CAES: COST-AWARE EXPERIMENT SELECTION VIA ONLINE SHADOW PRICES FOR AUTONOMOUS RESEARCH

Anonymous authors
Paper under double-blind review

ABSTRACT

Research agents built on language models propose many experiments for each task, and running all of them consumes a large amount of compute. Recent methods study how to choose which experiments deserve compute. Cost-aware Bayesian optimization sets in advance the rate at which expected improvement is traded against compute, early stopping pays for a partial run of every candidate, and selectors built for research agents compare candidates only with one another. Each of these methods decides one experiment at a time, implicitly assuming that the decision does not affect the experiments that come after. In practice, compute given to one experiment is no longer available to later ones, whose value is still unknown when the current one is decided. To address this, we propose Cost-Aware Experiment Selection (CAES), which ties these decisions together through a single shadow price on compute. Given the shadow price, we frame the decision for each candidate as a Pandora's box problem with a single box, whose exact solution chooses among skipping the candidate, running it in full, and first running a cheap partial version. The shadow price is updated online, rising when compute is spent faster than the search moves through the candidates and falling when it is spent more slowly. A parameter-free form of this update also admits a distribution-free certificate built with Learn-then-Test, which bounds the probability that the final result falls more than a chosen tolerance behind running everything. We evaluate CAES on TabRepo, LCBench, and AIRS-Bench. Given 5 to 40 percent of the compute needed to run everything, CAES outperforms ten baselines across the three benchmarks, ending closer to the result of running everything in 419 of 520 task-paired comparisons. These results show that cost-aware experiment selection with an online shadow price can guide research agents toward compute-efficient autonomous research.

1 INTRODUCTION

Language model agents now carry out machine learning research end to end, writing code, running it, reading the result, and writing the next version (Lu et al., 2024; Jiang et al., 2025; Chan et al., 2025). What they produce on one task is a stream of candidate experiments, each a configuration to train or a script to execute. Producing a candidate is cheap and getting cheaper, and running one is not. Machine time decides how far an agent gets before a deadline, and an agent that runs everything it thinks of spends most of that time on candidates that return nothing, because the chance that the next attempt improves on the best result so far falls steeply as a search goes on. Before every candidate, the agent therefore has to decide whether that candidate is worth the machine time.

Three lines of work answer parts of it. Cost-aware Bayesian optimization scores a candidate by expected improvement divided by its cost, so that an expensive candidate has to promise more (Snoek et al., 2012), with later work letting the cost exponent decay as the budget is consumed (Lee et al., 2020). Multi-fidelity and early stopping methods start many candidates at low cost and promote the survivors (Li et al., 2018; 2020; Golovin et al., 2017; Domhan et al., 2015). A third line, much newer, selects among the candidates a research agent proposes by asking a language model to compare them or by running a short pilot for each (Zheng et al., 2026; Foster et al., 2026).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

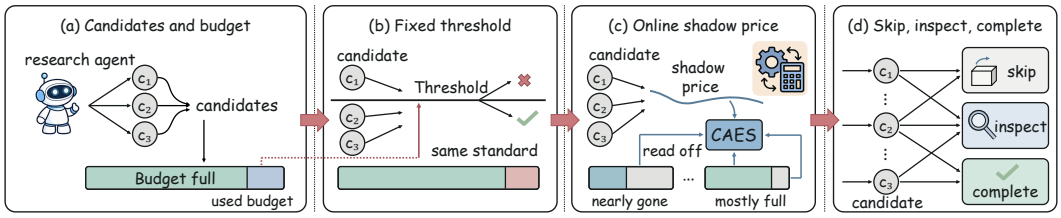

Figure 1: A research agent produces candidates faster than a fixed budget can absorb them. A threshold fixed before the run applies the same standard whether the budget is full or nearly gone. CAES replaces it with a shadow price read off the remaining budget while the stream runs, and the price sorts each arriving candidate into skip, inspect, and complete.

None of these treats the compute budget as the thing that couples the decisions. A cost-aware acquisition function carries a coefficient fixed before the run, so the same coefficient governs a stream with money to spare and a stream that is nearly out. Early stopping pays the first slice of every candidate before learning anything about the rest, the largest possible commitment to a stream whose candidates mostly fail. Agentic selection rules compare candidates against each other and never against the price of the machine time they would consume, so they cannot tighten when the budget runs short. All three lack a quantity that carries the state of the budget into the decision about the candidate in front of the agent. Figure 1 puts the two side by side.

The best score a search reaches is a sum of the improvements its completed candidates make over the incumbent, so a multiplier on the budget moved into the objective gives every step its own term, improvement minus a price times cost. The multiplier is a shadow price on machine time, and it splits the problem in two. Inside, at a fixed price and with the incumbent and the belief of that moment, one candidate that may be declined at no cost and inspected cheaply before the decision is a Pandora problem with a single box and partial inspection, which has an exact solution returning one of skip, inspect, and complete. Outside, the price is whatever makes total spending land on the budget, and since the arrivals are unknown it has to be tracked online, which is a dual update on the multiplier. We call the decomposition Cost-Aware Experiment Selection (CAES).

The decomposition makes each layer testable on its own, and it gives our four contributions. The first is the decomposition itself (Section 2). The second is that the outer layer transfers, since any rule comparing a value against a threshold can take the price in place of its own, and across three benchmarks the pricing layer carries the improvement and moves to expected improvement per unit cost unchanged (Section 4). The third comes from recordings of a live research agent, where the inner layer places its inspections on the candidates whose readings overturn an execution decision that hindsight confirms was wrong (Section 4). The fourth is a distribution free certificate on what declining can cost, built with Learn-then-Test (Angelopoulos et al., 2025), which licenses spending well below the full budget where every parameter carrying alternative is licensed only to run everything (Section 4).

2 Method

2.1 A stream of candidates under a compute budget

An agent works on one task and proposes candidate experiments one after another, each a concrete thing to run such as a configuration to train or a piece of code it has just written. We index them by arrival order $c_1, \ldots, c_T$. Candidate i carries a predicted execution cost k_i in machine time and a score y_i visible only after the candidate has been executed in full, and the best score before it arrives is the incumbent b_i. The first few candidates are executed without judgement so that the predictor of Section 3.3 has something to learn from. For every later one, CAES takes one of three actions. Skipping costs nothing, reveals nothing, and the candidate never returns. Completing it costs k_i and reveals y_i. Inspecting costs fk_i for a fraction $f < 1$ and reveals a noisy reading z_i, after which CAES either pays the remaining cost r_i or declines to finish. An inspection is a cheap partial version of the same experiment, such as one cross-validation fold, the first epochs of a training run, or a run on subsampled data, and a reading never counts as a score. A fold or the first epochs carry over into

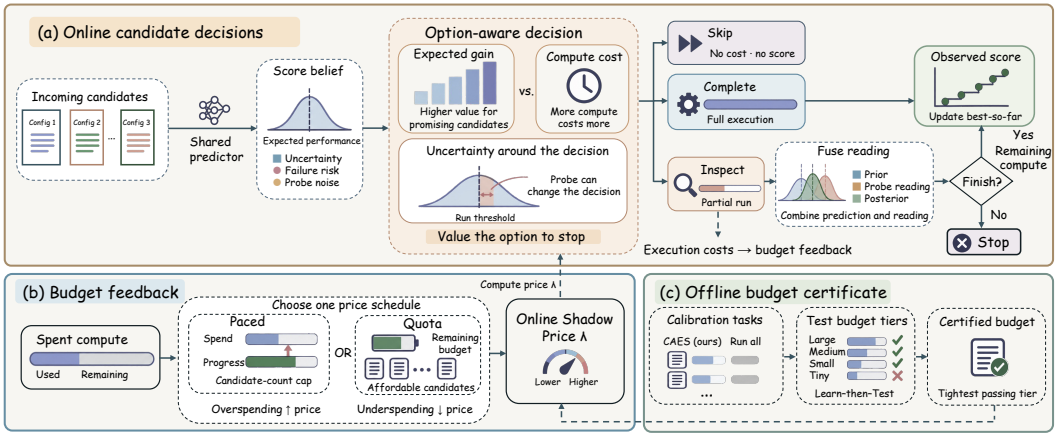

Figure 2: CAES on one task. A shared predictor turns each arriving candidate into a belief, and the current price decides between skipping, inspecting, and completing. A reading is fused with the belief by precision and the decision is taken again on the posterior. Realized spending feeds back into the price. The certificate sits offline and turns per task gaps into a licensed budget.

the full run, so $r_i = (1 - f)k_i$, and a run on subsampled data does not, so $r_i = k_i$. A payment that no longer fits the remaining budget ends the task. Writing a_i for the action on candidate i, $\mathrm{cost}(a_i)$ for what it spends, and B for the budget,

$$\max \max_{i\,:\,a_i=\text{complete}} y_i \qquad \text{subject to} \qquad \sum_{i=1}^{T} \mathrm{cost}(a_i) \le B. \tag{1}$$

2.2 One price for one constraint

The budget is the only thing that couples the decisions. Machine time spent on candidate i is unavailable to candidate $i+1$, and what candidate $i+1$ will look like is unknown when candidate i is decided, which separates the setting from ranking a fixed pool offline. The objective itself splits into one term per step, since each completed candidate raises the incumbent by $(y_i - b_i)^+$ and the final result is $b_1 + \sum_i \mathbb{1}[a_i = \text{complete}]\,(y_i - b_i)^+$, with b_1 the incumbent after the warm start. Attaching a multiplier $\lambda \ge 0$ to the budget and moving it into the objective removes the constraint and gives every step its own term, improvement over the current incumbent minus $\lambda \cdot \mathrm{cost}(a_i)$. The multiplier is a shadow price on machine time, stated in units of score improvement per second. CAES maximizes the expected term of each step under the incumbent and the belief of that moment, and what the action does to later steps reaches them through the price and the incumbent. This is the dual decomposition of online allocation (Balseiro et al., 2023), and it splits CAES into an inner layer that decides a single candidate at a given price and an outer layer that sets the price so that total spending lands on the budget. Figure 2 shows the two together.

2.3 Deciding one candidate at a given price

An inspection is worth buying only when the reading can still change what happens next, and buying one for a candidate whose fate the belief already settles leaves the action unchanged. CAES prices an inspection by the option it creates. A reading has value when it can move the candidate across the line between running and skipping, and that value is available before paying, because the belief already says how the reading is distributed.

The predictor supplies a Gaussian belief $Y_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$ over the score of candidate i, a probability p_i that its code fails and returns nothing, and a reading noise s_i, so that a reading is $z_i = Y_i + \varepsilon_i$ with $\varepsilon_i \sim \mathcal{N}(0, s_i^2)$ independent of Y_i. An inspection runs the same code as the full execution and fails with the same probability p_i, and q is the probability that the full execution succeeds once its inspection has. A candidate whose inspection fails is skipped. Expected improvement over the incumbent is $\mathrm{EI}(b, \mu, \sigma) = \mathbb{E}[\max(Y - b, 0)]$ for $Y \sim \mathcal{N}(\mu, \sigma^2)$. The value of completing candidate

i at price λ is

$$v_i^{\text{run}} = (1 - p_i)\,\mathrm{EI}(b_i, \mu_i, \sigma_i) \;-\; \lambda k_i, \tag{2}$$

and skipping is worth zero. With only these two actions, CAES completes the candidate whenever $v_i^{\text{run}} > 0$, which is expected improvement per unit cost compared against a threshold.

The third action is valued against the same two. Fusing a reading z with the prior by precision gives a posterior with mean and standard deviation

$$\mu_i(z) = \frac{\mu_i/\sigma_i^2 + z/s_i^2}{1/\sigma_i^2 + 1/s_i^2}, \qquad \sigma_i(z) = \left(\frac{1}{\sigma_i^2} + \frac{1}{s_i^2}\right)^{-1/2}. \tag{3}$$

Deciding now is worth $v_i^{\text{now}} = \max(v_i^{\text{run}}, 0)$. Buying the reading first is worth

$$v_i^{\text{inspect}} = (1 - p_i)\,\mathbb{E}_z\Big[\max\left(q\mathrm{EI}(b_i, \mu_i(z), \sigma_i(z)) - \lambda r_i,\; 0\right)\Big] \;-\; \lambda f k_i, \tag{4}$$

where $1 - p_i$ is the probability that the inspection returns a reading, q the probability that the full execution then succeeds, and the expectation runs over the predictive distribution $z \sim \mathcal{N}(\mu_i, \sigma_i^2 + s_i^2)$, evaluated by nine point Gauss-Hermite quadrature. The inner maximum is what makes the inspection worth anything, because it lets CAES stop after a bad reading while paying only for the inspection. CAES inspects when $v_i^{\text{inspect}} > v_i^{\text{now}}$, completes when $v_i^{\text{run}} \geq \max(v_i^{\text{inspect}}, 0)$, and skips otherwise. After a reading z_i it completes when $q\mathrm{EI}(b_i, \mu_i(z_i), \sigma_i(z_i)) \geq \lambda r_i$, the comparison inside the maximum of Equation 4 (Algorithm 1 in Appendix D).

Two properties follow from the form of Equation 4. A confident belief drives $\sigma_i(z)$ toward σ_i and the two values toward each other, so CAES stops paying for readings it will not act on. A candidate far from the price on either side has the same property for a different reason, since no plausible reading moves it across the line. The remaining candidates form a band around the line. On recordings of a live research agent, CAES spends its inspections on the candidates whose reading changes the decision (Section 4). Figure 2a traces the decision for one candidate.

For a single candidate that can be declined at no cost and whose score carries a Gaussian belief, the pair of comparisons above is exact at the given price, incumbent, and belief. The structure is Weitzman's Pandora problem (Weitzman, 1979), in which a searcher pays to open boxes of unknown value and keeps the best value found, extended so that a box may be inspected partially first. Our three actions are that model's three states, and it has an exact threshold solution for a single box while being NP-hard once several boxes are solved jointly (Aouad et al., 2026). Under the relaxation of Section 2.2 every step of the stream is one such box, posed at the incumbent and the belief of that step, so the inner layer solves each step exactly and leaves the joint problem to the price.

2.4 Setting the price from the budget

A price held fixed for the whole stream cannot respect the budget. Set it high and the agent stops before the machine time is used, set it low and the money is gone while candidates are still arriving. The price has to react to how fast the budget is being consumed relative to how far the search has come, which is an online problem because neither the arrivals nor their costs are known in advance. Both schedules below read that signal, raising the price when spending runs ahead of progress and lowering it when spending falls behind. They differ in what stands in for progress and in whether they carry parameters.

The paced schedule measures progress by position in the stream. Writing C_i for the machine time spent before candidate i and T for the cap on the number of candidates,

$$\lambda_i = \lambda_0 \exp\left(\eta\left(\frac{C_i}{B} - \frac{i}{T}\right)\right), \tag{5}$$

clipped to the band $[10^{-3}\lambda_0,\, 10^{3}\lambda_0]$. This is the closed form of dual mirror descent on the multiplier of Section 2.2 under a negative entropy reference function (Balseiro et al., 2023). The schedule carries two parameters, the initial price λ_0 and the step size η, chosen as in Section 3.2.

The quota schedule removes them. It measures progress by what the remaining budget can still buy. Let m_i be the number of candidates still expected, the smaller of the remaining candidate cap $T - i$

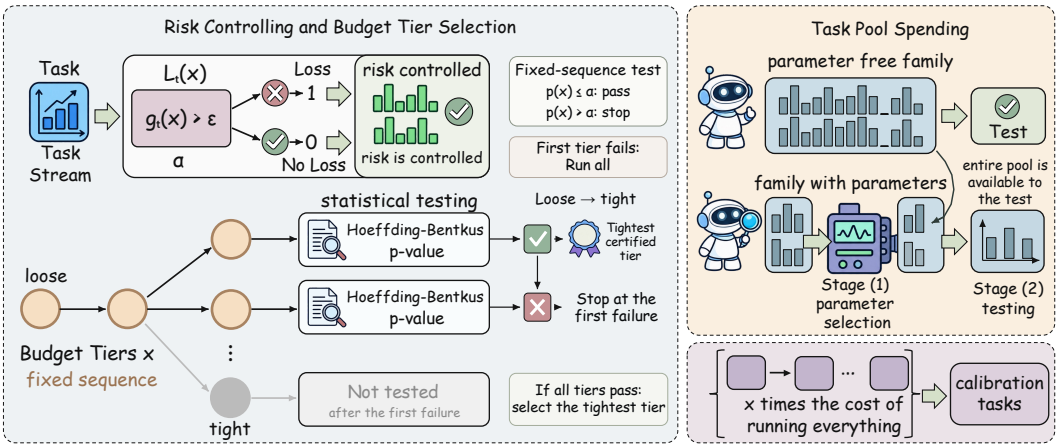

Figure 3: The certificate. Each calibration task contributes a gap against the unbudgeted run, comparing it against the tolerance turns it into a binary loss, and the budget tiers are tested from loose to tight at level α until one fails. The tier before the failure is the one that is certified, and a family that spends no tasks on choosing parameters leaves the whole pool available to the test.

and the remaining wall clock divided by the recent wall clock per candidate, and let $\bar{k}_i$ be the mean predicted cost of a full execution over the arrivals so far. The affordable fraction and the price are

$$\phi_i = \mathrm{clip}\left(\frac{B - C_i}{m_i \bar{k}_i},\, 0,\, 1\right), \qquad \lambda_i = Q_{1-\phi_i}\Big(\big\{(1-p_j)\mathrm{EI}(b_i, \mu_j, \sigma_j)/k_j\big\}_{j \le i}\Big), \tag{6}$$

where Q_τ is the empirical τ quantile over the arrivals since the warm start and $\lambda_i = 0$ whenever $\phi_i = 1$. The quantity inside the braces is the improvement a candidate is expected to return per second of machine time. If the remaining money buys a ϕ_i fraction of what is still coming, the price belongs where exactly that fraction of the stream clears it. When ϕ_i reaches one the rest of the stream is affordable and every candidate is completed without inspection. The schedule leaves nothing for the selection of Section 3.2 to tune, and Section 4 shows that this is what makes the certificate reachable on a small pool of tasks. Appendix D gives its averaging window and start.

The two schedules occupy the same slot and CAES uses whichever the setting supports. Equation 6 needs a wall clock cap to estimate m_i, so it applies where the agent runs against one, and Equation 5 applies where the stream is capped by candidate count alone.

2.5 A CERTIFICATE FOR WHAT SKIPPING COSTS

The certificate states what declining can cost on a new task. Treating tasks as the exchangeable unit, the loss on task t is $L_t(x) = \mathbb{1}[\, g_t(x) > \epsilon \,]$, where $g_t(x)$ is how far behind the unbudgeted run the same stream ends when CAES is given a budget of x times the cost of running everything, and ϵ is the tolerance the user accepts. Learn-then-Test (Angelopoulos et al., 2025) turns a family of candidate configurations into a set whose risk is controlled, by testing one null hypothesis per configuration and correcting for multiplicity. We index the family by the budget tier x, order the tiers from loose to tight, compute a Hoeffding-Bentkus p-value (Hoeffding, 1963; Bentkus, 2004) for the hypothesis that the risk at tier x exceeds δ, and test them in that fixed sequence at level α. The output is the tightest certified tier, and running everything when no tier passes. The guarantee is distribution free and holds on a task drawn from the same population as the calibration tasks. The certified tier is what a new task is given to spend. The budget B of Section 2.4 is that tier times the cost of running everything on the task, and the schedule there turns it into the price at which each decision is taken. Figure 3 draws the sequence.

The quota schedule of Equation 6 matters here for a structural reason. A family with parameters has to spend part of the task pool choosing them before anything can be tested, and the p-values are then computed on what is left. A parameter free family spends none, so the entire pool is available to the test. Where tasks are expensive to produce, this difference decides whether any tier is certifiable.

3 EXPERIMENTAL SETUP

3.1 BENCHMARKS

TabRepo (Salinas & Erickson, 2024) records the score and the training time of 1090 fixed configurations on tabular tasks from a standard suite (Gijsbers et al., 2024), grouped into 117 binary classification, 94 multiclass, and 33 regression tasks. LCBench (Zimmer et al., 2021) records 2000 multilayer perceptron configurations trained for 50 epochs on 34 tasks. On both, a candidate is one configuration, completing it means reading its recorded score, and an inspection is a partial fit worth about one third of the full cost, a single cross-validation fold on TabRepo and the first 17 epochs on LCBench.

AIRS-Bench (Lupidi et al., 2026) gives 14 machine learning research tasks on which an agent writes code, runs it, and is scored on a held out split. We ran a research agent on all 14 with the operator prompts of aira-dojo (Toledo et al., 2025) and recorded every node it produced with its inspection reading, its wall clock, and whether it failed. The agent proposes five sibling candidates from one parent before any is executed, so the choice of what to run is already present in the recording. The recording holds 1364 nodes in 277 sibling groups over 151 hours of wall clock, of which 89 were spent executing experiments, and an inspection subsamples the data at one fifth of the full cost capped at 300 seconds. Replaying against it lets every method face the same arrivals at the same costs.

The agent builds each candidate on its current best result, so a skipped node changes what its descendants mean. When a rule skips a node we let the descendants arrive and re-base them, setting a descendant's score to the rule's own incumbent plus the improvement it showed over its recorded parent, and carrying the offset down the tree. Appendix A reports two other conventions and the direction of every result is unchanged.

3.2 PROTOCOL

Each task receives a hard budget B equal to a fraction x of what executing its entire stream costs, with x in $\{5, 10, 20, 40\}$ percent. The warm start of three candidates is always paid, and a step whose cost exceeds the remaining budget ends the task. The reported quantity is the gap between the best score a rule reaches inside the budget and the best score the same stream reaches with no budget at all, with scores normalized per task before the subtraction, so a gap is dimensionless and lower is better.

Every parameter a rule carries, including ours, is chosen leave one task out, by averaging the gap of each setting over the remaining tasks. Selection is done separately for each combination of benchmark, proposer, and budget, which we call a cell, giving 12 cells per tabular benchmark and four on AIRS-Bench. Inside a cell the two methods are paired by task, and we report both the point estimate and a paired bootstrap over tasks with 2000 resamples.

3.3 BASELINES AND THE SHARED PREDICTOR

Run-all executes in arrival order until the money is gone, and an oracle that knows every score in advance bounds what any rule could reach. Random skip declines at a fixed rate as a sanity check, and small-scale first inspects every candidate and completes it when the reading is no worse than the incumbent. Asynchronous successive halving (Li et al., 2020), median stopping (Golovin et al., 2017), and learning curve extrapolation (Domhan et al., 2015) start candidates and cut the unpromising ones. Expected improvement per unit cost (Snoek et al., 2012) and CArBO (Lee et al., 2020) price compute against expected improvement, the first against a fixed threshold and the second with a cost exponent that decays as the budget is consumed. ForeAgent (Zheng et al., 2026) compares siblings pairwise with a language model, and research preference models (Foster et al., 2026) run a small pilot for each before choosing one.

Within a benchmark every rule receives the same predictor, so no comparison turns on prediction quality. On the recorded benchmarks it is a Gaussian process meta-learned across tasks. On AIRS-Bench it is the agent's own score estimate for the code it has just written, calibrated within the task, with a cross-task prior for the failure probability and the reading noise. CAES uses the quota

Table 1: Left, gap to an unbudgeted run at four budgets given as a percentage of what the whole stream costs, lower is better, TabRepo averaged over its three groups and broken out in Appendix C. Right, CAES against that baseline in cells paired by task, credited by lower mean gap and, in the second column, only when the 95 percent paired interval excludes zero.

	TabRepo				LCBench				AIRS-Bench				CAES wins-losses	
Method	5	10	20	40	5	10	20	40	5	10	20	40	point	boot.
Run-all	.134	.087	.055	.031	.330	.239	.140	.073	.223	.186	**.086**	.085	43-9	30-1
Random skip	.127	.082	.049	.024	.327	.224	.140	.090	.243	.225	.148	.147	42-10	23-3
Small-scale first	.125	.074	.040	**.014**	.301	.179	.094	.049	.197	.196	.196	.195	35-17	17-4
Successive halving	.129	.079	.043	.017	.307	.173	.093	.030	.194	.193	.193	.193	39-13	16-3
Median stopping	.130	.080	.045	.018	.309	.186	.094	.027	.194	.193	.193	.193	41-11	21-2
Curve extrapolation	.127	.078	.046	.021	.298	.187	.089	.031	.194	.193	.193	.192	41-11	21-2
ForeAgent	.125	.080	.048	.043	.264	.121	.091	.093	**.190**	.187	.186	.185	43-9	32-1
Preference model	.141	.093	.059	.031	.317	.200	.129	.072	.247	.189	.187	.187	49-3	39-0
EI per cost	.129	.080	.047	.028	.213	.087	.053	**.023**	.225	.186	.111	.112	42-10	25-4
CArBO	.131	.081	.047	.027	.213	.095	.051	.024	.232	.186	.128	.120	44-8	27-0
CAES	**.119**	**.070**	**.039**	.021	**.183**	**.075**	**.045**	.027	.192	**.167**	.127	**.043**	419-101	251-20

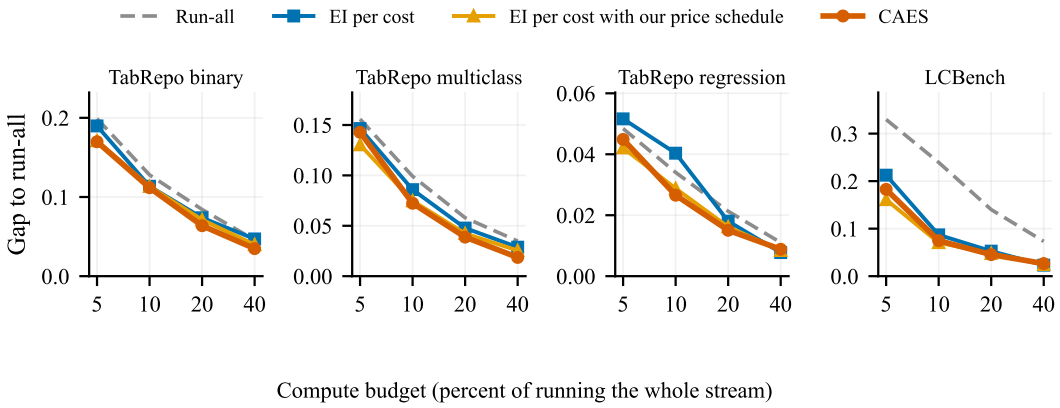

Figure 4: Gap against budget on the recorded benchmarks, averaged over the three proposers. Replacing only the fixed threshold of expected improvement per unit cost with the CAES price schedule moves that rule's whole curve down, so the price schedule transfers to another cost-aware rule.

schedule of Equation 6 on AIRS-Bench, where the agent runs against a wall clock cap, and the paced schedule of Equation 5 on the recorded benchmarks, where the stream is capped by candidate count.

4 Results

Budgeted search improves on all three benchmarks. Table 1 reports the gap of every method at the four budgets, and the margin of CAES is widest where the budget binds hardest. On LCBench at one tenth of the full cost it reaches 0.075 against 0.239 for run-all and 0.087 for the strongest cost-aware baseline. Across the 520 paired cells on the right of the table, CAES has the lower mean gap in 419, and under the bootstrap it wins 251 and loses 20. The two cost-aware baselines are its closest competitors and it still takes 42 and 44 of their cells. An oracle that knows every score in advance appears in Appendix C.

The price layer carries the gain and moves to other rules. Taking expected improvement per unit cost and replacing only its fixed threshold with the schedule of Equation 5, leaving that rule otherwise untouched, wins 18 TabRepo cells and 5 LCBench cells against 3 and 1 (Figure 4). The same schedule added on top of our own fixed price rule wins 19 and 5 against 3 and 0. The price carries the improvement and moves to another rule unchanged.

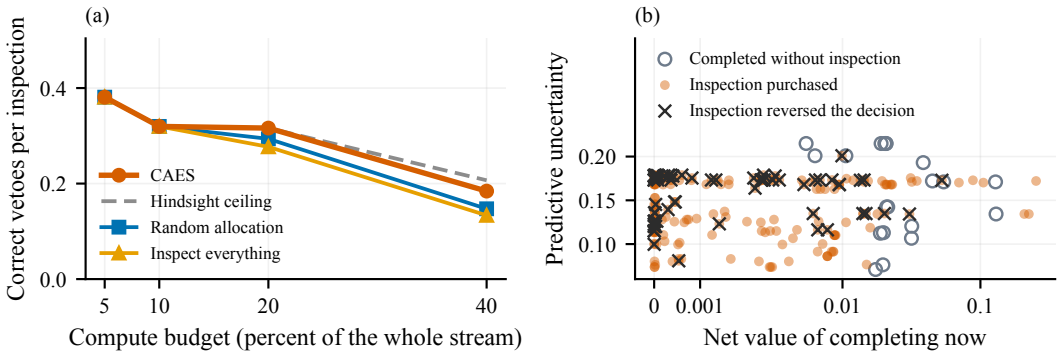

Figure 5: Information acquisition on AIRS-Bench. (a) Correct vetoes per inspection with the inspection spend held fixed and only the choice of recipients varying, against a ceiling that spends the same and picks with knowledge of the recording. (b) Every admitted candidate, placed by the net value of completing it and the width of the belief.

Inspections land where the reading changes the decision. On AIRS-Bench every candidate was actually executed during the recording, so what an inspection would have revealed is known afterwards. We collect the candidates that a reference run without inspections reaches at a positive price, that clear the price, and that have a recorded reading, and ask what the reading does.

Figure 5a holds the inspection spend fixed and varies only which candidates receive one. A correct veto is a reading that turns the action from complete to skip where the recording confirms the candidate was not worth its cost at the price in force. On every task CAES captures at least as many correct vetoes per inspection as an equal cost random allocation and as inspecting every admitted candidate, and at one fifth of the full budget its rate of 0.316 equals what a rule with hindsight could reach. Of the 69 readings that reverse an action, 67 removed a candidate that was not worth its cost and 53 hold under the stricter reading that asks whether it would have improved the incumbent at all.

Figure 5b shows the placement following the geometry Section 2.3 predicts. Among the candidates the price admits, CAES buys a reading for the 193 whose net value of completing sits near zero, at a median of 0.0013, and completes the other 20 at once, whose net value is fifteen times larger at a median of 0.0207. A reading would have overturned 36 percent of the first group and 5 percent of the second. Spreading inspections uniformly over the admitted candidates, or scattering them at random at the same rate, gives a higher gap at all four budgets, so the value of inspection lies in where it is placed.

CAES declines mostly small improvements. A selective rule should decline the candidates that matter least. We compare the share of arrivals a rule declines against the share of the large improvements among them that it declines, where a large improvement is worth more than 0.02 on the normalized scale, and read their ratio. CAES declines between 59 and 80 percent of everything that arrives while declining between 12 and 36 percent of the large improvements, for a ratio between 0.16 and 0.45 across the four budgets. The two rules written for research agents decline far less overall and still lose a larger share of the large improvements, with ratios from 1.15 to 1.49. Pairing by task, CAES protects more than ForeAgent at all four budgets and more than the research preference model at three. Table 3 in the appendix gives the numbers behind these ranges.

A certificate on what selective execution costs. Learn-then-Test certifies a tightest budget tier for each family on AIRS-Bench, at level $\alpha = 0.1$ with tasks as the exchangeable unit. At a tolerance of 0.1 and a risk level of 0.2, CAES is licensed to spend 0.586 of what running everything would take, and at a tolerance of 0.03 it is licensed to spend 0.971, while every other family is licensed only to run everything at every tolerance and risk level we tested. The reason is the one Section 2.5 gives. With fourteen tasks, a family with parameters spends part of them on selection, and the Hoeffding-Bentkus p-value of the remainder never falls below the level, while the parameter free quota price spends none and sends all 13 leave one out tasks to the test. On the recorded benchmarks, where tasks are plentiful, leave one task out verification finds no task whose realized risk exceeds the

certified level, over 72 cells for each of four families without a budget and seven families with one, and the same on LCBench.

5 Related Work

Hyperparameter search moved from random sampling (Bergstra & Bengio, 2012) to Bayesian optimization, which allocates evaluations by maximizing an acquisition value (Shahriari et al., 2016). When evaluations differ in cost, the standard remedy divides expected improvement by the predicted cost (Snoek et al., 2012), and refinements decay the cost exponent as the budget is consumed (Lee et al., 2020) or treat the remaining budget as part of the state (Abdolshah et al., 2019; Astudillo et al., 2021; Lee et al., 2021). Closest to us in spirit, a recent line prices a candidate by the Gittins index of the Pandora problem itself, used as a cost-aware acquisition function (Xie et al., 2024). It scores a candidate whose only action is a full evaluation, where CAES adds the partial evaluation and reads its price off a hard budget the run has to respect. All of these set the exchange rate between improvement and compute before the run. Early stopping instead buys information first, starting many candidates at low fidelity and promoting the survivors (Li et al., 2018; 2020), cutting a run whose partial score trails the history (Golovin et al., 2017), predicting the final score from the beginning of the curve (Domhan et al., 2015), or modelling the relation between fidelities (Falkner et al., 2018; Kandasamy et al., 2017). These methods commit the first slice of every arriving candidate, where CAES treats that slice as one of three actions and buys it only where the reading can change what happens next.

Weitzman's Pandora problem gives the optimal rule for a searcher who pays to open boxes of unknown value and keeps the best one found (Weitzman, 1979). Allowing a box to be inspected partially before the decision to open it changes the structure, and the single box version of that model has an exact threshold solution while the many box version is NP-hard (Aouad et al., 2026). Hardness also holds for the variant where inspection is optional (Beyhaghi & Kleinberg, 2019; Fu et al., 2023) and for correlated values (Chawla et al., 2020). Our inner layer is the exact solution of the single box version, and the relaxation of Section 2.2 poses every step of the stream as that version, which keeps the joint problem and its hardness in the price. Turning a resource constraint into a price and updating it online is the standard treatment of online allocation. Dual mirror descent on the multiplier gives the exponentiated update we adopt (Balseiro et al., 2023), and threshold rules for the online knapsack problem raise the price as the resource is consumed (Zhou et al., 2008). Both assume an additive reward observed once an item is taken, while the objective here is the best score found and the reward of a candidate is unknown when it is decided.

Agents that write and run machine learning code have moved from single scripts to search over a tree of solutions (Lu et al., 2024; Jiang et al., 2025; Toledo et al., 2025; Hambardzumyan et al., 2026), and benchmarks now score them on held out research tasks (Chan et al., 2025; Huang et al., 2024; Wijk et al., 2025; Lupidi et al., 2026). Their search loops propose several candidates per step, which raises the question of which ones to run. ForeAgent compares siblings pairwise with a language model and discards the loser when the comparison is confident (Zheng et al., 2026), and research preference models run a short pilot for each and pick one (Foster et al., 2026). Both compare candidates against each other, neither against the machine time they would consume. Selective execution also needs a statement of what it risks, and distribution free risk control supplies one (Bates et al., 2021; Angelopoulos et al., 2025).

6 Conclusion

An agent under a binding compute budget has to decide where to spend information before it decides where to spend execution, and both answer to the same constraint. CAES routes that constraint through a single number. A shadow price on machine time gives each step its own term, each step is a Pandora problem with one box and partial inspection and is solved exactly, and the price follows spending so that these decisions add up to the global budget. Across recorded configuration streams and recordings of a live research agent this improves budgeted search, the price moves to another cost-aware rule and improves it as well, the inspections land on the candidates whose readings overturn a wrong execution decision, and the certificate states per task what skipping costs.

REFERENCES

Majid Abdolshah, Alistair Shilton, Santu Rana, Sunil Gupta, and Svetha Venkatesh. Cost-aware multi-objective Bayesian optimisation. arXiv preprint arXiv:1909.03600, 2019. URL https://arxiv.org/abs/1909.03600.

Anastasios N. Angelopoulos, Stephen Bates, Emmanuel J. Candès, Michael I. Jordan, and Lihua Lei. Learn then test: Calibrating predictive algorithms to achieve risk control. *The Annals of Applied Statistics*, 19(2):1641–1662, 2025. doi: 10.1214/24-AOAS1998.

Ali Aouad, Jingwei Ji, and Yaron Shaposhnik. The Pandora's box problem with sequential inspections. *Operations Research*, 74(4):2179–2195, 2026. doi: 10.1287/opre.2024.0733.

Raul Astudillo, Daniel R. Jiang, Maximilian Balandat, Eytan Bakshy, and Peter I. Frazier. Multi-step budgeted Bayesian optimization with unknown evaluation costs. In *Advances in Neural Information Processing Systems*, volume 34, pp. 20197–20209. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/hash/a8ecbabae151abacba7dbde04f761c37-Abstract.html.

Santiago R. Balseiro, Haihao Lu, and Vahab Mirrokni. The best of many worlds: Dual mirror descent for online allocation problems. *Operations Research*, 71(1):101–119, 2023. doi: 10.1287/opre.2021.2242.

Stephen Bates, Anastasios Angelopoulos, Lihua Lei, Jitendra Malik, and Michael Jordan. Distribution-free, risk-controlling prediction sets. *Journal of the ACM (JACM)*, 68(6):1–34, 2021.

Vidmantas Bentkus. On Hoeffding's inequalities. *The Annals of Probability*, 32(2):1650–1673, 2004. doi: 10.1214/009117904000000360.

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(10):281–305, 2012. URL https://www.jmlr.org/papers/v13/bergstra12a.html.

Hedyeh Beyhaghi and Robert Kleinberg. Pandora's problem with nonobligatory inspection. In *Proceedings of the 2019 ACM Conference on Economics and Computation (EC)*, pp. 131–132. ACM, 2019. doi: 10.1145/3328526.3329626.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Aleksander Madry, and Lilian Weng. MLE-bench: Evaluating machine learning agents on machine learning engineering. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://openreview.net/forum?id=6s5uXNWGIh.

Shuchi Chawla, Evangelia Gergatsouli, Yifeng Teng, Christos Tzamos, and Ruimin Zhang. Pandora's box with correlations: Learning and approximation. In *Proceedings of the 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 1214–1225. IEEE, 2020. doi: 10.1109/FOCS46700.2020.00116.

Tobias Domhan, Jost Tobias Springenberg, and Frank Hutter. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 3460–3468, 2015. URL https://www.ijcai.org/Abstract/15/487.

Stefan Falkner, Aaron Klein, and Frank Hutter. BOHB: Robust and efficient hyperparameter optimization at scale. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1437–1446. PMLR, 2018. URL https://proceedings.mlr.press/v80/falkner18a.html.

Thomas Simon Foster, Bassel Al Omari, Tingchen Fu, Thomas Mann, Carl Domond, Lucia Cipolina-Kun, Bhavul Gauri, Muna Aghamelu, Alexander D. Goldie, Eryk Helenowski, Jean-Christophe Gagnon-Audet, Alberto Pepe, Saba Nazir, Daniel Izcovich, Noam Levi, Rishi Hazra, Karen Hambardzumyan, Nicolas Baldwin, Xian Li, Martin Josifoski, Paris Giampouras, Masoud Jalili Sabet, Anya Sims, Hela Momand, Tatiana Shavrina, Despoina Magka, Jason Weston,

Yulin Wang, Anirudh Goyal, João Henriques, Yoram Bachrach, Emily McMilin, and Jakob Nicolaus Foerster. Ai research preference models, 2026. URL https://arxiv.org/abs/2608.13940.

Hu Fu, Jiawei Li, and Daogao Liu. Pandora box problem with nonobligatory inspection: Hardness and approximation scheme. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 789–802. ACM, 2023. doi: 10.1145/3564246.3585229.

Pieter Gijsbers, Marcos L. P. Bueno, Stefan Coors, Erin LeDell, Sébastien Poirier, Janek Thomas, Bernd Bischl, and Joaquin Vanschoren. AMLB: An AutoML benchmark. *Journal of Machine Learning Research*, 25(101):1–65, 2024. URL https://www.jmlr.org/papers/v25/22-0493.html.

Daniel Golovin, Benjamin Solnik, Subhodeep Moitra, Greg Kochanski, John Karro, and D. Sculley. Google Vizier: A service for black-box optimization. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 1487–1495. ACM, 2017. doi: 10.1145/3097983.3098043.

Karen Hambardzumyan, Nicolas Baldwin, Edan Toledo, Rishi Hazra, Michael Kuchnik, Bassel Al Omari, Thomas Simon Foster, Anton Protopopov, Jean-Christophe Gagnon-Audet, Ishita Mediratta, Kelvin Niu, Michael Shvartsman, Alisia Lupidi, Alexis Audran-Reiss, Parth Pathak, Tatiana Shavrina, Despoina Magka, Hela Momand, Derek Dunfield, Nicola Cancedda, Pontus Stenetorp, Carole-Jean Wu, Jakob Nicolaus Foerster, Yoram Bachrach, and Martin Josifoski. AIRA_2: Overcoming bottlenecks in AI research agents. arXiv preprint arXiv:2603.26499, 2026. URL https://arxiv.org/abs/2603.26499.

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963. doi: 10.1080/01621459.1963.10500830.

Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. MLAgentBench: Evaluating language agents on machine learning experimentation. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pp. 20271–20309. PMLR, 2024. URL https://proceedings.mlr.press/v235/huang24y.html.

Zhengyao Jiang, Dominik Schmidt, Dhruv Srikanth, Dixing Xu, Ian Kaplan, Deniss Jacenko, and Yuxiang Wu. AIDE: AI-driven exploration in the space of code. arXiv preprint arXiv:2502.13138, 2025. URL https://arxiv.org/abs/2502.13138.

Kirthevasan Kandasamy, Gautam Dasarathy, Jeff Schneider, and Barnabás Póczos. Multi-fidelity Bayesian optimisation with continuous approximations. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1799–1808. PMLR, 2017. URL https://proceedings.mlr.press/v70/kandasamy17a.html.

Eric Hans Lee, Valerio Perrone, Cedric Archambeau, and Matthias Seeger. Cost-aware Bayesian optimization. arXiv preprint arXiv:2003.10870, 2020. URL https://arxiv.org/abs/2003.10870.

Eric Hans Lee, David Eriksson, Valerio Perrone, and Matthias Seeger. A nonmyopic approach to cost-constrained Bayesian optimization. In *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence (UAI)*, volume 161 of *Proceedings of Machine Learning Research*, pp. 568–577. PMLR, 2021. URL https://proceedings.mlr.press/v161/lee21a.html.

Liam Li, Kevin Jamieson, Afshin Rostamizadeh, Ekaterina Gonina, Jonathan Ben-tzur, Moritz Hardt, Benjamin Recht, and Ameet Talwalkar. A system for massively parallel hyperparameter tuning. In *Proceedings of Machine Learning and Systems (MLSys)*, volume 2, pp. 230–246, 2020. URL https://proceedings.mlsys.org/paper_files/paper/2020/hash/a06f20b349c6cf09a6b171c71b88bbfc-Abstract.html.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18(185):1–52, 2018. URL https://www.jmlr.org/papers/v18/16-558.html.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The AI scientist: Towards fully automated open-ended scientific discovery. arXiv preprint arXiv:2408.06292, 2024. URL https://arxiv.org/abs/2408.06292.

Alisia Lupidi, Bhavul Gauri, Thomas Simon Foster, Bassel Al Omari, Despoina Magka, Alberto Pepe, Alexis Audran-Reiss, Muna Aghamelu, Nicolas Baldwin, Lucia Cipolina-Kun, Jean-Christophe Gagnon-Audet, Chee Hau Leow, Sandra Lefdal, Hossam Mossalam, Abhinav Moudgil, Saba Nazir, Emanuel Tewolde, Isabel Urrego, Jordi Armengol Estape, Amar Budhiraja, Gaurav Chaurasia, Abhishek Charnalia, Derek Dunfield, Karen Hambardzumyan, Daniel Izcovich, Martin Josifoski, Ishita Mediratta, Kelvin Niu, Parth Pathak, Michael Shvartsman, Edan Toledo, Anton Protopopov, Roberta Raileanu, Alexander Miller, Tatiana Shavrina, Jakob Foerster, and Yoram Bachrach. AIRS-Bench: A suite of tasks for frontier AI research science agents. arXiv preprint arXiv:2602.06855, 2026. URL https://arxiv.org/abs/2602.06855.

David Salinas and Nick Erickson. TabRepo: A large scale repository of tabular model evaluations and its AutoML applications. In *Proceedings of the Third International Conference on Automated Machine Learning (AutoML)*, volume 256 of *Proceedings of Machine Learning Research*, pp. 19/1–30. PMLR, 2024. URL https://proceedings.mlr.press/v256/salinas24a.html.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1): 148–175, 2016. doi: 10.1109/JPROC.2015.2494218.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper_files/paper/2012/hash/05311655a15b75fab86956663e1819cd-Abstract.html.

Edan Toledo, Karen Hambardzumyan, Martin Josifoski, Rishi Hazra, Nicolas Baldwin, Alexis Audran-Reiss, Michael Kuchnik, Despoina Magka, Minqi Jiang, Alisia Maria Lupidi, Andrei Lupu, Roberta Raileanu, Kelvin Niu, Tatiana Shavrina, Jean-Christophe Gagnon-Audet, Michael Shvartsman, Shagun Sodhani, Alexander H. Miller, Abhishek Charnalia, Derek Dunfield, Carole-Jean Wu, Pontus Stenetorp, Nicola Cancedda, Jakob Nicolaus Foerster, and Yoram Bachrach. Ai research agents for machine learning: Search, exploration, and generalization in mle-bench, 2025. URL https://arxiv.org/abs/2507.02554.

Martin L. Weitzman. Optimal search for the best alternative. *Econometrica*, 47(3):641–654, 1979. doi: 10.2307/1910412.

Hjalmar Wijk, Tao Roa Lin, Joel Becker, Sami Jawhar, Neev Parikh, Thomas Broadley, Lawrence Chan, Michael Chen, Joshua M Clymer, Jai Dhyani, Elena Ericheva, Katharyn Garcia, Brian Goodrich, Nikola Jurkovic, Megan Kinniment, Aron Lajko, Seraphina Nix, Lucas Jun Koba Sato, William Saunders, Maksym Taran, Ben West, and Elizabeth Barnes. RE-bench: Evaluating frontier AI r&d capabilities of language model agents against human experts. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 66772–66832. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/wijk25a.html.

Qian Xie, Raul Astudillo, Peter I. Frazier, Ziv Scully, and Alexander Terenin. Cost-aware Bayesian optimization via the Pandora's box Gittins index. In *Advances in Neural Information Processing Systems*, volume 37, pp. 115523–115562. Curran Associates, Inc., 2024. doi: 10.52202/079017-3669. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/d14c355d5e88cff437a6303d2d716252-Abstract-Conference.html.

Jingsheng Zheng, Jintian Zhang, Yujie Luo, Yuren Mao, Yunjun Gao, Lun Du, Huajun Chen, and Ningyu Zhang. Can we predict before executing machine learning agents? In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3941–3974. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.182. URL https://aclanthology.org/2026.acl-long.182/.

Yunhong Zhou, Deeparnab Chakrabarty, and Rajan Lukose. Budget constrained bidding in keyword auctions and online knapsack problems. In *Internet and Network Economics (WINE)*, Lecture Notes in Computer Science, pp. 566–576. Springer, 2008. doi: 10.1007/978-3-540-92185-1_63.

Lucas Zimmer, Marius Lindauer, and Frank Hutter. Auto-PyTorch: Multi-fidelity MetaLearning for efficient and robust AutoDL. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(9):3079–3090, 2021. doi: 10.1109/TPAMI.2021.3067763.

A Descendant Conventions on AIRS-Bench

The agent builds each candidate on its current best result, so a node that a selection rule skips never becomes a parent in that rule's run, and the descendants the recording holds for it no longer describe what would have happened. Three conventions resolve this and we report all three.

Under the first, descendants arrive as recorded and keep their recorded scores. Under the second, the whole subtree of a skipped node is removed from the stream. Under the third, which the main text uses, descendants arrive and are re-based, so a descendant whose recorded score exceeded its recorded parent by some amount is credited with the rule's own incumbent plus that amount, clipped to the score range, and the offset is carried down the tree. The first convention rewards skipping, because a rule that skips a node still receives everything built on it. The second penalizes skipping beyond what the agent would actually lose, because in a real run the agent would have continued from whatever it did have. The third keeps the improvement the recording measured while attaching it to the state the rule is actually in.

All replays were run under all three. The direction of every comparison reported in the main text is the same under all three conventions.

B Degeneracy Checks

Every replay is checked before any number is read off it. The checks state properties the replay must satisfy by construction, so each one tests the replay code itself. Table 2 lists them with the counts that passed.

C Additional Results

C.1 TabRepo by task group

Table 1 averages the three TabRepo groups. Table 4 separates them.

C.2 Each part of the rule matters on AIRS-Bench

Each row of Table 5 changes one thing and leaves the rest of CAES untouched, and every arm receives the same predictor fitted leave one task out. Replacing the quota price with a fixed threshold gives up cells, and so does removing inspection. Keeping inspection but spreading it over every admitted candidate, or scattering it at the same rate at random, loses every budget, which supports the placement result of Section 4.

C.3 A dense budget grid

The four budgets of the main text are a coarse grid. Figure 6 sweeps 24 logarithmically spaced budgets on AIRS-Bench, restricted to the families that carry no parameters so that no selection step is needed at any grid point.

Table 2: Degeneracy checks. Each entry counts the instances that passed out of the instances tested, where an instance is one combination of task, budget, method family, and descendant convention as the check requires. Checks whose premise is that the budget is ample use a budget of one million times what running the whole stream costs, because the affordable fraction of Equation 6 divides by predicted costs and a merely generous budget does not drive the price to zero.

Replay	Property	Passed
Main replay	A price family with an ample budget equals run-all	16/16
	Spending never exceeds the budget or the warm start	44016/44016
	A budget equal to the warm start executes only those candidates	3668/3668
	Siblings share a parent and no member parents its own group	277/277
Decision log	Logging decisions leaves the trajectory unchanged	1048/1048
Dense budgets	Repeating a budget point reproduces it exactly	28/28
	A price family with an ample budget equals run-all	12/12
Ablations	Removing inspection reproduces the degenerate density	56/56
	A zero failure probability makes two arms coincide	56/56
	An ample budget makes every added family equal run-all	84/84
	CAES reproduces the main replay to the bit	56/56
Inspections	Inspecting every admitted candidate captures every veto	43/43
	The hindsight ceiling is never below another arm	86/86
	Correct vetoes are a subset of vetoes	43/43
	The equal cost arms stay inside the inspection budget	86/86
	The reference trajectory matches the no-inspection ablation to the bit	56/56
Certificate	Gaps of a budget independent family fall as the budget rises	602/602
	An ample budget gives CAES a gap of zero	14/14
	The loss matrix agrees with the gap crossing the tolerance	198/198
	A test set of one task certifies nothing and falls back to run-all	pass

Table 3: Selectivity on AIRS-Bench. Left, the share of arriving candidates a rule declines. Right, the share of large improvements it declines divided by that overall share, where a large improvement is worth more than 0.02 on the normalized scale. One means both are declined alike and smaller means refusals steer away from the improvements.

	Declined share of arrivals				Ratio on large improvements			
Method	5	10	20	40	5	10	20	40
ForeAgent	0.34	0.45	0.54	0.53	1.49	1.33	1.20	1.33
Preference model	0.43	0.42	0.56	0.54	1.16	1.19	1.15	1.19
Random skip	0.42	0.34	0.26	0.26	1.05	0.84	1.70	0.76
CAES	**0.79**	**0.80**	**0.79**	**0.59**	**0.16**	**0.45**	**0.23**	**0.28**

D THE DECISION RULE AS ONE PROCEDURE

Algorithm 1 collects the rule of Section 2 for one task in the order in which its steps run. The price is computed once per arriving candidate, before the candidate is decided, and the predictor is updated only after a full execution returns a score.

Predicted costs are floored at one second. On TabRepo and LCBench no execution fails, so $p_i = 0$ and $q = 1$, an inspection is a prefix of the full run with $f = 1/3$ and $r_i = (1 - f)k_i$, the candidate cap is $T = 150$, and the paced schedule selects λ_0 from ten values spaced evenly on a log scale between 10^{-9} and 10^{-3} and η from $\{1, 4\}$. On AIRS-Bench an inspection runs on subsampled data, so $r_i = k_i$, and its predicted cost is k_i times the median ratio of inspection to full cost seen so far on the task, one fifth before any inspection has run. The failure probability p_i is the failure rate of the operator that produced the candidate, pooled over the other tasks and updated with the candidates already executed on this task, while q and s_i are fitted on the recordings of the other tasks. The quota schedule there uses the candidate cap $T = 150$ and a wall clock cap of twelve hours per task, and averages the wall clock of the last ten candidates to estimate m_i, falling back to the elapsed wall clock divided by the number of arrivals before any candidate has finished. It keeps m_i at least one,

Table 4: Gap to an unbudgeted run on each TabRepo group at the four budgets, lower is better.

	Binary				Multiclass				Regression			
Method	5	10	20	40	5	10	20	40	5	10	20	40
Run-all	.199	.128	.084	.046	.156	.099	.058	.035	.048	.034	.021	.011
Oracle	.091	.025	.009	.003	.075	.013	.001	.000	.025	.007	.001	.000
Random skip	.185	.125	.073	.035	.148	.089	.055	.028	.050	.032	.021	.010
Small-scale first	.186	.115	.066	.028	.140	.075	.039	.010	.049	.031	.017	.006
Successive halving	.190	.119	.072	.029	.147	.085	.041	.012	.049	.033	.017	.009
Median stopping	.192	.120	.073	.030	.148	.085	.044	.016	.050	.036	.019	.009
Curve extrapolation	.191	.122	.080	.042	.142	.079	.040	.013	.049	.033	.019	.009
ForeAgent	.187	.124	.074	.066	.138	.079	.049	.043	.052	.038	.022	.021
Preference model	.212	.147	.096	.054	.148	.085	.048	.019	.063	.048	.034	.020
EI per cost	.190	.114	.074	.047	.146	.086	.048	.029	.052	.040	.018	.008
CArBO	.188	.117	.076	.044	.150	.087	.045	.029	.054	.039	.019	.008
CAES, fixed price	.181	.114	.070	.043	.142	.084	.040	.023	.043	.029	.018	.007
CAES	.170	.112	.064	.035	.143	.072	.039	.019	.045	.027	.015	.009

Table 5: One change at a time on AIRS-Bench, counted in cells over the four budgets, CAES against the altered rule.

Change	point	bootstrap
Quota price replaced by a fixed threshold	3-1	0-1
Inspection removed	2-2	0-0
Inspection given to every admitted candidate	4-0	1-0
Inspection scattered at random at the same rate	4-0	0-0

takes the quantile as a weighted average of the two nearest ordered values, and holds the price at zero until five candidates have arrived, so that the quantile has a sample to read.

E The Rule Takes Milliseconds per Candidate

Deciding one candidate takes one predictor query, one evaluation of Equation 2, and one evaluation of Equation 4 over a fixed number of quadrature nodes, a cost that does not grow with how many candidates have already arrived and is measured in milliseconds against experiments that take minutes to hours. The quota schedule additionally reads one quantile of the value densities seen so far, which costs $O(\log i)$ per candidate with a running order statistic. No part of the rule trains a model of its own.

F Reproducibility Statement

Section 3.1 states the benchmarks, the recording protocol, and the descendant convention. Section 3.2 states the budget definition, the metric, the leave one task out selection of every parameter, and the pairing and bootstrap used for every comparison. Section 3.3 states the predictor each rule receives. Appendix B lists the degeneracy checks each replay must pass before any number is reported. The recordings, the replay code, and the analysis scripts will be released.

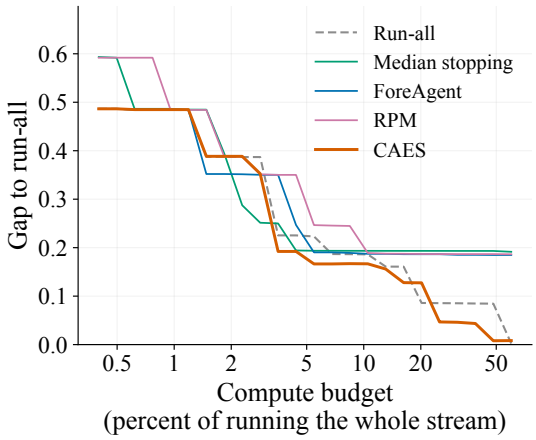

Figure 6: Gap against budget on AIRS-Bench over 24 logarithmically spaced budgets, averaged over the 14 tasks, for the families that carry no parameters.

Algorithm 1 CAES on one task with budget B

Complete the warm start candidates, set C to what they cost and b to their best score
for each arriving candidate i **do**
    Query the predictor for $\mu_i, \sigma_i, p_i, s_i, k_i$ and set λ_i by Equation 5 or Equation 6
    if $\lambda_i = 0$ **then**
        $a \leftarrow$ complete
    else
        Compute v_i^{run} and v_i^{inspect} by Equations 2 and 4
        if $v_i^{\text{inspect}} > \max(v_i^{\text{run}}, 0)$ **then**
            $a \leftarrow$ inspect
        else if $v_i^{\text{run}} \geq 0$ **then**
            $a \leftarrow$ complete
        else
            $a \leftarrow$ skip
        end if
    end if
    if $a =$ inspect **then**
        if $C + fk_i > B$ **then stop**
        end if
        Pay fk_i, read z_i, and set $a \leftarrow$ complete if $q\text{EI}(b, \mu_i(z_i), \sigma_i(z_i)) \geq \lambda_i r_i$, else skip
        The cost still due is r_i
    else
        The cost still due is k_i
    end if
    if $a =$ complete **then**
        if $C +$ cost still due $> B$ **then stop**
        end if
        Pay it, observe y_i, set $b \leftarrow \max(b, y_i)$, and update the predictor
    end if
end for
return b

G USAGE OF AI ASSISTANTS

We used AI assistants to polish the writing of this paper and to search for related work. Every suggested edit and every retrieved reference was checked by us before inclusion, and we take full responsibility for the content of the paper.