4). Paired 95% bootstrap intervals over the 240 tasks.

Pruning score	Tokens per task	Accuracy (%)	Unpruned at equal tokens (%)	Difference (pp)
Set model on hidden states	2,702	57.9	58.7	−0.8 [−2.7, +0.9]
DeepConf lowest-group	2,716	56.7	58.7	−2.1 [−4.5, +0.1]
DeepConf tail	2,734	56.7	58.8	−2.1 [−4.3, +0.0]
Verification-query probe	3,028	56.7	59.3	−2.6 [−4.8, −0.8]
Zero-shot yes-probability	3,068	56.7	59.3	−2.7 [−4.9, −0.6]
Hidden-state probe	2,744	55.8	58.8	−3.0 [−5.2, −0.9]
Set model on verification states	3,012	56.2	59.2	−3.0 [−5.0, −1.1]

4.3 Study 3: online amortization on code-repair trees

Study 3 replaces static probes with fast online learning within a stream of real search trees, where verifier feedback is available at every node. For 120 LeetCode problems released after mid-2024 [21] (25 easy, 65 medium, 30 hard), we built complete three-level repair trees with per-assertion execution feedback: 4 initial attempts, 2 feedback-conditioned repairs of each failed attempt and 1 repair of each failed first repair, for 1,743 nodes. Repair helps: pass@1 is 31.25%, pass@4 is 40.83% and the full tree solves 55.83%. We replayed persistent-frontier search over 30 random orderings of the stream at budgets of 2 to 12 generations, comparing fixed schedules, non-amortized AB-MCTS [6], an online nearest-neighbour value estimate, and online Bayesian ridge regression, updated exactly after each observation, on 12 verifier statistics or on a 64-dimensional random projection of layer-18 hidden states plus those statistics. Scoring a node through its hidden state requires prefilling its repair prompt, and we charge that prefill whenever the node is scored but never repaired.

Amortization adds little (Figure 3 and Table 7). The best online learner, ridge regression on hidden states, is +0.73 [−1.17, +2.66] points above AB-MCTS (one-sided 95% upper bound +2.34), and over the second half of the stream, where it has seen the most data, its advantage is 0.61 points. Its scoring prefill raises the token cost at budget 6 from 6,308 to 8,138. Even an offline model trained on 80% of the tasks is only +1.41 [−0.63, +3.54] points ahead, so the ceiling for amortization in this setting is low.

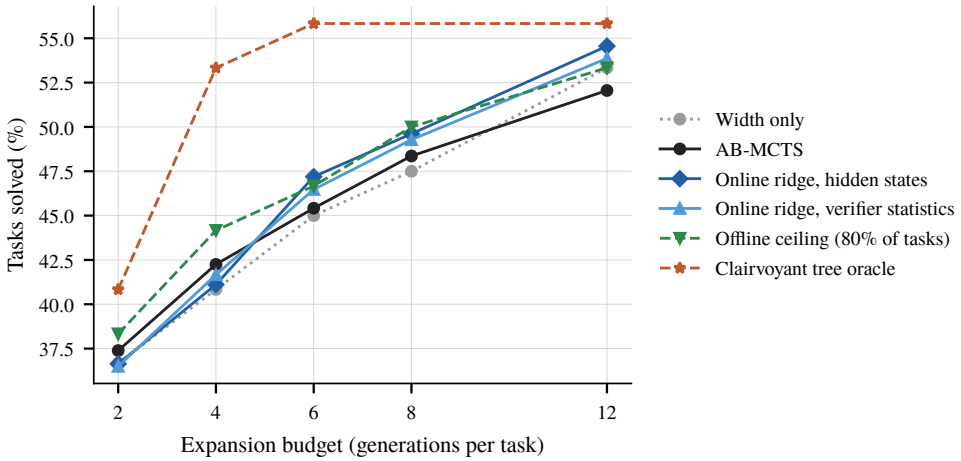

Figure 3: Study 3: tasks solved against expansion budget, averaged over 30 stream orderings. The clairvoyant oracle takes the shortest path to a solved node in each recorded tree.

Table 7: Study 3: primary solve rate over budgets 2 to 12, paired difference from AB-MCTS with 95% bootstrap intervals over the 120 tasks, and mean total tokens per task at budget 6 including scoring prefill.

Policy	Primary (%)	Difference vs. AB-MCTS (pp)	Tokens at $B = 6$
Width only (fixed)	44.67	−0.43 [−1.87, +1.07]	6,127
Depth first (fixed)	45.17	+0.07 [−1.32, +1.47]	6,373
Pass-rate best first (fixed)	43.17	−1.93 [−4.15, −0.22]	6,364
AB-MCTS (no amortization)	45.09	–	6,308
Online nearest neighbours	45.20	+0.11 [−1.21, +1.45]	7,945
Online ridge, verifier statistics	45.56	+0.46 [−1.04, +1.93]	6,097
Online ridge, hidden states (mean)	45.82	+0.73 [−1.17, +2.66]	8,138
Online ridge, hidden states (UCB)	45.60	+0.51 [−1.11, +2.16]	7,985
Offline ridge, 80% of tasks (ceiling)	46.50	+1.41 [−0.63, +3.54]	7,887

The ceiling is low even though the hidden state predicts which failed nodes can be repaired (Table 8): its AUROC is 0.787 on failed initial attempts and 0.662 within tasks, and on the 31 initial attempts that finished with a wrong answer it is 0.864 (95% interval 0.70 to 0.98). Part of the global figure reflects difficulty and failure type, which the hidden state also encodes: among failed nodes, 26.6% of easy, 9.8% of medium and 4.6% of hard ones are repairable, and among failed initial attempts 28.0% after missing code and 21.4% after a syntax error, against 0% after a timeout. The test pass rate, a natural heuristic, ranks repairability below chance. The signal buys little search. Only 85 of 933 failed nodes (9.1%) are repairable, so a within-task AUROC of 0.66 leaves most top-ranked nodes unrepairable, and at the largest budget a fixed schedule already solves 53.3% of tasks against 55.8% for the clairvoyant oracle; the room is at small budgets (53.3% for the oracle at budget 4, against 42.3% for AB-MCTS), where the controller must pick the one useful node from little evidence.

5 What separates the regimes

Table 9 lists how the regimes differ. With two regimes we cannot say which differences are necessary, but three of them line up with where learned control paid. Large gains appeared when search was deep and

Table 8: Study 3: AUROC for predicting whether a failed node's subtree contains a solution, from task-disjoint cross-validated predictions. "Within task" compares nodes of the same task. The last column restricts to initial attempts that finished with a wrong answer (9 of 31 repairable).

	All failed nodes ($n = 933$)		Failed initial attempts ($n = 330$)		Finished, wrong answer
Predictor	global	within task	global	within task	($n = 31$)
Negative depth	0.687	0.721	–	–	–
Test pass rate	0.468	0.363	0.479	0.338	0.422
Verifier statistics (12)	0.732	0.764	0.563	0.559	0.520
Layer-18 hidden state	0.756	0.599	0.787	0.662	0.864
Hidden state and statistics	0.766	0.657	0.786	0.662	0.843

Table 9: Properties of the two regimes as tested here.

Property	Regime I	Regime II
Step granularity	One deduction, push or move (microseconds)	One generation or repair of 150 to 1,280 tokens (seconds)
Depth and branching	32 to 512 steps with exponential branching	2 to 12 macro-steps; fixed schedules span 43.2 to 45.2%
What one state reveals	Exact state encodes distance to goal	Code correctness at chance (AUROC 0.47 to 0.51); some repairability signal, but only 9% of failed nodes are repairable
Shared representation	None (separate encoders)	Shared backbone, but correctness directions orthogonal across domains
Cost of guidance	Large expansion gains; uncached guidance 11.6 to 80.2× slower	Scoring prefill erases small expansion gains

fine-grained, with tens to hundreds of cheap steps over which unguided search blows up, rather than a handful of expensive generations. They appeared when a single state carried information about distance to a solution that the controller could read within its budget; probes on a 4B model's hidden state rank finished code solutions at chance, and where a signal exists, as for repairability, shallow trees with few repairable nodes leave little to exploit. And they require guidance that is cheap relative to the work it prunes, $G_\pi \ll \Delta N_\pi\, c$ in the notation of Equation (1), either through cached readouts of state on deep trees or through a large ratio between proposer and controller cost.

If these conditions are right, search over formal proofs, such as tactic search in Lean, has fine-grained steps, informative states and an exact verifier, should behave like Regime I. A large proposer paired with a small controller changes the cost ratio and could make C5 attainable even for coarse steps. Moving search and verification inside the proposer through reinforcement learning on verifiable rewards removes the external controller altogether, and is outside the scope of our negative results.

6 Related work

Learned guidance for search. Policy-guided tree search with guarantees [12, 13], learned SAT solving [16] and value- and policy-guided search in games [17] show that learned models can make search far more efficient within a domain. Our Regime I results agree, and our audits ask what that efficiency transfers to. Selecting computations under a budget is a classical problem in metareasoning [4, 15], and amortized inference studies how past inference can speed future inference [2].

Allocating test-time compute in language models. Test-time compute can be allocated adaptively across difficulty [18], across width and depth of a search tree [6], and by filtering traces with the model's confidence [1]. Tree-structured prompting [22] and process reward models [8] give other ways to score partial solutions. We use AB-MCTS and DeepConf as baselines and find that, under full accounting, lightweight external controllers do not improve on them in our settings.

Self-evaluation signals. Language models carry information about their own correctness [7]. Our probing results add that, in a frozen 4B model, the directions that carry it differ across domains, that apparent alignment across domains can come from truncation, and that a real signal need not translate into search gains.

Imitation from privileged teachers. When a teacher acts on information the student lacks, imitation marginalizes over that information and can produce poor policies [19, 20]. Proposition 1 is an instance of this gap for search, in which the privileged information is the realized outcome of untried branches.

7 Limitations

The Regime I systems are single trained models rather than populations over training seeds. The new-function synthesis cohort also changes expression size, so it does not isolate novelty. The SAT wall-clock measurement uses uncached inference; caching or batching would reduce the gap, and we report it as a measurement of this implementation. Study 1 completed 64 of 108 planned cells, and its untrained-controller and oracle rows are not part of the released per-task results. Regime II uses one 4B proposer and cohorts of 120 to 240 tasks; our upper bounds say how large an effect could have gone undetected, and the finished wrong-answer subset in Study 3 has only 31 nodes. We did not test larger proposers with small controllers, or search over formal proofs.

8 Conclusion

Learned search guidance generalizes to new instances and extrapolates to larger ones in exact combinatorial search, by wide margins. The stronger claims fail in every test we ran. Gains on held-out synthesis tasks come from functions already seen in training, and a domain-agnostic scheduler does not transfer. A shared language-model backbone does not change this for lightweight external controllers. Their correctness signals point in different directions in different domains, and online amortization gains less than a point before its scoring costs are charged. The cases where learned control paid were deep, fine-grained searches over informative states with cheap guidance, and those are the settings where we expect it to pay next.

References

[1] Yichao Fu, Xuewei Wang, Hao Zhang, Yuandong Tian, and Jiawei Zhao. Deep think with confidence. In *International Conference on Learning Representations*, 2026.

[2] Samuel J. Gershman and Noah D. Goodman. Amortized inference in probabilistic reasoning. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, 2014.

[3] Arthur Guez et al. An investigation of model-free planning. In *International Conference on Machine Learning*, 2019.

[4] Nicholas Hay, Stuart Russell, David Tolpin, and Solomon Eyal Shimony. Selecting computations: Theory and applications. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2012.

[5] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.

[6] Yuichi Inoue, Kou Misaki, Yuki Imajuku, So Kuroki, Taishi Nakamura, and Takuya Akiba. Wider or deeper? scaling LLM inference-time compute with adaptive branching tree search. In *Advances in Neural Information Processing Systems*, 2025.

[7] Saurav Kadavath et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

[8] Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, 2024.

[9] Bill Yuchen Lin, Ronan Le Bras, Kyle Richardson, Ashish Sabharwal, Radha Poovendran, Peter Clark, and Yejin Choi. ZebraLogic: On the scaling limits of LLMs for logical reasoning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 37889–37905, 2025.

[10] Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by ChatGPT really correct? rigorous evaluation of large language models for code generation. In *Advances in Neural Information Processing Systems*, 2023.

[11] Noah Mitchell. Amortizing search with predictive guidance, 2026. Preprint.

[12] Laurent Orseau and Levi H. S. Lelis. Policy-guided heuristic search with guarantees. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

[13] Laurent Orseau, Levi H. S. Lelis, Tor Lattimore, and Théophane Weber. Single-agent policy tree search with guarantees. In *Advances in Neural Information Processing Systems*, 2018.

[14] Qwen Team. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[15] Stuart Russell and Eric Wefald. *Do the Right Thing: Studies in Limited Rationality*. MIT Press, 1991.

[16] Daniel Selsam, Matthew Lamm, Benedikt Bünz, Percy Liang, Leonardo de Moura, and David L. Dill. Learning a SAT solver from single-bit supervision. In *International Conference on Learning Representations*, 2019.

[17] David Silver et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[18] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. In *International Conference on Learning Representations*, 2025.

[19] Andrew Warrington, Jonathan Wilder Lavington, Adam Scibior, Mark Schmidt, and Frank Wood. Robust asymmetric learning in POMDPs. In *International Conference on Machine Learning*, 2021.

[20] Luca Weihs, Unnat Jain, Iou-Jen Liu, Jordi Salvador, Svetlana Lazebnik, Aniruddha Kembhavi, and Alexander Schwing. Bridging the imitation gap by adaptive insubordination. In *Advances in Neural Information Processing Systems*, 2021.

[21] Yunhui Xia, Wei Shen, Yan Wang, Jason Klein Liu, Huifeng Sun, Siyue Wu, Jian Hu, and Xiaolong Xu. LeetCodeDataset: A temporal dataset for robust evaluation and efficient training of code LLMs. *arXiv preprint arXiv:2504.14655*, 2025.

[22] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, 2023.

A Experimental details

Cost accounting. Table 10 lists what counts as an expansion and as controller work in each setting.

Table 10: Units of N_π and contents of G_π in Equation (1).

Setting	Expansion (N_π)	Controller work (G_π)
SAT	Branching decision with exact unit propagation	Graph network, transition model and frontier controller forward passes; reported as wall-clock time
Boxoban	Push attempt	Convolutional value prediction once per discovered state, matching distance, novelty lookup
Blocksworld	Move attempt	Deep Sets excess-cost prediction once per discovered state, goal mismatch, move-signature lookup
Synthesis	Partial-program expansion	Terminal-probe evaluations (both systems) and production scoring
Study 1	Proposer token	None (proposer mode), or 0.19 proposer tokens per controller token deducted from the budget (proxy mode)
Study 2	Generated token, pruned branches charged up to the cut	None for confidence and hidden-state scores; 38 tokens per self-check question
Study 3	Proposer prompt and generation tokens	Repair-prompt prefill for every node scored through its hidden state but never repaired

Regime I domains. *SAT*: satisfiable random 3-CNF formulas with 36 variables, filtered for difficulty under the reference search; test cohorts of 1,500 in-distribution, 1,000 48-variable and 1,000 community-structured formulas, disjoint from training and calibration; primary budgets $\{8,10,12,14,16,20,24,32\}$ decisions. The control is a calibrated schedule of variable-polarity ordering rules with the same propagation and frontier. *Boxoban*: the public `hard` levels, with 400 training, 100 calibration and 600 test levels checked for overlap under translation, rotation, reflection and player-region equivalence; primary budgets $\{32,64,128,256\}$ pushes. The learned score is $h_{\text{match}}(e) - 0.25\,\widehat{\Delta}(e) + 4R(e)$, where R marks previously seen box configurations; the control sets $\widehat{\Delta} = 0$. *Blocksworld*: goal-conditioned instances with optimal plans of 10 to 16 moves, trained on 8 blocks and tested on 120 ten-block tasks; primary budgets $\{32,64,128,256\}$ moves. The learned score adds predicted excess cost to goal mismatch and a penalty for repeated move signatures; the control is goal mismatch with exact-duplicate deprioritization, the strongest non-learned rule we tested. *Synthesis*: grammar $\{x, y, -1, 0, 1, +, -, \times\}$ with 10 input–output examples per task, 6,000 training and 400 calibration tasks; primary budgets $\{8, \ldots, 512\}$ expansions. The control fills each hole of a partial program with each terminal and takes the smallest normalized error plus 0.02 per node; the learned model adds a length-normalized negative log-probability from an example encoder and a production GRU, with its weight tuned on calibration tasks. Random-weight controls re-initialize the whole learned model three times and keep everything else fixed.

Scheduler transfer. Snapshots: 15,000 fitting and 3,000 validation each from SAT and Boxoban; on synthesis, 4,612 fitting snapshots from 400 tasks and 2,315 held-out snapshots from 200 tasks. Adaptation uses 25, 100 or 400 target tasks with three paired seeds, and the comparison is the same architecture trained from scratch on the same target data.

Study 1. Controller: Qwen3-0.6B in non-thinking mode, LoRA rank 8 ($\alpha = 16$) on query and value projections, 128 updates at batch size 4, AdamW with learning rate 10^{-4}. Labels are the set of minimum-token successful continuations in recorded training graphs (1,943 informative snapshots out of 4,267). The

frontier holds at most 10 items. Thirteen fixed schedules were evaluated; Table 4 shows the strongest and a representative weak one.

Study 2. Verification: symbolic equivalence of the boxed answer for MATH-500, the full EvalPlus test suite for MBPP+, and exact table match for ZebraLogic. The self-check question asks the model to review the reasoning for errors and answer yes or no. Ridge probes use a penalty of 500 on within-task centered features (layer 27, or layers 18, 27 and 36 concatenated). The set model is a two-layer permutation-equivariant transformer over the 8 branches' centered hidden states and 10 relative features, trained with an equal mix of binary and listwise losses. Unpruned pass@m uses the unbiased estimator with cost m times the mean trajectory length.

Study 3. Generation at temperature 0.7 with at most 1,024 new tokens. The test harness instruments up to 40 assertions per problem and returns the pass rate, the error type and the first failing assertion. The 12 verifier statistics are the pass rate, an indicator of a nonzero pass rate, depth, log generation length, log code length, a truncation flag and a one-hot error type. AB-MCTS uses hierarchical Beta–Bernoulli Thompson sampling reset for each problem. Online ridge models use exact rank-one updates; the offline ceiling uses five-fold task-disjoint ridge regression on the full 2,560-dimensional hidden state and the verifier statistics.