ADAPTIVE TRANSFER FOR COST-AWARE BAYESIAN OPTIMIZATION OF LLM AGENT CONFIGURATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM agents can now autonomously perform long-horizon development tasks. An agent's configuration, including its reasoning effort, tool access, and resource allocation, affects its performance, runtime, and monetary cost. An optimal configuration needs to balance these outcomes, as greater resource use does not necessarily yield higher utility. A full grid search on each new task is not practical because every configuration requires a costly, time-consuming run. Bayesian optimization can improve sample efficiency in search, and transferring knowledge from earlier tasks can further reduce search cost. The challenge is to identify what transfers when outcomes change across tasks and few target evaluations are affordable. In this work, we propose adaptive-level transfer (ALT), a cost-aware Bayesian optimization method that uses target evidence to adapt what is transferred from earlier tasks. Through systematic evaluation of every configuration of one agent across several tasks, recording agent action trajectories and outcomes, we obtain complete landscapes for exact replay of configuration search. Our analysis shows that configuration effects on several agent behavior features are relatively stable across tasks, while their effects on utility shift. ALT transfers configuration-to-behavior patterns from source trajectories and learns their utility contributions from target runs. It combines this behavior-level hypothesis with an objective-level hypothesis that transfers source utility surfaces. The target evidence decides how much each contributes to the prediction. Its acquisition weighs expected improvement against predicted cost, with the cost discount learned from target evidence on whether resource-intensive behavior raises utility. In cross-task search under monetary budgets, ALT reduces mean regret area by 4–12% relative to our main transfer baseline, ScaML-GP with expected improvement across all four evaluated utilities, with the largest reductions on the performance- and time-focused utilities. When sources include other repetitions of the target task, ALT outperforms its behavior-only ablation.

1 INTRODUCTION

LLM agents can complete development tasks autonomously. They iterate over code editing, tool calls, and evaluation of intermediate results during development (Yang et al., 2024; Jiang et al., 2025; Chan et al., 2025). Before an agent runs a task, a deployer needs to choose its configuration, including the model, reasoning effort, resources, skills, and tools. The configuration affects how the agent behaves on the task and what the run consumes (e.g. tokens and time) (Kapoor et al., 2025; Wang et al., 2025). For example, the highest reasoning effort may not produce the best outcome on every task. The return on additional compute depends on the problem (Snell et al., 2025), while excessive reliance on internal reasoning has been associated with lower success on agentic tasks (Cuadron et al., 2025). When performance, runtime, and monetary cost are combined into one utility, the configurations form a complicated utility landscape that the deployer has to investigate on each task. This work focuses on configuration search for a single agent under a utility that combines performance with runtime and cost.

Each candidate configuration requires a complete agent run, which can take hours and cost dollars, limiting the number of evaluations affordable on each task. Bayesian optimization (BO) can improve search efficiency (Snoek et al., 2012; Garnett, 2023), and transfer BO can further reduce search cost by reusing completed runs (Feurer et al., 2022; Wistuba & Grabocka, 2021; Tighineanu et al., 2024). We call tasks with completed runs the source tasks and the task being configured the target task. Common transfer BO methods use objective values observed on source tasks to guide search on the target.

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.

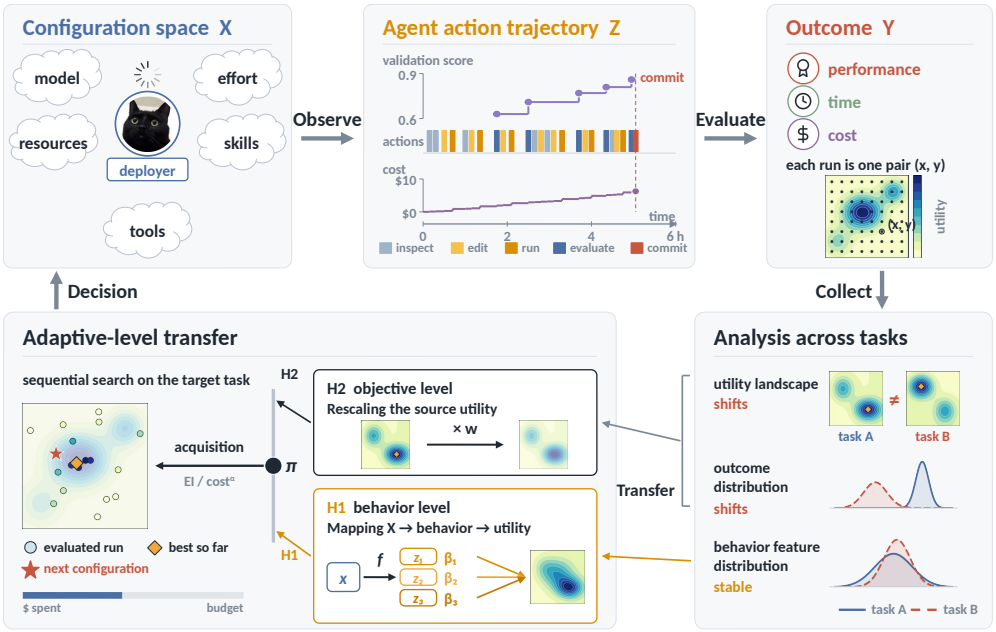

Figure 1: Overview of this work. In the top row a deployer fixes an agent configuration x, the run produces an action trajectory Z, and its outcome Y combines held-out performance with recorded runtime and cost (Section 3). The lower-right panel illustrates similar configuration effects on behavior features alongside shifts in utility effects (Section 4). The lower-left panel presents ALT, which rescales source utility surfaces with weights w under H_2, learns the utility contributions β of transferred behavior patterns under H_1, weights the two hypotheses by target evidence π, and selects the next run by expected improvement discounted by cost with a learned exponent α (Section 5). Landscapes, curves, and the search canvas are for illustrative purposes.

However, agent runs record action trajectories (time-stamped sequences of tool calls and development reports) in addition to outcomes, and a configuration change may induce similar behavior on two tasks while its contribution to utility differs. For example, running more experiments during development may improve performance on one task but mainly add runtime and cost on another. This raises the question of whether source runs are most useful through their utility surfaces or through the behavior that configuration choices induce, and how to decide this from a few target runs.

We propose adaptive-level transfer (ALT), a cost-aware transfer BO method, shown in the context of our overall workflow in Fig. 1. The top row shows one recorded run, where a run evaluates one configuration on one task and a landscape records one run for every configuration on that task. Figure 1 also gives an overview of our three contributions. First, we evaluate the full configuration grid of a single LLM agent on four development tasks and record every run in full, yielding complete landscapes for replaying configuration search (top row, Section 3). Second, we compare configuration effects on agent behavior with their effects on utility across tasks (lower-right panel, Section 4). Third, ALT combines behavior-level and objective-level transfer using target evidence and selects the next run with an adaptive cost discount (lower-left panel, Section 5). We consider four tasks to collect six landscapes, with three replications of one task and one replication for the remaining tasks. Each run records its action trajectory, held-out performance, runtime, and cost. The shared factors make configuration effects comparable across tasks, while repeated landscapes quantify variation across executions of the same configuration on the same task. Because every configuration has been evaluated, sequential search can be replayed exactly and scored against the best recorded configuration. The lower-right panel summarizes the exploratory analysis in Section 4. Behavior features capture measurable aspects of the action trajectories, and several of them respond to configuration factors in similar ways across tasks, whereas the effects on utility shift and can reverse direction even at fixed utility weights. Repeated landscapes of the same task show similar average utility effects but less consistent configuration rankings. These findings motivate transferring behavior patterns with utility contributions learned on the target, and suggest that the useful level of transfer depends on the source.

The lower-left panel presents ALT, detailed in Section 5. The behavior hypothesis H_1 learns a linear configuration-to-behavior map from source trajectories and an affine behavior-to-utility map from target outcomes, and the objective hypothesis H_2 transfers source utility posteriors with positive weights using ScaML-GP (Tighineanu et al., 2024). Both hypotheses include a Gaussian process for target residuals, and after each target run ALT compares their marginal likelihoods at the fitted parameters and mixes their predictions accordingly. Our ADAPT acquisition (building on Snoek et al., 2012; Lee et al., 2020) uses cost-discounted expected improvement with an exponent learned from target runs.

We evaluate configuration search across tasks and across repetitions of a task under monetary and evaluation-count budgets (Section 6), comparing ALT with transfer methods, target-only search, and ablations of its transfer and acquisition components. In cross-task search under monetary budgets, ALT achieves lower mean regret area than our main transfer baseline, ScaML-GP with expected improvement, across all evaluated utility settings. It also improves over its behavior-only ablation when sources include other repetitions of the target task. We release the landscapes, full data with recorded action trajectories, the experiment launcher, and the code for ALT, the baselines, and the ablations.

2 Related work

Prior work on LLM systems optimizes prompts (Chen et al., 2024; Opsahl-Ong et al., 2024), tunes inference parameters under cost constraints (Wang et al., 2023), and searches over workflow and agent designs (Saad-Falcon et al., 2025; He et al., 2025; Hu et al., 2025; Shang et al., 2025; Zhang et al., 2025). AgentOpt uses bandit and Bayesian search to assign models to the stages of an agent pipeline (Hua et al., 2026). Taparia et al. (2026) use reinforcement learning to select agent configurations for individual queries, while Barker et al. (2025) use multi-objective BO to tune LLM and retrieval-augmented generation systems. We study sequential configuration search for a single agent on a target task, where completed source runs provide information to guide the search.

Transfer BO reuses source evaluations through several mechanisms (Bai et al., 2023), including multitask Gaussian processes (GPs) (Swersky et al., 2013), ranking ensembles (Feurer et al., 2022), learned kernels (Wistuba & Grabocka, 2021), and pretrained priors (Wang et al., 2024). ScaML-GP fits source weights on the target and combines source posterior means and covariances with a target residual GP (Tighineanu et al., 2024). We adopt this structure for the objective hypothesis in ALT, allowing source utility surfaces to be rescaled and corrected using target observations. Bayesian model averaging accounts for uncertainty in the choice of surrogate during search (Malkomes & Garnett, 2018). ALT weights its two transfer hypotheses using their marginal likelihoods at fitted parameters. Cost-aware acquisition rules account for evaluation cost through expected improvement per unit cost (Snoek et al., 2012), budget-dependent cooling (Lee et al., 2020), budgeted lookahead (Astudillo et al., 2021), and the Pandora's-box Gittins index (Xie et al., 2024).

Grey-box BO uses information generated within evaluations to guide search (Astudillo & Frazier, 2021). Trajectory kernels incorporate information from policy rollouts (Wilson et al., 2014), while other methods use partial learning curves (Swersky et al., 2014). For LLM systems, GEPA reflects on execution traces to propose prompts (Agrawal et al., 2026), while VeRO records structured traces of agent harnesses for optimization by coding agents (Ursekar et al., 2026). ALT learns configuration-to-behavior patterns from source trajectories and their utility contributions from target outcomes. It adaptively combines this behavior-level transfer with objective-level transfer.

Recorded evaluations support controlled comparisons of search methods. NAS-Bench-101, HPO-B, and YAHPO Gym provide tabular or surrogate benchmarks (Ying et al., 2019; Pineda Arango et al., 2021; Pfisterer et al., 2022). Studies of hyperparameter importance examine how the influence of configuration choices varies across datasets (Hutter et al., 2014; van Rijn & Hutter, 2018). Agent studies examine interface design (Yang et al., 2024), performance–cost tradeoffs (Wang et al., 2025), and cost-controlled evaluation (Kapoor et al., 2025). Our full factorial agent landscapes connect trajectories with outcomes across tasks and repetitions, supporting behavior analysis and exact replay.

3 Experiment and dataset

We select tasks with numerical development feedback and separate evaluation of the final artifact. Two screening rounds evaluated nine configurations per task on eight and six candidates, respectively,

prioritizing performance variation, manageable runtime and cost, and task diversity. The selected tasks are Spooky Author Identification (text classification) and Ventilator Pressure Prediction (sequence regression) from MLE-bench (Chan et al., 2025), and Adversarial Splay (combinatorial optimization) and Scaling Law (language model training) from AutoLab (Xu et al., 2026). We collect three landscapes for Spooky to examine repeated-execution variation and one for each remaining task. We use a *full factorial design* over five factors, namely model, reasoning effort, resource budget, workflow skills, and tool interface. Their three levels (Table 1) give $3^5 = 243$ configurations $x \in \mathcal{X}$ per landscape. The shared grid supports cross-task transfer, while Spooky repetitions provide same-task sources that can be used alone or combined with other-task sources in the mixed setting (Section 6).

Table 1: Configuration factors and levels. Parentheses give the number of exposed tools. Resource caps limit the number of counted workspace tool calls within the common six-hour walltime. All tool interfaces retain shell access.

Factor	Levels	Description
Model M	M1: Sonnet 5, M2: Opus 5, M3: Fable 5	Requested language model. Served versions are recorded.
Effort E	E1: medium, E2: high, E3: max	Reasoning-effort setting passed through the SDK.
Resources R	R1: uncapped, R2: 86 calls, R3: 45 calls	Caps from screening tool-call usage at the 75th and 50th percentiles.
Skills S	S0: none, S1: principles, S2: procedures	General principles or explicit procedures with required records.
Tools T	T0: generic (5), T1: structured (8), T2: managed (13)	T1 adds file operations. T2 adds trial and budget management.

Each run starts in a fresh workspace with a dedicated NVIDIA A100 GPU (80 GB), 16 CPU cores, 128 GB of memory, and a six-hour walltime. We deliberately provide generous computing resources for sustained experimentation. The resource factor varies only the tool-call budget within this allocation. The Claude Agent software development kit (SDK) runs the agent with the configured tools, and a shared base prompt requires development reports, artifact validation, and stopping by commit or abstain. Provider and infrastructure failures are retried. Runs are not retried based on agent outcomes.

The outcome $y = (y_p, y_t, y_c)$ records performance, runtime, and cost. A task-specific grader evaluates the committed artifact or a valid artifact preserved when execution stops. Held-out labels and private grading data are unavailable during the run. For score m and nonzero reference m_0, performance is $y_p = \max\{-1, d(m - m_0)/|m_0|\}$, where $d = 1$ for maximization and $d = -1$ for minimization. Thus, $y_p = 0$ matches the reference, and positive values indicate improvement. References are the MLE-bench leaderboard medians, the random baseline for Adversarial Splay, and the median test perplexity across valid runs in the completed Scaling Law landscape. Invalid or missing artifacts receive $y_p = -1$. Runtime y_t is elapsed seconds from workspace setup to the end of the run. Cost y_c is the API-equivalent price of recorded model usage under a fixed rate card, excluding hardware charges. Complete usage records use the higher cache-write rate when cache lifetime is unspecified. The 21 records truncated by walltime retain flagged lower-bound costs.

For deployer-chosen weights $a_t, a_c \geq 0$, fixed before search, the utility of a run of x is

$$u(x) = y_p - a_t \log_2 y_t - a_c \log_2 y_c, \tag{1}$$

with runtime in seconds and cost in dollars. Holding the other outcomes fixed, doubling runtime or cost lowers utility by a_t or a_c, respectively. We evaluate four settings of (a_t, a_c), named balanced $(0.02, 0.02)$, performance-focused $(0.01, 0.01)$, time-focused $(0.05, 0.01)$, and cost-focused $(0.01, 0.05)$. At the balanced weights each resource term spans at most 0.16 utility units across the landscapes, a fraction of the performance spread, and the time and cost tracks raise one term to a spread comparable to performance (Appendix D.1).

The action trajectory z of a run is the time-stamped sequence of its tool calls and development reports. We label action components using tool names and command text. For example, a shell call that edits a file and launches training contributes an editing component and a training component. Seven categories occur, namely inspection, editing, execution or training, development evaluation, environment setup, checkpointing, and commit. Agent-reported development metrics are timestamped and stored separately from final evaluated scores. The six landscapes contain 1,458 runs with 4,822.5 hours of

recorded runtime and approximately $4,602 in API-equivalent cost. The corpus contains 62,329 tool calls, 145,753 action components, and 5,904 development reports. Two runs produced invalid artifacts.

Configuration search is replayed on a landscape. A method that selects a configuration receives its recorded outcome and pays its recorded cost, observes target outcomes only for the configurations it has selected, and is scored by regret against the best recorded configuration of the target landscape. Screening results, prompts, tool schemas, skill texts, execution details, action-labeling rules, grading references, accounting conventions, and the collection audit appear in Appendix A.

4 Exploratory Analysis of Task Landscapes

We examine outcome distributions and the consistency of configuration effects across tasks, with balanced utility using $a_t = a_c = 0.02$ in Eq. (1) on every task. Figure 2(a–d) shows that the same 243 configurations produce distinct landscapes. Scaling Law runtimes concentrate toward the six-hour limit, while the three Spooky repetitions have closely overlapping distributions. ALT allows source utility surfaces to be rescaled through source weights (Section 5). The following correlations assess whether configuration effects and rankings remain aligned across tasks.

We measure consistency at two levels for utility and behavior features. *Factor-effect correlation* is Pearson's r between matched 15-entry vectors of factor-level effects, where each entry is the mean over the 81 configurations at that level minus the landscape mean. *Configuration-rank correlation* is Spearman's ρ between the 243 configuration values, matching identical configurations and assigning average ranks to ties. For task pairs involving Spooky, we average the three correlations from its repetitions before summarizing the six pairs of the four tasks with equal weight.

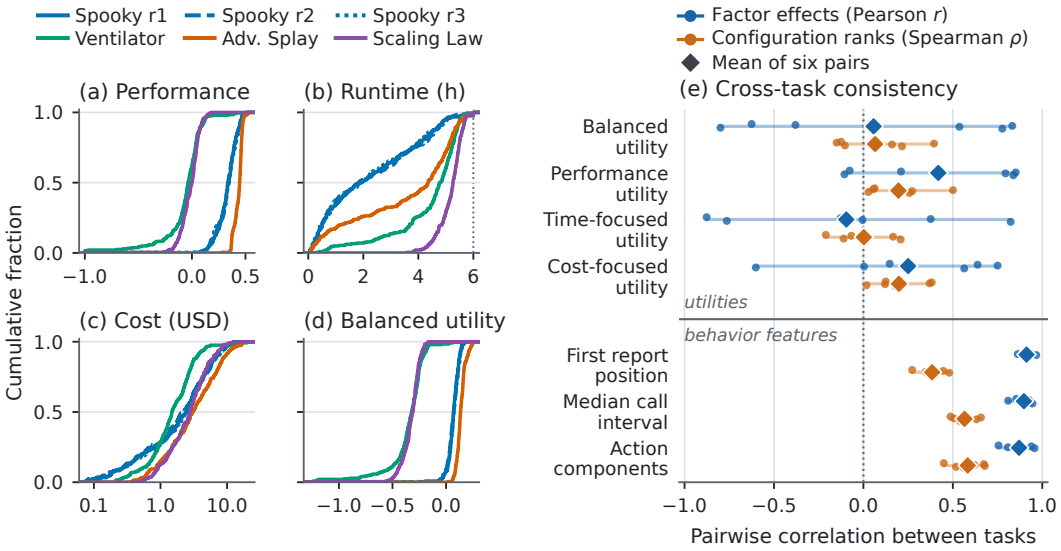

Figure 2: Task landscapes. (a–d) Empirical cumulative distributions over the 243 configurations of each landscape. Spooky repetitions share a color and differ in line style, cost uses a logarithmic axis, and the dotted line marks the six-hour limit. (e) Factor-effect correlations (Pearson's r, blue) and configuration-rank correlations (Spearman's ρ, orange) for six task pairs, with their range and mean. Call intervals and action-component counts use $\log(1 + \cdot)$ before computing factor effects.

Figure 2(e) compares the four utility tracks, each using fixed weights across tasks (Section 3), with three of the most stable behavior features: first development-report position, median interval between tool calls, and nonterminal action-component count. Report position is elapsed time to the first report divided by active duration, which ends at the last action (Appendix B defines all seventeen features). For both correlation measures, values closer to 1 indicate stronger agreement across tasks. Mean factor-effect correlations are 0.91, 0.90, and 0.87 for these behavior features, versus 0.06, 0.42, −0.10, and 0.25 for the balanced, performance-, time-, and cost-focused utilities, respectively. Mean configuration-rank correlations are 0.38–0.58 for these features and 0.00–0.20 for the utilities. These features therefore show greater cross-task consistency than the utilities in both

factor effects and configuration rankings. ALT selects its transfer features using only the available sources (Section 5). The outcome components also have aligned factor effects, but differences in their magnitudes can reverse utility effects under fixed weights (Appendix B). Spooky repetitions show more consistent balanced-utility factor effects ($r = 0.87$–0.94) than configuration rankings ($\rho = 0.39$–0.51). Together, these observations motivate learning behavior-to-utility contributions on the target while retaining objective-level transfer in ALT.

5 ADAPTIVE-LEVEL TRANSFER (ALT)

As defined in Section 3, a configuration $x \in \mathcal{X}$ has five categorical factors with three levels each, so $|\mathcal{X}| = 243$, and $\phi(x) \in \{0,1\}^{15}$ denotes its one-hot encoding. A run of x on a task produces a trajectory z and an outcome y with utility $u(x)$ of Eq. (1), whose weights a_t, a_c are fixed before the search. We denote the task to be configured as the *target task*. Its outcome or utility is unknown until a configuration is run, and the deployer allows a total cost C for the search. In each round one configuration is chosen and run, its outcome is observed, and all runs observed so far inform the next choice. The search continues while the recorded cost of the target runs stays within C and recommends the best configuration observed within the budget, so it is a sequential Bayesian optimization over $\mathcal{X}$ under the predefined budget C.

The S sources are complete landscapes collected before the search, one recorded run per configuration, on other tasks or on other repetitions of the target task, and the task of source s is its *source task*. Its data are $\mathcal{D}^{(s)} = \{(x, z_s(x), y_s(x))\}_{x\in\mathcal{X}}$ for $s = 1, \ldots, S$, and $u_s(x)$ is the source utility under the same weights. After n target runs the target data are $\mathcal{D}_n = \{(x_i, u_i, c_i)\}_{i\le n}$, where u_i is the utility of run i and c_i its cost, kept separately because the acquisition and the budget use it, and X_n is the set of configurations run so far. What ALT transfers, a map from configuration to behavior and a model of each source utility, is fitted on $\mathcal{D}^{(1)}, \ldots, \mathcal{D}^{(S)}$ once before the search, and everything else is fitted on $\mathcal{D}_n$ as it grows. ALT therefore predicts behavior features from configurations using a function learned from source trajectories, without requiring target trajectories. Every method starts from the same nine initial runs, a design in which every level of every factor appears three times, and then chooses one run per round.

5.1 TWO LEVELS OF TRANSFER

As shown in Section 4, several behavior features have similar configuration effects across tasks, while effects on the composite utility shift. Thus we consider two ways of using source landscape to infer hypotheses about the target utility. Under the behavior hypothesis H_1, the target reuses the factor effects on behavior, a map f_{src} from configuration to behavior fitted on the sources, and learns what that behavior is worth. This holds when behavior carries over but its contribution to utility does not. Under the objective hypothesis H_2, the target reuses the source utility surface itself, through the posterior of a Gaussian process fitted on each source. This holds when the target values configurations as the sources did. Which hypothesis is more useful for a given target is not known in advance. ALT fits both and compares them by their marginal likelihoods on the target runs at the fitted parameters. After every run, the resulting hypothesis weights mix the two predictions, and the next run maximizes expected improvement discounted by predicted cost, with an exponent that is itself learned from the target runs. The relative contributions of the two levels of transfer are therefore adapted to the target observations, which gives the method its name. Both hypotheses are Gaussian process (GP) regressions of the standardized target utility $\widetilde{u}$ (Section 5.2),

$$\widetilde{u}(x) = \begin{cases} h(f_{\mathrm{src}}(x)) + r_{\mathrm{tgt}}(x) + \varepsilon & \text{under } \mathrm{H}_1, \\ \widetilde{g}(x) + r_{\mathrm{tgt}}(x) + \varepsilon & \text{under } \mathrm{H}_2, \end{cases} \qquad r_{\mathrm{tgt}} \sim \mathcal{GP}(0, k_X), \qquad \varepsilon \sim \mathcal{N}(0, \sigma^2), \tag{2}$$

where f_{src} predicts behavior features from a configuration, h maps behavior to utility, $\widetilde{g}$ is the standardized weighted sum of the source utility posteriors, and r_{tgt} is a residual over configurations. The residual kernel is a weighted Hamming kernel, $k_X(x, x') = s^2 \exp\big(-\lambda^\top \delta(x, x')\big)$, where $\delta(x, x') \in \{0,1\}^5$ marks the factors on which the two configurations differ.

Behavior hypothesis. The candidates for f_{src} are seventeen trajectory features extracted from the tool calls and development reports of a run. Since the target learns h from its own runs, what

must transfer is the map from configuration to feature, and we use agreement across sources to select features before target search. For each feature, we fit a ridge regression on $\phi(x)$ with an intercept separately on each source, evaluate the fitted regressions on all 243 configurations, and compute Pearson correlations between their prediction vectors across source pairs. Each feature is scored by its minimum pairwise correlation, and we retain the three features with the highest scores. When there is one source, the two regressions are fitted on a fixed random split of its landscape into halves and scored by the correlation of their predictions on the full grid. Whether a stable feature matters for the target utility, and with which sign, is left to h. The map f_{src} is the ridge regression of these three features, standardized, on $\phi(x)$ over all source runs. The map h is affine, $h(f_{\mathrm{src}}(x)) = b + \beta^\top f_{\mathrm{src}}(x)$, with the prior $\beta \sim \mathcal{N}(0, \tau^2 I_3)$. The prior scale is not fitted but set to $\tau^2 = \left(\frac{1}{243}\sum_{x\in\mathcal{X}} \|f_{\mathrm{src}}(x)\|^2\right)^{-1}$, which gives $\beta^\top f_{\mathrm{src}}(x)$ unit prior variance on average over $\mathcal{X}$, the scale of the standardized utility. Integrating β out makes H_1 a GP with kernel

$$k_1(x, x') = k_X(x, x') + \tau^2 f_{\mathrm{src}}(x)^\top f_{\mathrm{src}}(x'), \tag{3}$$

so two configurations have correlated utilities under H_1 when the sources predict that they behave alike. The intercept b takes its maximum-likelihood value in closed form, and the target runs give β a closed-form Gaussian posterior $\mathcal{N}(m_\beta, V_\beta)$, which learns the sign and size of each feature's contribution.

Objective hypothesis. For each source s we fit a GP with kernel k_X to u_s over its landscape and keep its posterior $g^{(s)} \sim \mathcal{GP}(\mu^{(s)}, k^{(s)})$ in utility units, whose mean is retained as a fixed function of the configuration during target search. Following ScaML-GP (Tighineanu et al., 2024), which treats the source posteriors as independent, the transferred process is their positively weighted sum in the standardized units described in Section 5.2, $\widetilde{g}(x) = \left(\sum_s w_s g^{(s)}(x) - \nu_n\right)/\rho_n$, and adding the target residual gives H_2 the kernel

$$k_2(x, x') = k_X(x, x') + \rho_n^{-2} \sum_{s=1}^{S} w_s^2 \, k^{(s)}(x, x'). \tag{4}$$

The source weights $w_s > 0$, one scalar per source, carry a $\mathrm{Gamma}(1, 1)$ prior and are fitted on the target runs together with (s^2, λ, σ^2). They are distinct from the utility weights a_t, a_c and from the hypothesis weight π_n below.

5.2 Fitting, hypothesis weights, and acquisition

Every round, utilities are standardized by the mean ν_n and standard deviation ρ_n of all source utilities pooled with the target utilities observed so far, which defines $\widetilde{u}$, and predictions are mapped back to utility units. Each source posterior $g^{(s)}$ is fitted in the standardized units of its own source and converted to utility units once before the search. Under each hypothesis, (s^2, λ, σ^2), and under H_2 also $w_1, \ldots, w_S$, are refitted after every run by maximizing the marginal likelihood, with the stated source-weight prior under H_2. Let $\ell_{j,n}$ be the log marginal likelihood of the n observed utilities under hypothesis j at its fitted parameters. With equal prior odds, the evidence weight of the behavior hypothesis is $\pi_n = 1/(1 + \exp(\ell_{2,n} - \ell_{1,n}))$. Each hypothesis gives a predictive mean and variance $(\mu_{j,n}(x), v_{j,n}(x))$ at every configuration, and the mixture is summarized by its moments,

$$\mu_n = \pi_n \mu_{1,n} + (1 - \pi_n)\mu_{2,n}, \qquad v_n = \pi_n v_{1,n} + (1 - \pi_n) v_{2,n} + \pi_n(1 - \pi_n)(\mu_{1,n} - \mu_{2,n})^2, \tag{5}$$

where the last term is the disagreement between the two hypotheses.

A separate target-only GP with kernel k_X on $\log_2 c$ predicts the cost of every configuration, and $\hat{c}_n(x)$ denotes its predicted cost in dollars, the mean of the implied log-normal distribution. We compute expected improvement under a Gaussian matching the mixture's moments, $\mathrm{EI}_n(x) = \mathbb{E}[(U - u_n^+)_+]$ for $U \sim \mathcal{N}(\mu_n(x), v_n(x))$ and $u_n^+ = \max_{i\le n} u_i$ the best observed utility, and the next run is

$$x_{n+1} = \arg\max_{x\in\mathcal{X}\setminus X_n} \frac{\mathrm{EI}_n(x)}{\hat{c}_n(x)^\alpha}, \tag{6}$$

where $\alpha = 0$ is plain expected improvement, $\alpha = 1$ is expected improvement per dollar, and ALT uses the adaptive exponent $\alpha = \alpha_n$ below. Cost thus enters in two roles. The utility weights price it as part of what the deployer wants, and the exponent prices it as search budget, because a cheap run

leaves more runs before the budget is reached. The discount depends on whether behavior associated with higher source cost contributes positively to target utility, which H_1 estimates. Let $\varpi \in \mathbb{R}^3$ be the leading principal direction of f_{src} over $\mathcal{X}$, oriented so that behavior along it increases with the factor effects of the source cost. Under H_1 the utility contribution along this direction is $\varpi^\top \beta$, and the exponent is its posterior probability of being negative, $\alpha_n = P(\varpi^\top \beta < 0 \mid \mathcal{D}_n, \mathrm{H}_1) = \Phi\big(-\varpi^\top m_\beta / \sqrt{\varpi^\top V_\beta \varpi}\big)$ with Φ the standard normal distribution function, computed under H_1 regardless of the mixture weight π_n. Evidence of a negative contribution moves the acquisition toward expected improvement per dollar, and evidence of a positive contribution toward plain expected improvement. This adaptive exponent defines the ADAPT acquisition used by ALT.

5.3 Settings and ablations

The number of features, the ridge penalty, the rule for τ^2, the optimizer starts, and the refit schedule were fixed before the assessment seeds were run and are shared by every method that uses the corresponding component (Appendix C). The comparisons in Section 6 examine the transfer hypotheses, their weighting, and the acquisition rule. Without H_2, the behavior hypothesis alone (BEH) predicts with k_1 and keeps the ADAPT acquisition. The objective-only baseline uses H_2 with expected improvement, which is ScaML-GP, since removing H_1 also removes the coefficient posterior that ADAPT requires. The equal mixture fixes $\pi_n = \frac{1}{2}$ and keeps ADAPT. The fixed exponents $\alpha = 0$ and $\alpha = 1$ and the budget-annealed exponent of cost cooling (Lee et al., 2020) replace ADAPT and keep the mixture.

6 Results

We replay search in 57 scenarios, each a target landscape with source landscapes drawn from other tasks (other-task setting, one to three sources), from repetitions of the target task (same-task setting, one or two), or from both (mixed setting, one repetition plus the three other tasks), on the four utility tracks of Section 3 (Appendix D.1 lists the scenarios). The baselines are objective-level transfer with ScaML-GP (Tighineanu et al., 2024) and RGPE (Feurer et al., 2022), a GP on target data only, and random search, each GP method with expected improvement (EI) and with EI per dollar (/cost), and the ablations of Section 5.3 complete the set. Each search is scored at a dollar budget, the median cost of 60 random evaluations of the target, of which the nine initial runs use about 15 percent on average, and at a count budget of 60 evaluations. The score is the regret area, where the regret after a run is the gap between the best utility in the landscape and the best utility observed so far, in units of the utility range, and the area is its mean over the dollars spent, or the evaluations made, after the initial design. Scores are averaged over the scenarios of a setting within each task and then over tasks with equal weight, for 100 seeds that fix a balanced initial design of nine runs shared by all methods. We report the reduction of the regret area by ALT in percent of each method's area with 95 percent intervals from a paired bootstrap over seeds (Appendix D.1).

Table 2 starts with the other-task setting, the case of task shift that motivates the method. ALT reduces the regret area of ScaML-EI by 4 to 12 percent and that of the three EI-per-dollar baselines by 6 to 15 percent on every track, gains 6 to 9 percent on the performance and cost tracks against the target-only GP and RGPE-EI with the remaining intervals including zero, reduces the area of random search by 21 to 39 percent, and is never significantly worse than a baseline. The evaluated objective-level baselines do not consistently improve over the target-only GP under task shift, whereas the behavior hypothesis alone (BEH) has less area than ALT on every track, which indicates that the gains under task shift come from behavior-level transfer.

The mixed and same-task settings reverse the roles, because a repetition of the target task among the sources makes the source utility surface the right thing to transfer. ALT reduces the area of the target-only GP by 26 to 85 percent on every track and that of RGPE-EI by 19 to 78 percent on the time and cost tracks, matches ScaML-EI on the balanced and performance tracks, reduces its area by 16 percent on the time track of the mixed setting and by 60 to 64 percent on the cost track, and has 12 percent more area on the time track of the same-task setting. The EI-per-dollar baselines split, as ALT reduces the area of the target-only GP and of RGPE per dollar by 41 to 74 percent on the cost track but has 25 percent more area than all three on the time track of the mixed setting and 21 to 45 percent more than ScaML-EI per dollar on the time and cost tracks. The ablation row EI/cost, which differs from ALT only in its fixed exponent of one, lies within 10 points of ScaML-EI per dollar in every

Table 2: Reduction of the regret area by ALT at the dollar budget, in percent of the area of the method in the row, averaged over 100 seeds. Columns are the source settings and the utility tracks, and the mixed and same-task settings have Spooky targets only. /cost marks EI per dollar, and the equal mixture fixes $\pi_n = \frac{1}{2}$. Positive values favor ALT, bold marks reductions whose 95 percent paired bootstrap interval lies above zero, and italics mark intervals below zero (intervals in Appendix D.2). The last row gives the regret area of ALT in percent of the utility range.

	Other-task sources				Mixed sources				Same-task sources			
Method	Bal.	Perf.	Time	Cost	Bal.	Perf.	Time	Cost	Bal.	Perf.	Time	Cost
Random search	**28.0**	**20.7**	**28.8**	**38.8**	**51.8**	**62.9**	**58.6**	**91.4**	**53.9**	**63.6**	**72.6**	**89.1**
GP-EI	3.9	**8.3**	-6.3	**8.3**	**26.4**	**42.0**	**27.6**	**85.0**	**29.6**	**43.0**	**52.1**	**80.9**
GP-EI/cost	**14.7**	**13.3**	**7.2**	**9.3**	**23.8**	**42.9**	*-25.2*	**73.8**	**27.1**	**43.9**	**17.0**	**66.7**
RGPE-EI	2.4	**6.5**	-0.8	**8.8**	3.0	0.6	**19.4**	**77.9**	**22.4**	0.6	**39.4**	**70.7**
RGPE-EI/cost	**13.5**	**10.8**	**6.3**	**7.1**	0.9	3.3	*-25.4*	**55.6**	**14.2**	7.8	6.5	**41.4**
ScaML-EI	**4.4**	**11.0**	**11.5**	**5.3**	-1.6	1.0	**16.4**	**63.5**	3.3	-4.2	*-11.8*	**60.4**
ScaML-EI/cost	**10.4**	**14.5**	**9.0**	**5.8**	2.0	**13.0**	*-23.8*	*-33.3*	2.1	**11.4**	*-21.1*	*-45.0*
Ablations of ALT												
without H_2 (BEH)	*-3.5*	-2.5	*-15.4*	*-3.1*	**31.0**	**36.9**	**25.6**	**72.9**	**30.0**	**33.0**	**32.6**	**61.1**
equal mixture	*-2.8*	-0.7	*-12.3*	-0.6	**15.2**	**22.5**	**11.7**	**49.2**	**14.7**	**16.9**	**9.1**	**12.7**
EI ($\alpha = 0$)	-1.0	1.8	**2.8**	**3.0**	-2.6	-1.4	**16.4**	**64.6**	2.9	-2.3	**6.9**	**58.7**
EI/cost ($\alpha = 1$)	**7.8**	**4.4**	**5.7**	**2.2**	1.6	**9.0**	*-21.6*	*-25.9*	**6.4**	**9.5**	*-11.2*	*-46.1*
cost cooling	**3.5**	2.1	**2.8**	1.4	4.2	**10.0**	*-17.9*	*-21.6*	**7.9**	**12.9**	*-6.6*	*-22.6*
Regret area of ALT (%)	6.0	8.0	6.3	5.1	2.9	2.6	2.2	0.5	2.8	2.5	1.5	0.7

cell of these settings, so most of the gap is due to the exponent rather than the model. A fixed full discount performs better there, while the adaptive exponent starts between 0.5 and 0.7 at evaluation 10 and first reaches 0.9 at evaluation 40 on the same-task cost track.

At a fixed count of 60 evaluations, where a cheap run earns no credit, ALT stays within -3 to 8 percent of the three EI baselines under task shift, with intervals above zero for ScaML-EI on the performance, time and cost tracks, reduces the area of every EI-per-dollar baseline by 9 to 30 percent, and has 4 to 7 percent more area than its own EI ablation on the balanced, performance and time tracks, a loss it trades for its gains at the dollar budget. With a same-task source at a fixed count, ALT keeps its gains over the target-only GP but trails ScaML-EI and RGPE-EI by 6 to 19 percent on the balanced track of the mixed setting (Appendix D.3).

The weight of the behavior hypothesis at evaluation 60 is higher with other-task sources (0.54, 0.30 and 0.11 with one, two and three sources) than with same-task sources (0.05 and 0.02 with one and two) or in the mixed setting (below 0.01), so the evidence moves toward the objective hypothesis as sources are added and settles on it once a source shares the task. Under task shift this weight is too low on the time track, where ALT has 15 percent more area than BEH and 12 percent more than the equal mixture, whereas with a same-task source ALT reduces their areas by 26 to 73 and 9 to 49 percent. The adaptive exponent reduces the area of the EI ablation by 3 percent on the time and cost tracks under task shift and by 59 to 65 percent on the cost track with a same-task source. Appendix D.4 gives the average exponent by setting and track.

7 Conclusion

In summary, we recorded complete configuration landscapes of one LLM agent on four development tasks, found that several behavior features have more consistent configuration effects across tasks than the evaluated utilities, and built ALT, which lets the target runs weight behavior-level against objective-level transfer and learns its cost discount from the same evidence.

The study has clear limits. Four tasks and six landscapes from one agent stack are a small basis, and the same-task and mixed settings rest on a single task. Each landscape is one execution of every configuration, so the best recorded configuration is itself a noisy target, and the sources are complete landscapes, which a deployer rarely has. ALT is not uniformly best. It improves over its behavior-only ablation when same-task sources are available, but gives up part of that ablation's time-track gain under task shift and trails a fixed full cost discount on several time- and cost-focused settings with a same-task source. Future directions include additional factors such as multi-agent orchestration, in-flight intervention that acts on the trajectory during a run using the same behavior features, more flexible prompt construction, and transfer from partial source landscapes.

AI USE STATEMENT

In this work, we used generative AI tools to implement methods, namely the code for ALT, its ablations, the baselines, the experiment launcher, and the replay and evaluation pipeline, written from specifications given by the authors, to clean and reformat the dataset through the export of recorded runs into the released tables, to provide feedback on research methodology through the selection and implementation of baselines and the evaluation protocol, to interpret results by drafting the description of the result tables, and to assist in writing the derivations of Appendix C, which restate standard results for the implemented model. We have not used generative AI tools to develop the conceptual framework or the method, to propose or refine hypotheses, to generate synthetic data, or to formulate mathematical claims, since the research questions, the two-level transfer idea, the variants that use action-trajectory information, and the experimental design were proposed by the authors. Translation and qualitative or thematic data analysis are not applicable to this work. Additionally, we used generative AI tools to create and edit software code, to draft and edit parts of the paper for readability, to create scientific figures through plotting code, to identify and summarize related literature, and to format references, while the logic, structure, and claims of the paper were produced by the authors. We have reviewed all AI-assisted work. All code was verified against its specification and its outputs were checked by the authors, every reported number was recomputed by the authors from the released data, every cited work was checked against the publisher record, and all AI-edited text was checked by the authors for accuracy. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

All data and code needed to reproduce this paper are provided as supplementary material and will be released publicly upon acceptance. The supplementary material contains the six configuration landscapes as linked run, action, and development-report tables with a landscape manifest, the experiment launcher with its configuration catalog, prompts, tool schemas, skill texts, container recipes, and grading scripts, and the replay code for ALT, every baseline, and every ablation, together with the scored results of every replayed search and the notebooks that regenerate every table and figure of the paper from them in minutes. Section 3 and Appendix A describe the tasks, factors, execution environment, grading, cost accounting, and action labeling, Appendix B defines the behavior features, Appendix C gives the derivations, the estimation procedure with its bounds, starts, and refit schedule, and the feature selection rule, and Appendix D gives the scenarios, budgets, initial-design sampler, score, bootstrap, and baseline implementations. Search replay is deterministic given the seed, so rerunning any policy reproduces its score exactly, and regenerating all search paths from the landscapes takes about 1,600 CPU hours. Re-collecting a landscape requires access to the served models through the Claude Agent SDK and about 1,460 GPU hours at the six-hour walltime, whereas every search experiment in the paper runs on the released landscapes alone.

REFERENCES

Lakshya A. Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J. Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective prompt evolution can outperform reinforcement learning. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2507.19457.

Raul Astudillo and Peter I. Frazier. Thinking inside the box: A tutorial on grey-box Bayesian optimization. In *Proceedings of the 2021 Winter Simulation Conference*. IEEE, 2021. URL https://arxiv.org/abs/2201.00272.

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, 2021. URL https://arxiv.org/abs/2111.06537.

Tianyi Bai, Yang Li, Yu Shen, Xinyi Zhang, Wentao Zhang, and Bin Cui. Transfer learning for Bayesian optimization: A survey. *arXiv preprint arXiv:2302.05927*, 2023. URL https://arxiv.org/abs/2302.05927.

Matthew Barker, Andrew Bell, Evan Thomas, James Carr, Thomas Andrews, and Umang Bhatt. Faster, cheaper, better: Multi-objective hyperparameter optimization for LLM and RAG systems. *arXiv preprint arXiv:2502.18635*, 2025. URL https://arxiv.org/abs/2502.18635.

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*, 2025. URL https://openreview.net/forum?id=6s5uXNWGIh.

Lichang Chen, Jiuhai Chen, Tom Goldstein, Heng Huang, and Tianyi Zhou. InstructZero: Efficient instruction optimization for black-box large language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 6503–6518, 2024. URL https://proceedings.mlr.press/v235/chen24e.html.

Alejandro Cuadron, Dacheng Li, Wenjie Ma, Xingyao Wang, Yichuan Wang, Siyuan Zhuang, Shu Liu, Luis Gaspar Schroeder, Tian Xia, Huanzhi Mao, Nicholas Thumiger, Aditya Desai, Ion Stoica, Ana Klimovic, Graham Neubig, and Joseph E. Gonzalez. The danger of overthinking: Examining the reasoning-action dilemma in agentic tasks. *arXiv preprint arXiv:2502.08235*, 2025. URL https://arxiv.org/abs/2502.08235.

Matthias Feurer, Benjamin Letham, Frank Hutter, and Eytan Bakshy. Practical transfer learning for Bayesian optimization. *arXiv preprint arXiv:1802.02219*, 2022. URL https://arxiv.org/abs/1802.02219.

Roman Garnett. *Bayesian Optimization*. Cambridge University Press, 2023. URL https://bayesoptbook.com/.

Zijian He, Reyna Abhyankar, Vikranth Srivatsa, and Yiying Zhang. Cognify: Supercharging Gen-AI workflows with hierarchical autotuning. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, pp. 932–943. Association for Computing Machinery, 2025. doi: 10.1145/3711896.3736884. URL https://doi.org/10.1145/3711896.3736884.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2408.08435.

Wenyue Hua, Sripad Karne, Qian Xie, Armaan Agrawal, Nikos Pagonas, Kostis Kaffes, and Tianyi Peng. AgentOpt v0.1 technical report: Client-side optimization for LLM-based agent. *arXiv preprint arXiv:2604.06296*, 2026. URL https://arxiv.org/abs/2604.06296.

Frank Hutter, Holger Hoos, and Kevin Leyton-Brown. An efficient approach for assessing hyperparameter importance. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pp. 754–762, 2014. URL https://proceedings.mlr.press/v32/hutter14.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.

Sayash Kapoor, Benedikt Stroebl, Zachary S. Siegel, Nitya Nadgir, and Arvind Narayanan. AI agents that matter. *Transactions on Machine Learning Research*, 2025. URL https://openreview.net/forum?id=Zy4uFzMviZ.

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.

Gustavo Malkomes and Roman Garnett. Automating Bayesian optimization with Bayesian optimization. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/2b64c2f19d868305aa8bbc2d72902cc5-Abstract.html.

Krista Opsahl-Ong, Michael J. Ryan, Josh Purtell, David Broman, Christopher Potts, Matei Zaharia, and Omar Khattab. Optimizing instructions and demonstrations for multi-stage language model programs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 9340–9366. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-main.525. URL https://aclanthology.org/2024.emnlp-main.525/.

Florian Pfisterer, Lennart Schneider, Julia Moosbauer, Martin Binder, and Bernd Bischl. YAHPO Gym: An efficient multi-objective multi-fidelity benchmark for hyperparameter optimization. In *Proceedings of the First International Conference on Automated Machine Learning*, volume 188 of *Proceedings of Machine Learning Research*, 2022. URL https://proceedings.mlr.press/v188/pfisterer22a.html.

Sebastian Pineda Arango, Hadi S. Jomaa, Martin Wistuba, and Josif Grabocka. HPO-B: A large-scale reproducible benchmark for black-box HPO based on OpenML. In *NeurIPS Datasets and Benchmarks Track*, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/hash/ec8956637a99787bd197eacd77acce5e-Abstract-round2.html.

Jon Saad-Falcon, Adrian Gamarra Lafuente, Shlok Natarajan, Nahum Maru, Hristo Todorov, Etash Kumar Guha, E. Kelly Buchanan, Mayee F. Chen, Neel Guha, Christopher Ré, and Azalia Mirhoseini. An architecture search framework for inference-time techniques. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 52475–52507, 2025. URL https://proceedings.mlr.press/v267/saad-falcon25a.html.

Yu Shang, Yu Li, Keyu Zhao, Likai Ma, Jiahe Liu, Fengli Xu, and Yong Li. AgentSquare: Automatic LLM agent search in modular design space. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.06153.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/1b623663fd9b874366f3ce019fdfdd44-Abstract-Conference.html.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, volume 25, pp. 2951–2959, 2012. URL https://papers.nips.cc/paper_files/paper/2012/hash/05311655a15b75fab86956663e1819cd-Abstract.html.

Kevin Swersky, Jasper Snoek, and Ryan P. Adams. Multi-task Bayesian optimization. In *Advances in Neural Information Processing Systems*, volume 26, pp. 2004–2012, 2013. URL https://papers.nips.cc/paper_files/paper/2013/hash/f33ba15effa5c10e873bf3842afb46a6-Abstract.html.

Kevin Swersky, Jasper Snoek, and Ryan P. Adams. Freeze-thaw Bayesian optimization. *arXiv preprint arXiv:1406.3896*, 2014. URL https://arxiv.org/abs/1406.3896.

Aditya Taparia, Som Sagar, and Ransalu Senanayake. Learning to configure agentic AI systems. *arXiv preprint arXiv:2602.11574*, 2026. URL https://arxiv.org/abs/2602.11574.

Petru Tighineanu, Lukas Grossberger, Paul Baireuther, Kathrin Skubch, Stefan Falkner, Julia Vinogradska, and Felix Berkenkamp. Scalable meta-learning with Gaussian processes. In *International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, pp. 1981–1989, 2024. URL https://proceedings.mlr.press/v238/tighineanu24a.html.

Varun Ursekar, Apaar Shanker, Veronica Chatrath, Yuan Xue, and Samuel Marc Denton. VeRO: A harness for agents to optimize agents. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2602.22480.

Jan N. van Rijn and Frank Hutter. Hyperparameter importance across datasets. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 2367–2376. Association for Computing Machinery, 2018. doi: 10.1145/3219819.3220058. URL https://arxiv.org/abs/1710.04725.

Chi Wang, Xueqing Liu, and Ahmed Hassan Awadallah. Cost-effective hyperparameter optimization for large language model generation inference. In *Proceedings of the Second International Conference on Automated Machine Learning*, volume 224 of *Proceedings of Machine Learning Research*, pp. 21/1–17, 2023. URL https://proceedings.mlr.press/v224/wang23b.html.

Ningning Wang, Xavier Hu, Pai Liu, He Zhu, Yue Hou, Heyuan Huang, Shengyu Zhang, Jian Yang, Jiaheng Liu, Ge Zhang, Changwang Zhang, Jun Wang, Yuchen Eleanor Jiang, and Wangchunshu Zhou. Efficient agents: Building effective agents while reducing cost. *arXiv preprint arXiv:2508.02694*, 2025. URL https://arxiv.org/abs/2508.02694.

Zi Wang, George E. Dahl, Kevin Swersky, Chansoo Lee, Zachary Nado, Justin Gilmer, Jasper Snoek, and Zoubin Ghahramani. Pre-trained Gaussian processes for Bayesian optimization. *Journal of Machine Learning Research*, 25(212):1–83, 2024. URL https://jmlr.org/papers/v25/23-0269.html.

Aaron Wilson, Alan Fern, and Prasad Tadepalli. Using trajectory data to improve Bayesian optimization for reinforcement learning. *Journal of Machine Learning Research*, 15(8):253–282, 2014. URL https://jmlr.org/papers/v15/wilson14a.html.

Martin Wistuba and Josif Grabocka. Few-shot Bayesian optimization with deep kernel surrogates. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2101.07667.

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, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/d14c355d5e88cff437a6303d2d716252-Abstract.html.

Zhangchen Xu, Junda Chen, Yue Huang, Dongfu Jiang, Jiefeng Chen, Hang Hua, Zijian Wu, Zheyuan Liu, Zexue He, Lichi Li, Shizhe Diao, Jiaxin Pei, Jinsung Yoon, Hao Zhang, Mengdi Wang, Radha Poovendran, Misha Sra, Alex Pentland, and Zichen Chen. AutoLab: Can frontier models solve long-horizon auto research and engineering tasks? *arXiv preprint arXiv:2606.05080*, 2026. URL https://arxiv.org/abs/2606.05080.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://arxiv.org/abs/2405.15793.

Chris Ying, Aaron Klein, Eric Christiansen, Esteban Real, Kevin Murphy, and Frank Hutter. NAS-Bench-101: Towards reproducible neural architecture search. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 7105–7114, 2019. URL https://proceedings.mlr.press/v97/ying19a.html.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xiong-Hui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. AFlow: Automating agentic workflow generation. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=z5uVAKwmjf.

APPENDIX

Appendix A supports the experiment and dataset of Section 3, Appendix B the analysis of Section 4, Appendix C the method of Section 5, and Appendix D the evaluation of Section 6.

A EXPERIMENT DETAILS

A.1 SCREENING AND TASK SELECTION

Tasks were selected in two screening rounds. Each round ran the nine combinations of requested model and reasoning effort on every candidate task, with the remaining factors fixed, under the six-hour walltime (72 runs on eight candidates in the first round and 54 runs on six in the second). Table 3 summarizes the spread of the reference-normalized performance across the nine runs of each candidate, together with runtime and cost. We chose two MLE-bench and two AutoLab tasks for diversity of task kind, preferring candidates with a larger performance spread and manageable runtime and cost. In the first round, Ventilator Pressure Prediction and Spooky Author Identification had the largest spreads, and Adversarial Splay was preferred over the other AutoLab candidate, whose spread was small and cost high. The second round added Scaling Law as an AutoLab task of a different kind, whereas the five image tasks of that round showed little performance variation across the nine configurations. The first round also fixed the resource caps of Table 1, since the 75th and 50th percentiles of the number of counted tool calls across its runs are 86 and 45.

Table 3: Screening rounds. For each candidate, the range and interquartile range of the reference-normalized performance y_p over the nine screening runs, the median runtime, and the mean API-equivalent cost. Scaling Law had no reference at screening time, so its raw test perplexity range is shown. The four selected tasks are Ventilator Pressure Prediction, Spooky Author Identification, Adversarial Splay, and Scaling Law.

Round	Task	Source	Kind	y_p range	y_p IQR	Runtime (h)	Cost (USD)
1	Ventilator Pressure Prediction	MLE-bench	sequence regression	0.42	0.13	5.0	4.65
1	Spooky Author Identification	MLE-bench	text classification	0.27	0.10	2.6	3.46
1	Nomad2018 Conductors	MLE-bench	tabular regression	0.18	0.08	3.3	4.53
1	Denoising Dirty Documents	MLE-bench	image restoration	0.12	0.12	1.6	1.52
1	Adversarial Splay	AutoLab	combinatorial optimization	0.10	0.05	4.8	6.04
1	PetFinder Pawpularity	MLE-bench	multimodal regression	0.07	0.04	3.4	3.24
1	Adaptive Compression	AutoLab	iterative optimization	0.05	0.01	1.1	10.12
1	LMSYS Chatbot Arena	MLE-bench	text preference	0.04	0.01	3.7	6.71
2	Scaling Law	AutoLab	LM pretraining	perplexity 18.5–22.0		5.3	4.51
2	TGS Salt Identification	MLE-bench	image segmentation	0.09	0.02	4.5	2.96
2	APTOS 2019 Blindness Detection	MLE-bench	image ordinal regression	0.04	0.03	4.8	2.03
2	Plant Seedlings Classification	MLE-bench	image classification	0.04	0.02	4.6	2.93
2	Paddy Disease Classification	MLE-bench	image classification	0.02	0.00	5.1	2.86
2	Histopathologic Cancer Detection	MLE-bench	image classification	0.01	0.00	3.4	1.67

Table 4 lists the selected tasks. The two MLE-bench tasks (Chan et al., 2025) are Kaggle competitions with the public training data in the workspace and the private test labels held by the grader. The two AutoLab tasks (Xu et al., 2026) are Adversarial Splay, in which the agent writes an access sequence that maximizes the number of rotations of a splay tree, and Scaling Law, in which the agent pretrains a GPT-style language model on WikiText-103 within the walltime. The references m_0 of Section 3 are the pinned MLE-bench leaderboard medians, the uniform-random baseline declared by AutoLab for Adversarial Splay, and the median test perplexity of the 242 valid Scaling Law runs, computed at export time.

Table 4: The four tasks and the performance references m_0 of Section 3.

Task	Source	Kind	Metric (direction)	Artifact	Reference m_0
Spooky Author Identification	MLE-bench	text classification	log loss (lower)	`submission.csv`	0.4188
Ventilator Pressure Prediction	MLE-bench	sequence regression	masked MAE (lower)	`submission.csv`	0.1638
Adversarial Splay	AutoLab	combinatorial optimization	rotations (higher)	`accesses.txt`	48,656
Scaling Law	AutoLab	LM pretraining	test perplexity (lower)	`checkpoint.pt`	20.59

Table 5: Tools of the three interface levels.

Tool	Levels	Description shown to the agent
`run_command`	T0–T2	Run a shell command in the offline task container; only the workspace is mounted.
`validate_artifact`	T0–T2	Check the required terminal artifact without revealing private performance.
`report_metric`	T0–T2	Record one development or validation metric event; this is not private grading.
`commit`	T0–T2	Validate and submit the current terminal artifact for sealed evaluation and end the run.
`abstain`	T0–T2	Deliberately end the run without submitting a terminal artifact.
`list_files`	T1–T2	List files under a workspace-relative directory.
`read_file`	T1–T2	Read a UTF-8 text file in the workspace with optional line bounds.
`write_file`	T1–T2	Create or overwrite one UTF-8 text file in the workspace.
`trial_start`	T2	Open a tracked trial with a name and a one-line intent.
`trial_record`	T2	Record a development-metric outcome for a tracked trial.
`trial_list`	T2	List all tracked trials with their recorded outcomes.
`trial_compare`	T2	Compare recorded outcomes across tracked trials sharing a metric.
`budget_status`	T2	Report the counted-tool budget, wall-clock use, and trial count.

A.2 Factors

Model. The three levels request `claude-sonnet-5`, `claude-opus-5`, and `claude-fable-5` through the Claude Agent SDK (version 0.2.127), and the served model of every response is recorded. The provider may serve an allowlisted fallback model for part of a run, and such runs count as observations of the requested level and are flagged in the released tables. This happened in eight Fable 5 runs on Adversarial Splay, partly served by `claude-opus-4-8` and priced at that model's rates, and in no other landscape.

Effort. The levels medium, high, and max are passed to the SDK as its effort option. The batch environment is checked before every run so that no environment variable overrides the model or effort of the manifest.

Resources. The cap counts calls of `run_command`, `report_metric`, `list_files`, `read_file`, `write_file`, and the four trial tools, whereas `validate_artifact` and `budget_status` are never counted. Every tool result carries the running count and, under a cap, the remaining budget. Once the cap is reached, counted tools return an error stating that the budget is exhausted and that the agent may still validate, commit, or abstain. A guard ending the run after 30 consecutive refused calls was never triggered.

Skills. Levels S1 and S2 add one skill document to the SDK session (Appendix A.3), and S0 adds none.

Tools. The interface is an MCP server whose schemas are exactly the tools of the level (Table 5). Since `run_command` executes shell commands at every level, the factor changes the exposed interface, not the set of operations the agent can perform.

A.3 Prompts and skills

The system prompt concatenates the base prompt, the budget sentence of the resource level, and the orchestration sentence. The task prompt concatenates the family prompt of the benchmark, the walltime of the run, and the task description in `TASK.md` of the workspace (the competition description for MLE-bench, the task card for AutoLab).

> **`Base prompt.`** `You are the machine-learning engineer for one experimental episode. Work only through the supplied tools. Your command tool executes inside an offline container that exposes only the current workspace. Do not access paths outside the workspace. You may install nothing from the`

internet; prefer the packages already present. Inspect the task, build and evaluate candidate solutions, and leave the required terminal artifact at the exact path specified in the task prompt. Use `report_metric` after every meaningful development evaluation so the experiment records the validation trajectory in structured form. The development metric is not the private terminal score. Set `terminal_score_scale=true` only when the reported metric has the same definition, direction, and scale as the terminal score. Use `validate_artifact` before finishing. End the run yourself: call `commit` to submit the terminal artifact for sealed evaluation, or `abstain` if you determine that no valid completion is possible.

Budget sentence, capped (R2, R3). This run permits at most {cap} counted workspace tool interactions. validate_artifact and, when available, budget_status do not consume this budget. Every tool result includes a running count and the remaining budget. When the budget is exhausted, counted tools stop executing; you can still validate the artifact, inspect the budget when that tool is available, commit it, or abstain.

Budget sentence, uncapped (R1). This run has no cap on counted workspace tool interactions; only the wall-time limit applies. Every workspace tool result includes a running count of your counted interactions. Checking the artifact and ending the run never consume budget.

Orchestration. You complete this run alone, from start to finish. Do not delegate or simulate subagents.

Family prompt, MLE-bench. The public MLE-bench task files are in `/workspace/data`, and the competition description is `/workspace/TASK.md`. Build the best valid submission you can. Required terminal artifact: `/workspace/submission.csv`. The private MLE-bench split is graded after the episode. Use visible training data and your own validation scheme. `validate_artifact` checks only submission shape and columns; it never reveals private performance.

Family prompt, AutoLab. The AutoLab instruction is `/workspace/TASK.md`. All agent-visible files are in `/workspace`. Follow the edit restrictions in the instruction exactly. Final grading runs after the episode. Use the local evaluator described by the task for development, call `report_metric` for its results, and call `validate_artifact` before finishing.

Skill S1, working principles. Plan before acting: identify what the task rewards, what a minimal valid solution looks like, and which uncertainties matter most. Spend early effort on the cheapest probes that reduce the largest uncertainty. Establish a valid baseline artifact early, then improve it. A weak valid artifact always beats a strong invalid one. Change one thing at a time and measure it. Evaluate before and after every substantive change on the same development data, and record every meaningful evaluation with `report_metric`. Trust measurements over impressions. When a result surprises you, reproduce it before building on it. Validate assumptions about data, formats, and interfaces directly; do not carry an unverified assumption across steps. Budget deliberately. Weigh the expected improvement of further work against the remaining time and experimental-work budget shown in tool results.

```
Stop when the expected marginal gain no longer justifies
the cost, validate the artifact, and commit.  If no valid
completion is achievable, abstain deliberately rather than
submitting an artifact you know to be invalid.


Skill S2, working procedures.  1.  Orientation (do first).
(1) Inspect the task instructions and every provided file
before writing code.  (2) Write PLAN.md in the workspace
containing:  the objective in one sentence; the terminal
artifact path and its required format; a minimal-baseline
plan; 2-4 candidate improvements ordered by expected value
per unit cost; the development-evaluation scheme you will
use.  (3) Confirm the execution environment:  run a one-line
probe of the interpreter, key packages, and hardware
before any long computation.  2.  Baseline.  (4) Build the
simplest end-to-end pipeline that produces the terminal
artifact.  (5) Run `validate_artifact`.  Do not proceed
until it reports valid.  (6) Evaluate the baseline with
your development scheme and record it with `report_metric`.
3.  Improvement loop (repeat).  (7) Take the highest-value
item from PLAN.md.  State in TRIALS.md one line before
running:  trial number, what changes, expected effect.  (8)
Make the change, run it, evaluate on the same development
data.  (9) Append the outcome to TRIALS.md:  metric value,
comparison to the current best, decision (keep / revert).
Record the evaluation with `report_metric`.  (10) Revert
changes that did not help before starting the next trial.
(11) After each trial, re-check the remaining time and
experimental-work budget shown in tool results, and update
the plan ordering if priorities changed.  4.  Error recovery
(whenever a step fails).  (12) Read the complete error
output before changing anything.  (13) Reproduce the failure
with the smallest, fastest command that still fails.  (14)
Fix one candidate cause, re-run the minimal reproduction,
and only then re-run the full step.  If three successive
fixes fail, return to the last known-good state recorded in
TRIALS.md and choose a different approach.  5.  Termination
checklist (before ending).  (15) Ensure the best kept
trial's artifact is the one at the terminal path.  (16) Run
`validate_artifact`; resolve any failure.  (17) Record the
final development evaluation with `report_metric`.  (18)
Commit.  If no valid artifact is achievable within the
remaining budget, state the reason in TRIALS.md and abstain
deliberately.
```

A.4 EXECUTION

Every run receives a dedicated NVIDIA A100 GPU (80 GB), 16 CPU cores, 128 GB of memory, and a walltime of 21,600 seconds, and starts in a fresh workspace with the public task data copied in. Commands execute in an Apptainer container built from a pinned PyTorch image with pinned scientific libraries and no network access. Two revisions of the image were used, with the same base digest and package pins, and the second, which adds the language-model training libraries of the Scaling Law starter, was used for Scaling Law. The SDK session runs on the host with at most 128,000 output tokens per response, tool results truncated to 16,000 characters, and command logs capped at 10 MB. Each landscape was collected as nine batches of 27 configurations, where every batch contains each of the nine model–effort pairs with all three tool interfaces and the resource–skill pairs rotate across batches, so every level of every factor appears nine times in every batch. Provider and infrastructure failures, such as authentication errors, container faults, or node failures before the agent acted, were rerun in place with the failed attempt retained, and outcomes chosen by the agent were never rerun. The run manifests, container recipes, configuration catalog, and launcher are released.

Table 6: Rate card in US dollars per million tokens.

Served model	Input	Cache write (5 min)	Cache write (1 h)	Cache read	Output
`claude-sonnet-5`	2.00	2.50	4.00	0.20	10.00
`claude-opus-5`	5.00	6.25	10.00	0.50	25.00
`claude-opus-4-8`	5.00	6.25	10.00	0.50	25.00
`claude-fable-5`	10.00	12.50	20.00	1.00	50.00

A.5 Outcomes and accounting

Grading. The terminal artifact preserved by the launcher is graded twice in independent processes, and the run is exported only if both grades agree. MLE-bench artifacts are scored by the benchmark's grader against the private test labels, and Adversarial Splay artifacts by the AutoLab verifier under a hidden seed, which checks validity and counts rotations. Scaling Law checkpoints are scored by the perplexity over non-overlapping 512-token windows of the WikiText-103 test set, which is never mounted during a run, with tensors-only deserialization and a shorter context evaluated at its own length. A missing, unloadable, or malformed artifact receives $y_p = -1$, whereas a grader failure is an infrastructure failure and is retried.

Runtime. Runtime is the elapsed time from workspace setup to the end of the run, including model latency, and the released tables also give the time of the last action and the idle tail.

Cost. Runs executed under a subscription, so cost is the API-equivalent price of the recorded token usage under the rate card of Table 6, by served model. The cache-write price depends on a cache lifetime that the usage record does not always classify, so complete records price unclassified cache writes at the upper edge of the band, and both edges are released. For 21 runs that reached the walltime the final usage total is missing, so their usage is reconstructed from the per-message records, priced at the lower edge, and flagged as censored. Replay uses the recorded cost of every run as is.

Action labels. Action labels are a pure function of the tool name and the command text of each call, and the released action table carries the label of every component. A compound shell command is split at unquoted newlines, semicolons, logical operators, and pipelines, each substantive segment being one component, while heredoc payloads stay opaque, shell comments are removed, and control syntax such as a leading `cd` stays attached to its command. The classes are inspection (file listing and reading, non-writing shell inspection), editing (file writes and file-mutating shell commands), execution or training (the default for running code), development evaluation (metric reports, artifact validation, trial comparison, and test or evaluation scripts), environment (package installation and setup), checkpointing (trial records and version-control milestones), commit, abstain, and coordination under multi-agent orchestration. Within a shell component the precedence is environment, checkpoint, evaluation, inspection, editing, and execution. Only seven classes occur, since the catalog is single-agent and no run abstained. Refused calls after a cap is exhausted are events, not actions.

A.6 Collection audit

Table 7 gives the audit per landscape. Of the 1,458 runs, 1,435 ended by commit, 21 by walltime, and two by a final assistant turn without a terminal call, and the artifacts of the latter two were valid and graded. Two artifacts are invalid, both from walltime stops. No run abstained, and no run was ended by the refusal guard or the controller.

B Behavior features

Table 8 defines the seventeen behavior features that are candidates for f_{src} in Section 5.1. All are computed from the nonterminal tool calls and development reports of a run, with the active duration taken as the time of the last action, and counts and the median call interval enter as $\log(1 + x)$. Table 9 gives the cross-task and same-task correlations of every feature in the format of Fig. 2(e), and Fig. 3 the main effect of every factor on the outcomes and on the balanced utility.

Table 7: Collection audit per landscape: recorded runtime, API-equivalent cost, corpus sizes, stopping sources, invalid artifacts, and cost-censored runs. Eight Adversarial Splay runs were partly served by a fallback model (Appendix A.2).

Landscape	Runs	Hours	Cost ($)	Tool calls	Components	Reports	Commit	Walltime	End turn	Invalid	Censored
Adversarial Splay	243	863.1	1,052.7	11,297	29,388	887	239	4	0	0	4
Scaling Law	243	1,256.4	801.1	8,902	22,780	757	241	2	0	1	2
Spooky r1	243	547.1	724.3	11,052	24,451	1,115	235	6	2	0	6
Spooky r2	243	551.2	749.6	11,157	24,150	1,102	238	5	0	0	5
Spooky r3	243	540.3	756.4	11,435	25,981	1,167	240	3	0	1	3
Ventilator	243	1,064.5	518.1	8,486	19,003	876	242	1	0	0	1
Total	1,458	4,822.5	4,602.1	62,329	145,753	5,904	1,435	21	2	2	21

Table 8: The seventeen behavior features. Counts and the median call interval are transformed by $\log(1 + x)$.

Feature	Definition
calls	Number of distinct nonterminal tool calls.
components	Number of nonterminal action components.
reports	Number of development reports.
train components	Number of execution or training components.
eval components	Number of development-evaluation components.
edit components	Number of editing components.
inspect components	Number of inspection components.
tool error share	Fraction of nonterminal calls whose result was an error.
median call interval	Median time between consecutive tool calls, or zero with fewer than two calls.
eval after train	Fraction of calls with a training component that carry an evaluation component or are followed by one before the next training call, or zero without training.
first report position	Elapsed time of the first development report divided by the active duration, or one if there is no report.
early token fraction	Fraction of the cumulative token consumption that occurred in the first quarter of the active duration.
late token fraction	Fraction that occurred after three quarters of the active duration.
report token fraction	Cumulative token consumption at the first development report divided by the final consumption, or one if there is no report.
dev gain time	Gain-weighted mean position in the active duration of the improvements of the running-best development metric, using only reports declared on the terminal score scale with a finite value and computed within each stream of reports sharing a metric name, split, and direction, or one without gains.
dev late gain fraction	Share of those improvements after half the active duration, or zero without gains.
dev checkpoints	Number of development reports declared on the terminal score scale with a finite value.

C Derivations and estimation details

Throughout, X_n is the set of the n target configurations run so far, $\widetilde{\mathbf{u}} \in \mathbb{R}^n$ the vector of their standardized utilities, and $\mathbf{1}$ the all-ones vector.

C.1 Standardization

With ν_n and ρ_n the mean and standard deviation of the source utilities pooled with $u_1, \ldots, u_n$, the standardized target utility is $\widetilde{u}_i = (u_i - \nu_n)/\rho_n$, and predictions are mapped back as $\mu = \nu_n + \rho_n \widetilde{\mu}$ and $v = \rho_n^2 \widetilde{v}$. The GP of source s is fitted to u_s standardized by the mean $\nu^{(s)}$ and standard deviation $\rho^{(s)}$ of that source alone, and its posterior is converted to utility units once, $\mu^{(s)} = \nu^{(s)} + \rho^{(s)} \widetilde{\mu}^{(s)}$ and $k^{(s)} = (\rho^{(s)})^2 \widetilde{k}^{(s)}$. Under H_2 the prior mean in the current target units is $m_2(x) = \big(\sum_s w_s \mu^{(s)}(x) - \nu_n\big)/\rho_n$, the mean of $\widetilde{g}$ in Eq. (2), and the transferred covariance is the second term of Eq. (4). Under H_1 the features are standardized on the sources (Appendix C.7) and the prior mean is the constant $m_1(x) = b$.

Table 9: Correlations of the seventeen behavior features between tasks (mean and range over the six task pairs of the factor-effect correlation, mean rank correlation) and between the three Spooky repetitions, sorted by the cross-task factor-effect correlation.

	Across tasks			Same task	
Behavior feature	effects	range	ranks	effects	ranks
tool error share	0.92	[0.82, 0.96]	0.76	0.99	0.85
first report position	0.91	[0.86, 0.97]	0.38	0.97	0.54
median call interval	0.90	[0.81, 0.95]	0.57	0.99	0.79
components	0.87	[0.76, 0.96]	0.58	0.99	0.80
calls	0.85	[0.70, 0.92]	0.57	0.99	0.78
report token fraction	0.84	[0.67, 0.97]	0.36	0.97	0.55
inspect components	0.84	[0.78, 0.94]	0.51	0.98	0.77
train components	0.83	[0.60, 0.96]	0.51	0.99	0.77
dev checkpoints	0.82	[0.70, 0.93]	0.35	0.98	0.55
reports	0.79	[0.56, 0.92]	0.35	0.98	0.58
eval components	0.76	[0.45, 0.93]	0.36	0.97	0.62
edit components	0.76	[0.64, 0.91]	0.40	0.98	0.77
dev gain time	0.71	[0.42, 0.91]	0.25	0.97	0.45
eval after train	0.41	[-0.17, 0.79]	0.13	0.96	0.59
dev late gain fraction	0.28	[-0.26, 0.59]	0.14	0.78	0.24
late token fraction	0.19	[-0.48, 0.90]	0.10	0.88	0.22
early token fraction	0.17	[-0.54, 0.75]	0.08	0.79	0.23

Algorithm 1 ALT on one target task

Require: source landscapes $\mathcal{D}^{(1)}, \dots, \mathcal{D}^{(S)}$, utility weights (a_t, a_c), budget C
1: On the sources, select the three most stable features and fit f_{src} (Appendix C.7), fit ϖ (Appendix C.5), and fit one GP per source utility (Appendix C.3)
2: Run the nine initial configurations on the target, $n \leftarrow 9$
3: **while** recorded cost $< C$ **do**
4: Standardize the utilities and convert the source posteriors to the target scale (Appendix C.1)
5: Fit H_1, H_2, and the cost model on $\mathcal{D}_n$ (Appendix C.6)
6: Compute π_n and α_n as in Section 5.2, (μ_n, v_n) by Eq. (5), and x_{n+1} by Eq. (6)
7: Run x_{n+1}, add the result to $\mathcal{D}_n$, $n \leftarrow n + 1$
8: **end while**
9: **return** the best configuration among the runs completed within C

C.2 Behavior hypothesis

Under H_1, $\widetilde{u}(x) = b + \beta^\top f_{\mathrm{src}}(x) + r_{\mathrm{tgt}}(x) + \varepsilon$ with independent $\beta \sim \mathcal{N}(0, \tau^2 I_3)$, $r_{\mathrm{tgt}} \sim \mathcal{GP}(0, k_X)$, and $\varepsilon \sim \mathcal{N}(0, \sigma^2)$. Integrating β out leaves a GP with mean b and covariance $k_X(x, x') + f_{\mathrm{src}}(x)^\top \mathrm{Cov}(\beta) f_{\mathrm{src}}(x')$, which is Eq. (3). The choice $\tau^2 = \left(\frac{1}{243}\sum_{x \in \mathcal{X}} \|f_{\mathrm{src}}(x)\|^2\right)^{-1}$ makes the average over $\mathcal{X}$ of the prior variance $\tau^2 \|f_{\mathrm{src}}(x)\|^2$ of the behavior term equal to one, the variance of the standardized utility. Let K_1 be the matrix of k_1 over X_n plus $\sigma^2 I$, K_X the residual kernel matrix, and $Z_n = [f_{\mathrm{src}}(x_i)^\top]_{i \le n} \in \mathbb{R}^{n \times 3}$. With b fixed, the model is Bayesian linear regression $\widetilde{\mathbf{u}} - b\mathbf{1} = Z_n \beta + \boldsymbol{\eta}$ with correlated errors $\boldsymbol{\eta} \sim \mathcal{N}(0, K_X + \sigma^2 I)$, whose posterior is $V_\beta = \left(\tau^{-2} I_3 + Z_n^\top (K_X + \sigma^2 I)^{-1} Z_n\right)^{-1}$ and $m_\beta = V_\beta Z_n^\top (K_X + \sigma^2 I)^{-1} (\widetilde{\mathbf{u}} - b\mathbf{1})$. Since $K_1 = K_X + \sigma^2 I + \tau^2 Z_n Z_n^\top$, the Woodbury identity gives the form used in the implementation,

$$V_\beta = \tau^2 I_3 - \tau^4 Z_n^\top K_1^{-1} Z_n, \qquad m_\beta = \tau^2 Z_n^\top K_1^{-1} (\widetilde{\mathbf{u}} - b\mathbf{1}), \tag{7}$$

which needs only the factorization of K_1 that the GP posterior needs anyway. The intercept is the generalized least-squares estimate $b = \mathbf{1}^\top K_1^{-1} \widetilde{\mathbf{u}} / \mathbf{1}^\top K_1^{-1} \mathbf{1}$, the maximizer of the marginal likelihood in b at fixed kernel parameters, substituted at every evaluation of the objective.

C.3 Objective hypothesis

A GP with kernel k_X and zero mean is fitted to the standardized utilities of each source over its full landscape, and its posterior $g^{(s)} \sim \mathcal{GP}(\mu^{(s)}, k^{(s)})$ is kept, evaluated on all of $\mathcal{X}$. Following

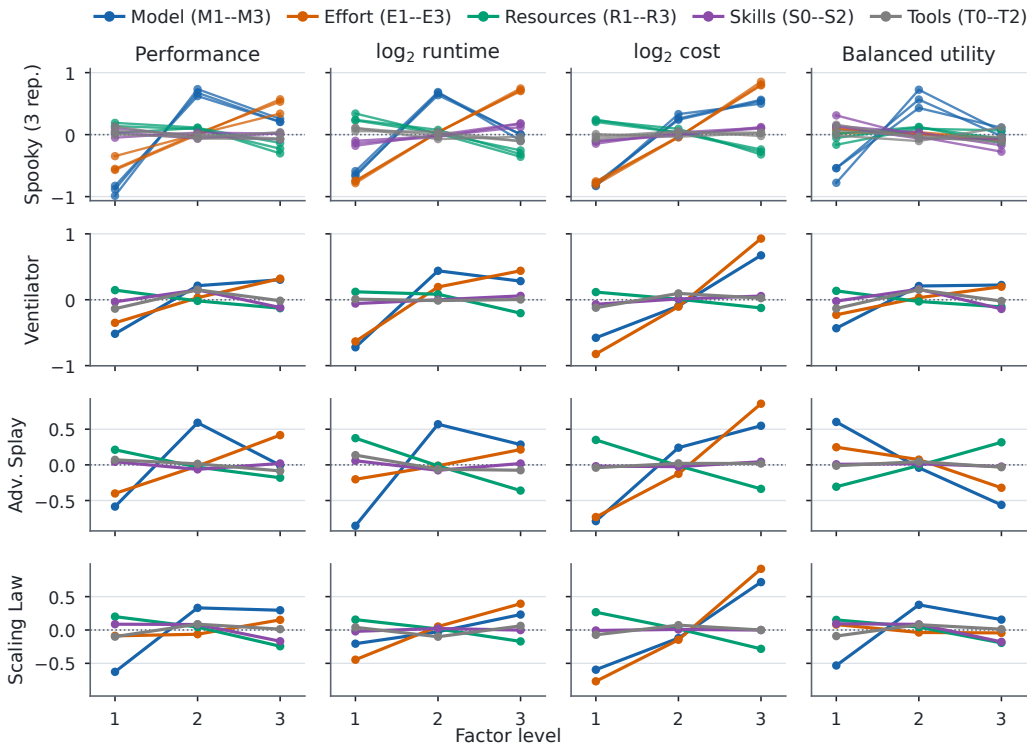

Figure 3: Main effect of every factor on performance, log runtime, log cost, and the balanced utility, per task, in standard deviations of the landscape. A point is the mean over the 81 configurations at that level minus the landscape mean; levels are ordered as in Table 1, and the three Spooky repetitions are drawn as separate lines.

ScaML-GP (Tighineanu et al., 2024), the source posteriors are treated as independent processes and the target utility is their positively weighted sum plus the target residual, so by independence the covariance of the sum is $\sum_s w_s^2 k^{(s)}(x, x')$, which divided by ρ_n^2 and added to the residual covariance gives Eq. (4). Following Tighineanu et al. (2024, Appendix C.6), we place independent $\mathrm{Gamma}(1, 1)$ priors on the positive source weights, whose negative log density contributes $\sum_s w_s$ to the fitting objective, up to an additive constant.

C.4 Predictive distributions, evidence, and mixture

With K_j the matrix of k_j over X_n plus $\sigma_j^2 I$, $\mathbf{k}_j(x)$ the vector of $k_j(x_i, x)$, and $\mathbf{m}_j$ the prior mean at X_n, hypothesis j predicts $\widetilde{\mu}_{j,n}(x) = m_j(x) + \mathbf{k}_j(x)^\top K_j^{-1}(\widetilde{\mathbf{u}} - \mathbf{m}_j)$ and $\widetilde{v}_{j,n}(x) = k_j(x, x) - \mathbf{k}_j(x)^\top K_j^{-1}\mathbf{k}_j(x)$ (Garnett, 2023), mapped to utility units by Appendix C.1. The residual parameters are fitted separately under each hypothesis, so k_X in k_1 and in k_2 are different members of the same family. The log marginal likelihood at the fitted parameters is $\ell_{j,n} = \log \mathcal{N}(\widetilde{\mathbf{u}}; \mathbf{m}_j, K_j) - n \log \rho_n$, where the last term expresses the density in utility units and cancels in π_n, and the prior on the source weights enters the fitting objective but not $\ell_{2,n}$. The weight $\pi_n = e^{\ell_{1,n}}/(e^{\ell_{1,n}} + e^{\ell_{2,n}})$ of Section 5.2 normalizes these marginal likelihoods at the fitted parameters, which is a plug-in, empirical-Bayes approximation to the posterior probability of H_1 under equal prior odds, since it does not integrate over the fitted kernel parameters, the intercept, and the source weights, although β is integrated out. For the mixture, let $J \in \{1, 2\}$ indicate the hypothesis with $P(J = 1) = \pi_n$ and $U \mid J = j \sim \mathcal{N}(\mu_{j,n}, v_{j,n})$, so that $\mathbb{E}[U] = \pi_n \mu_{1,n} + (1 - \pi_n)\mu_{2,n}$ and, by the law of total variance, $\mathrm{Var}(U) = \pi_n v_{1,n} + (1 - \pi_n) v_{2,n} + \pi_n(1 - \pi_n)(\mu_{1,n} - \mu_{2,n})^2$, the moments of Eq. (5). Expected improvement is computed for a Gaussian with these moments, $\mathrm{EI}_n(x) = (\mu_n - u_n^+)\Phi(z) + \sqrt{v_n}\,\phi(z)$ with $z = (\mu_n - u_n^+)/\sqrt{v_n}$.

C.5 Cost model and adaptive exponent

The cost model is a target-only GP with kernel k_X fitted to the standardized $\log_2$ costs of the target runs, refitted whenever the exponent is positive. With predictive mean $\mu_c(x)$ and latent variance $v_c(x)$ in $\log_2$ dollars, the predicted cost is the log-normal mean $\hat{c}_n(x) = \exp\big(\ln 2\,\mu_c(x) + \frac{1}{2}(\ln 2)^2 v_c(x)\big)$. The direction ϖ is the first right singular vector of the column-centered matrix $F \in \mathbb{R}^{243\times 3}$ of f_{src} over $\mathcal{X}$, signed so that $F\varpi$ correlates positively with the factor-effect pattern of $\log_2$ cost on the pooled source runs (the ridge regression of $\log_2$ cost on the one-hot encoding evaluated on $\mathcal{X}$). Under H_1 the contribution $\varpi^\top\beta$ is Gaussian with mean $\varpi^\top m_\beta$ and variance $\varpi^\top V_\beta \varpi$ by Eq. (7), so its posterior probability of being negative is the exponent α_n of Section 5.2, and the probability is invariant to the response scale.

C.6 Estimation

Under H_1 the free parameters are $(\log s^2, \log\lambda_1, \ldots, \log\lambda_5, \log\sigma^2)$, with b substituted in closed form, and under H_2 they are the same seven plus $w_1, \ldots, w_S$, with $\sum_s w_s$ added to the negative log marginal likelihood. Both objectives, and those of the source and cost GPs, are minimized with L-BFGS-B and analytic gradients under the bounds $\log s^2 \in [-8, 6]$, $\log\lambda_d \in [-6, 3]$, $\log\sigma^2 \in [\log 10^{-6}, 0]$, $w_s \geq 10^{-10}$, with a diagonal jitter of 10^{-8}, at most 200 iterations per start, and tolerances 10^{-8} on the objective and 10^{-5} on the projected gradient. Ten starts are used, namely the previous round's solution, a start derived from an additive ridge fit of the observed utilities on the one-hot encoding (signal variance from the fitted values, length scales from the spread of the level coefficients, noise from the residuals), the same start with all length scales divided and multiplied by four, a start with every $\lambda_d = 2$, and five Gaussian perturbations of the default start, and the best finite solution is kept. The target-side models are refitted after every target run, the cost GP is fitted whenever the acquisition uses it, and the source GPs are fitted once before the search.

C.7 Feature selection and behavior map

The factor-effect pattern of a feature on a source is its ridge regression on the one-hot encoding over the 243 runs of the source, with penalty one on the fifteen indicator coefficients and an unpenalized intercept, evaluated on all of $\mathcal{X}$. The stability of a feature is the smallest Pearson correlation between its patterns over any two sources, and with one source the landscape is split into two halves by one fixed random permutation and the stability is the correlation between the patterns fitted on the halves. The three most stable features are selected, ties broken by feature order, and f_{src} is the same ridge regression of the selected features, standardized by their mean and standard deviation over the pooled source runs, on the one-hot encoding over all source runs. The number of features, the penalty, the rule for τ^2, the refit schedule, the starts, and the bounds were fixed before the assessment seeds were run and are shared by every method that uses the corresponding component.

D Evaluation protocol and additional results

D.1 Protocol and baselines

Scenarios. With the aliases A1, A2, A3 for the Spooky landscapes and B, C, D for Ventilator, Adversarial Splay, and Scaling Law, the other-task scenarios take each landscape as the target with every nonempty subset of the other tasks as sources, one Spooky repetition serving as the source for B, C, and D (A1, A2, and A3 respectively), which gives 18 scenarios with one source, 18 with two, and 6 with three. The 6 mixed scenarios add one other Spooky repetition to the three other tasks, and the 9 same-task scenarios take one or both other repetitions as sources.

Initial design and budgets. A seed fixes the initial design of nine distinct configurations, drawn by rejection sampling so that every level of every factor appears exactly three times, and the order in which random search continues. The dollar budget of a target is the median total cost of 60 random evaluations over 10,000 such draws, namely \$126 for Ventilator, \$179, \$185, and \$186 for the Spooky repetitions, \$197 for Scaling Law, and \$259 for Adversarial Splay, and the budgets of Appendix D.2 are 0.75 and 1.25 times these values. Every policy runs until it has both made 60 evaluations and

spent 1.25 times the budget, and one stored path serves all six checkpoints, except for cost cooling, whose exponent depends on the budget and which is run once per dollar budget.

Score. For a path with utilities $u_1, u_2, \ldots$, run costs $c_1, c_2, \ldots$, and cumulative spending $s_i = \sum_{k \le i} c_k$ on a target with utility range R and maximum $u^\star$, the regret after i runs is $r_i = \big(u^\star - \max_{k \le i} u_k\big)/R$. The regret area at the count budget N is $\frac{1}{N-9} \sum_{i=9}^{N-1} r_i$, and at the dollar budget C it is $\frac{1}{C - s_9} \sum_{i \ge 9} r_i \big(\min(s_{i+1}, C) - s_i \big)_+$, the mean regret over the money spent between the end of the initial design and the budget. Scores are averaged over the scenarios of a setting within each task and then over tasks with equal weight, separately for each seed. The reduction of a baseline by ALT is the difference of the two seed-averaged scores in percent of the baseline's score, and its interval is the 2.5th and 97.5th percentile of the same ratio over 5,000 bootstrap resamples of the 100 seeds, drawn jointly for both methods.

Utility tracks. Across the landscapes, runtime spans four to seven doublings and cost five to eight, so at the balanced weights of Section 3 each resource term spans 0.09 to 0.16 utility units with a standard deviation of 0.01 to 0.04 over the 243 configurations, against 0.04 to 0.20 for performance, except runtime on Scaling Law, whose 0.75 doublings give a span of 0.015 and a standard deviation of 0.003. Each resource term therefore moves utility by a fraction of the performance spread without dominating it, the performance-focused track halves the price of a doubling, and the time- and cost-focused tracks raise one price to five percent, where that term's spread matches the performance spread on Spooky.

Baselines. GP-EI is a GP on the target data alone with kernel k_X, the fitting procedure of Appendix C.6, and expected improvement. ScaML-EI is H_2 alone, with the source weights fitted on the target as in Appendix C.3. RGPE-EI follows Feurer et al. (2022). The target GP and each source posterior are scored by the loss of the pairwise ordering of the observed target utilities, estimated from 128 posterior samples, with the target model evaluated by leave-one-out prediction. The weight of a model is the fraction of samples in which it has the lowest loss, ties going to the target model, and a source whose median loss exceeds the 95th percentile of the target model's loss is discarded. The ensemble mean is the weighted mean and its variance the weighted sum of variances with squared weights. The EI-per-dollar variants divide by the predicted cost of Appendix C.5 with exponent one. Random search follows the fixed order of the seed. Among the ablations, cost cooling uses the exponent of Lee et al. (2020), which decreases linearly from one at the end of the initial design to zero when the dollar budget is exhausted.

D.2 Dollar budgets

Table 10 repeats Table 2 with the 95 percent paired bootstrap intervals, and Tables 11 and 12 repeat it at 0.75 and 1.25 times the dollar budget. Under task shift ALT is never significantly worse than any baseline at any of the three budgets.

D.3 Evaluation counts

Table 13 gives the reductions at the count budget of 60 evaluations, including the ablations, and Fig. 4 shows them for the six baselines at 30, 45 and 60 evaluations. Under task shift ALT is never significantly worse than any baseline at any count.

D.4 Hypothesis weights and cost exponents

Table 14 gives the averages of the weight of the behavior hypothesis and of the adaptive exponent quoted in Section 6, for the choice of evaluation 10, the first after the initial design, and of evaluation 60.

Table 10: Reduction of the regret area by ALT at the dollar budget, in percent of the area of the method in the row, with the 95 percent paired bootstrap interval over 100 seeds below (bounds rounded to integers). Bold marks reductions whose interval lies above zero.

	Other-task sources				Mixed sources				Same-task sources			
Method	Bal.	Perf.	Time	Cost	Bal.	Perf.	Time	Cost	Bal.	Perf.	Time	Cost
ALT (ours)	0	0	0	0	0	0	0	0	0	0	0	0
Random search	**28.0** [24, 32]	**20.7** [17, 25]	**28.8** [24, 33]	**38.8** [36, 41]	**51.8** [48, 55]	**62.9** [59, 67]	**58.6** [55, 62]	**91.4** [90, 93]	**53.9** [50, 58]	**63.6** [60, 67]	**72.6** [70, 75]	**89.1** [87, 91]
GP-EI	3.9 [-2, 8]	**8.3** [3, 13]	-6.3 [-14, 0]	**8.3** [4, 12]	**26.4** [20, 32]	**42.0** [36, 47]	**27.6** [21, 34]	**85.0** [82, 88]	**29.6** [24, 35]	**43.0** [38, 48]	**52.1** [47, 56]	**80.9** [78, 84]
GP-EI/cost	**14.7** [11, 18]	**13.3** [10, 17]	**7.2** [2, 12]	**9.3** [6, 12]	**23.8** [19, 29]	**42.9** [37, 48]	-25.2 [-36, -15]	**73.8** [68, 79]	**27.1** [22, 32]	**43.9** [39, 48]	**17.0** [10, 23]	**66.7** [61, 72]
RGPE-EI	2.4 [-1, 6]	**6.5** [3, 10]	-0.8 [-5, 3]	**8.8** [6, 11]	3.0 [-5, 10]	0.6 [-12, 11]	**19.4** [12, 26]	**77.9** [74, 82]	**22.4** [16, 28]	0.6 [-10, 10]	**39.4** [34, 44]	**70.7** [65, 75]
RGPE-EI/cost	**13.5** [10, 17]	**10.8** [8, 14]	**6.3** [2, 10]	**7.1** [5, 9]	0.9 [-8, 9]	3.3 [-9, 14]	-25.4 [-37, -15]	**55.6** [46, 64]	**14.2** [7, 21]	7.8 [-2, 16]	6.5 [-2, 13]	**41.4** [30, 51]
ScaML-EI	**4.4** [2, 7]	**11.0** [8, 14]	**11.5** [8, 14]	**5.3** [3, 7]	-1.6 [-8, 4]	1.0 [-4, 6]	**16.4** [10, 22]	**63.5** [56, 70]	3.3 [-2, 8]	-4.2 [-11, 2]	-11.8 [-22, -3]	**60.4** [54, 66]
ScaML-EI/cost	**10.4** [7, 13]	**14.5** [12, 17]	**9.0** [6, 12]	**5.8** [4, 8]	2.0 [-3, 7]	**13.0** [5, 20]	-23.8 [-32, -16]	-33.3 [-64, -8]	2.1 [-4, 7]	**11.4** [4, 18]	-21.1 [-31, -13]	-45.0 [-72, -24]
Ablations of ALT												
without H_2 (BEH)	-3.5 [-6, -1]	-2.5 [-5, 0]	-15.4 [-20, -11]	-3.1 [-5, -1]	**31.0** [27, 35]	**36.9** [31, 43]	**25.6** [19, 32]	**72.9** [67, 78]	**30.0** [25, 34]	**33.0** [28, 38]	**32.6** [27, 38]	**61.1** [55, 66]
equal mixture	-2.8 [-5, 0]	-0.7 [-3, 2]	-12.3 [-16, -9]	-0.6 [-3, 1]	**15.2** [9, 21]	**22.5** [15, 30]	**11.7** [4, 19]	**49.2** [39, 58]	**14.7** [9, 20]	**16.9** [11, 23]	**9.1** [2, 14]	**12.7** [1, 24]
EI ($\alpha = 0$)	-1.0 [-3, 1]	1.8 [0, 4]	**2.8** [0, 5]	**3.0** [1, 5]	-2.6 [-8, 2]	-1.4 [-5, 2]	**16.4** [11, 21]	**64.6** [58, 71]	2.9 [-2, 8]	-2.3 [-5, 1]	**6.9** [2, 12]	**58.7** [52, 65]
EI/cost ($\alpha = 1$)	**7.8** [5, 10]	**4.4** [2, 7]	**5.7** [3, 8]	**2.2** [1, 4]	1.6 [-4, 7]	**9.0** [1, 17]	-21.6 [-29, -14]	-25.9 [-54, -4]	**6.4** [2, 11]	**9.5** [2, 16]	-11.2 [-18, -6]	-46.1 [-68, -29]
cost cooling	**3.5** [1, 6]	2.1 [0, 5]	**2.8** [0, 5]	1.4 [0, 3]	4.2 [-1, 10]	**10.0** [3, 17]	-17.9 [-25, -11]	-21.6 [-48, -1]	**7.9** [3, 13]	**12.9** [6, 19]	-6.6 [-14, -1]	-22.6 [-42, -7]
Regret area of ALT (%)	6.0	8.0	6.3	5.1	2.9	2.6	2.2	0.5	2.8	2.5	1.5	0.7

Table 11: Reduction of the regret area by ALT at 0.75 times the dollar budget, in the format of Table 10.

	Other-task sources				Mixed sources				Same-task sources			
Method	Bal.	Perf.	Time	Cost	Bal.	Perf.	Time	Cost	Bal.	Perf.	Time	Cost
ALT (ours)	0	0	0	0	0	0	0	0	0	0	0	0
Random search	**25.8** [22, 29]	**15.8** [12, 20]	**26.5** [22, 31]	**38.8** [36, 41]	**47.3** [44, 51]	**56.9** [53, 61]	**55.1** [51, 59]	**89.6** [88, 91]	**48.7** [45, 52]	**57.5** [54, 61]	**70.3** [68, 73]	**87.0** [85, 89]
GP-EI	**4.6** [0, 8]	**5.6** [1, 10]	-5.3 [-12, 0]	**10.2** [7, 13]	**23.8** [18, 29]	**37.5** [32, 43]	**28.1** [21, 35]	**83.5** [80, 86]	**25.9** [20, 31]	**38.4** [34, 43]	**52.5** [48, 57]	**79.2** [76, 82]
GP-EI/cost	**8.2** [4, 12]	**8.1** [5, 11]	-2.0 [-7, 2]	**5.6** [3, 8]	**20.0** [15, 25]	**38.7** [34, 44]	-26.5 [-39, -16]	**72.5** [66, 78]	**22.2** [17, 27]	**39.5** [35, 44]	**16.3** [8, 23]	**65.5** [59, 71]
RGPE-EI	3.1 [0, 6]	**4.9** [1, 8]	0.0 [-4, 4]	**10.2** [8, 13]	4.6 [-3, 12]	-1.1 [-12, 9]	**19.7** [13, 26]	**75.3** [71, 79]	**19.2** [14, 25]	-3.1 [-13, 6]	**39.7** [35, 44]	**68.1** [63, 73]
RGPE-EI/cost	**7.3** [4, 10]	**6.9** [4, 10]	-1.5 [-6, 2]	**3.7** [2, 6]	0.9 [-7, 9]	2.8 [-8, 13]	-25.6 [-37, -14]	**54.0** [45, 62]	**12.5** [6, 19]	4.7 [-4, 12]	6.4 [-2, 13]	**40.4** [29, 50]
ScaML-EI	**4.8** [2, 7]	**9.2** [7, 11]	**11.6** [9, 14]	**4.4** [3, 6]	-1.5 [-7, 4]	-0.4 [-4, 4]	**16.2** [10, 22]	**59.0** [52, 66]	1.7 [-3, 6]	-5.3 [-11, 0]	-8.9 [-18, -1]	**56.9** [50, 63]
ScaML-EI/cost	**6.7** [4, 9]	**10.6** [8, 13]	**5.2** [2, 8]	1.6 [0, 3]	1.1 [-4, 6]	**9.8** [3, 16]	-25.2 [-34, -17]	-31.1 [-62, -7]	-0.3 [-6, 5]	**9.1** [2, 15]	-19.7 [-28, -12]	-40.7 [-63, -21]
Ablations of ALT												
without H_2 (BEH)	-3.1 [-5, -1]	-2.4 [-5, 0]	-14.3 [-18, -11]	-1.3 [-3, 0]	**29.4** [25, 33]	**33.7** [28, 39]	**25.4** [19, 32]	**72.5** [67, 77]	**27.8** [23, 32]	**30.3** [25, 35]	**32.4** [27, 37]	**60.5** [55, 66]
equal mixture	-2.5 [-4, -1]	-1.4 [-3, 1]	-12.0 [-15, -9]	-0.4 [-2, 1]	**15.3** [10, 21]	**21.4** [15, 28]	**11.1** [4, 18]	**48.0** [38, 56]	**12.9** [8, 18]	**14.8** [9, 20]	5.8 [-1, 11]	**14.2** [4, 24]
EI ($\alpha = 0$)	0.0 [-2, 2]	1.1 [-1, 3]	**4.2** [2, 6]	**4.0** [3, 6]	-2.0 [-6, 2]	-1.2 [-4, 2]	**16.3** [11, 21]	**60.4** [53, 67]	2.6 [-2, 7]	-2.6 [-5, 0]	**8.7** [4, 13]	**55.8** [49, 62]
EI/cost ($\alpha = 1$)	**4.7** [3, 7]	**2.7** [0, 5]	1.9 [0, 4]	-0.5 [-2, 1]	0.8 [-4, 6]	**7.4** [1, 14]	-23.7 [-32, -16]	-24.7 [-52, -4]	**4.7** [0, 9]	**8.6** [2, 15]	-12.9 [-20, -7]	-40.4 [-58, -26]
cost cooling	**2.6** [1, 4]	2.3 [0, 4]	1.9 [0, 4]	-0.1 [-1, 1]	2.2 [-3, 7]	5.9 [-1, 12]	-17.9 [-26, -11]	-18.8 [-45, 1]	**4.9** [0, 9]	**11.1** [5, 17]	-7.6 [-15, -2]	-19.2 [-35, -6]
Regret area of ALT (%)	7.1	9.4	7.5	5.7	3.5	3.3	2.8	0.7	3.4	3.2	1.9	0.9

Table 12: Reduction of the regret area by ALT at 1.25 times the dollar budget, in the format of Table 10.

	Other-task sources				Mixed sources				Same-task sources			
Method	Bal.	Perf.	Time	Cost	Bal.	Perf.	Time	Cost	Bal.	Perf.	Time	Cost
ALT (ours)	0	0	0	0	0	0	0	0	0	0	0	0
Random search	**30.1** [26, 34]	**24.6** [20, 29]	**31.0** [26, 35]	**38.8** [36, 42]	**54.3** [51, 58]	**66.9** [63, 70]	**61.0** [57, 64]	**92.6** [91, 94]	**57.3** [53, 61]	**67.6** [65, 70]	**74.0** [72, 76]	**90.4** [88, 92]
GP-EI	3.9 [-2, 9]	**10.6** [5, 16]	-6.6 [-14, 0]	**6.1** [1, 10]	**27.7** [22, 33]	**44.5** [38, 50]	**28.2** [21, 35]	**85.5** [83, 88]	**32.4** [27, 38]	**45.8** [40, 51]	**52.1** [48, 56]	**81.1** [78, 84]
GP-EI/cost	**19.8** [15, 24]	**17.2** [13, 21]	**14.3** [9, 19]	**12.3** [9, 15]	**27.0** [22, 32]	**45.2** [39, 51]	-22.5 [-34, -13]	**74.0** [68, 79]	**31.8** [27, 37]	**46.4** [42, 51]	**18.3** [11, 25]	**66.0** [59, 72]
RGPE-EI	2.2 [-2, 6]	**7.5** [3, 11]	-0.8 [-6, 4]	**6.7** [3, 10]	1.8 [-7, 10]	0.3 [-13, 12]	**19.6** [13, 26]	**78.9** [75, 82]	**25.3** [19, 31]	3.2 [-9, 13]	**39.4** [34, 44]	**71.0** [65, 76]
RGPE-EI/cost	**18.1** [15, 21]	**13.2** [10, 17]	**12.0** [7, 16]	**10.0** [8, 12]	1.8 [-7, 10]	3.8 [-9, 15]	-25.1 [-37, -14]	**56.1** [47, 64]	**17.1** [10, 24]	10.0 [0, 19]	6.1 [-1, 13]	**40.2** [28, 50]
ScaML-EI	**4.2** [1, 7]	**12.2** [9, 15]	**11.2** [8, 14]	**6.1** [4, 8]	-1.5 [-8, 5]	2.0 [-4, 8]	**16.4** [10, 22]	**65.1** [58, 71]	4.9 [-1, 11]	-3.3 [-11, 4]	-13.5 [-24, -5]	**61.5** [54, 68]
ScaML-EI/cost	**12.6** [9, 16]	**16.9** [14, 19]	**11.6** [8, 15]	**9.5** [7, 11]	1.9 [-4, 8]	**15.0** [7, 23]	-22.2 [-30, -15]	-34.3 [-66, -9]	4.2 [-2, 10]	**12.2** [4, 19]	-21.7 [-31, -14]	-49.1 [-79, -26]
Ablations of ALT												
without H_2 (BEH)	-3.4 [-6, -1]	-2.4 [-5, 0]	-16.1 [-21, -12]	-5.3 [-8, -3]	**31.5** [27, 36]	**38.6** [32, 45]	**26.2** [20, 32]	**73.1** [67, 78]	**31.4** [27, 36]	**34.1** [28, 39]	**32.3** [26, 38]	**60.2** [54, 66]
equal mixture	-2.3 [-5, 1]	-0.4 [-3, 2]	-12.3 [-17, -8]	-1.3 [-3, 1]	**15.2** [8, 21]	**23.3** [15, 31]	**12.7** [5, 19]	**49.3** [40, 58]	**16.3** [11, 22]	**18.2** [11, 25]	**12.5** [6, 18]	10.0 [-3, 21]
EI ($\alpha = 0$)	-1.5 [-4, 1]	**2.5** [0, 5]	1.6 [-1, 4]	2.1 [0, 4]	-3.1 [-9, 2]	-1.3 [-6, 3]	**16.3** [11, 21]	**66.2** [59, 72]	3.3 [-2, 9]	-2.1 [-6, 1]	**5.8** [0, 11]	**59.5** [52, 66]
EI/cost ($\alpha = 1$)	**10.1** [8, 13]	**5.1** [2, 8]	**8.3** [6, 11]	**4.5** [3, 6]	1.8 [-5, 8]	**10.1** [1, 19]	-19.4 [-27, -12]	-26.8 [-56, -4]	**8.1** [3, 13]	**9.6** [2, 17]	-8.9 [-15, -3]	-51.2 [-77, -31]
cost cooling	**4.6** [2, 7]	2.7 [0, 5]	**4.8** [2, 7]	**2.2** [0, 4]	3.9 [-2, 10]	**10.3** [2, 19]	-14.9 [-22, -8]	-24.7 [-53, -1]	**7.8** [2, 13]	**12.6** [5, 20]	-6.0 [-13, 0]	-25.4 [-50, -7]
Regret area of ALT (%)	5.3	6.9	5.4	4.6	2.5	2.1	1.9	0.4	2.4	2.1	1.2	0.6

Table 13: Reduction of the regret area by ALT at the count budget of 60 evaluations, in the format of Table 10.

	Other-task sources				Mixed sources				Same-task sources			
Method	Bal.	Perf.	Time	Cost	Bal.	Perf.	Time	Cost	Bal.	Perf.	Time	Cost
ALT (ours)	0	0	0	0	0	0	0	0	0	0	0	0
Random search	**27.1** [22, 32]	**25.0** [21, 29]	**30.0** [25, 34]	**29.0** [26, 32]	**48.6** [45, 52]	**74.1** [71, 77]	**45.0** [41, 48]	**70.1** [67, 73]	**53.1** [49, 57]	**74.5** [72, 77]	**55.1** [52, 58]	**70.0** [67, 73]
GP-EI	1.2 [-5, 7]	**8.0** [2, 13]	-2.5 [-10, 4]	2.7 [-2, 7]	**17.4** [11, 23]	**50.3** [44, 56]	**17.9** [12, 23]	**58.6** [55, 62]	**24.7** [19, 30]	**51.1** [46, 56]	**32.9** [28, 37]	**58.4** [54, 62]
GP-EI/cost	**25.0** [21, 29]	**29.5** [27, 32]	**22.9** [19, 27]	**12.1** [10, 15]	**38.3** [34, 42]	**60.2** [55, 64]	**14.4** [9, 19]	**49.9** [44, 55]	**43.7** [39, 48]	**60.8** [57, 64]	**30.0** [25, 34]	**49.6** [44, 55]
RGPE-EI	-1.0 [-5, 3]	2.7 [-2, 7]	0.7 [-4, 5]	**3.9** [1, 7]	-18.6 [-29, -9]	1.4 [-14, 14]	**8.1** [2, 13]	**46.9** [42, 52]	**17.6** [11, 23]	6.1 [-6, 17]	**19.5** [14, 24]	**44.7** [39, 50]
RGPE-EI/cost	**23.0** [20, 26]	**25.0** [22, 28]	**20.9** [17, 25]	**10.9** [9, 13]	**15.4** [8, 23]	**19.7** [7, 30]	**9.0** [3, 14]	**32.8** [26, 40]	**33.9** [28, 39]	**26.4** [17, 35]	**19.4** [15, 24]	**29.8** [22, 37]
ScaML-EI	-1.0 [-5, 2]	**4.9** [2, 8]	**4.0** [0, 8]	**3.4** [1, 6]	-6.0 [-13, 0]	1.0 [-6, 8]	0.9 [-3, 5]	**30.4** [24, 37]	-3.4 [-10, 2]	-3.6 [-13, 5]	-15.3 [-21, -10]	**34.6** [28, 40]
ScaML-EI/cost	**18.5** [16, 21]	**20.1** [18, 23]	**17.3** [14, 21]	**9.2** [7, 11]	**14.3** [10, 19]	**24.0** [16, 32]	-3.0 [-8, 2]	-7.2 [-17, 2]	**15.1** [10, 20]	**24.0** [16, 31]	-2.0 [-6, 2]	-9.7 [-18, -2]
Ablations of ALT												
without H_2 (BEH)	-4.7 [-8, -1]	-1.3 [-4, 1]	-10.7 [-15, -6]	-6.8 [-9, -4]	**20.2** [15, 25]	**38.3** [31, 45]	**22.3** [17, 27]	**47.2** [41, 53]	**25.5** [20, 30]	**30.3** [23, 37]	**24.0** [20, 28]	**41.6** [37, 46]
equal mixture	-3.4 [-7, 0]	0.5 [-2, 3]	-8.1 [-12, -4]	-2.1 [-4, 0]	5.8 [-1, 12]	**23.6** [14, 32]	**10.8** [6, 15]	**22.7** [15, 29]	**12.7** [7, 18]	**15.5** [7, 23]	**4.0** [0, 7]	**9.0** [2, 15]
EI ($\alpha = 0$)	-6.9 [-10, -4]	-4.7 [-7, -2]	-4.4 [-7, -1]	-0.4 [-2, 1]	-8.0 [-13, -3]	-2.3 [-7, 3]	1.1 [-2, 5]	**31.9** [25, 38]	-6.5 [-12, -1]	-3.4 [-8, 1]	-6.9 [-12, -2]	**34.0** [28, 39]
EI/cost ($\alpha = 1$)	**16.3** [14, 19]	**14.8** [12, 17]	**15.5** [13, 18]	**6.4** [5, 8]	**14.5** [10, 19]	**20.3** [11, 29]	-1.6 [-6, 3]	-6.6 [-15, 1]	**18.6** [14, 23]	**21.0** [13, 28]	1.0 [-2, 4]	-10.8 [-17, -5]
cost cooling	**12.2** [10, 15]	**10.6** [8, 13]	**12.6** [10, 15]	**5.0** [4, 7]	**14.5** [10, 19]	**18.3** [9, 27]	-1.8 [-6, 2]	-7.0 [-16, 1]	**16.4** [12, 21]	**21.3** [14, 28]	1.6 [-2, 5]	-8.0 [-15, -2]
Regret area of ALT (%)	6.0	7.5	6.1	5.9	3.1	1.8	3.0	1.9	2.9	1.8	2.5	1.9

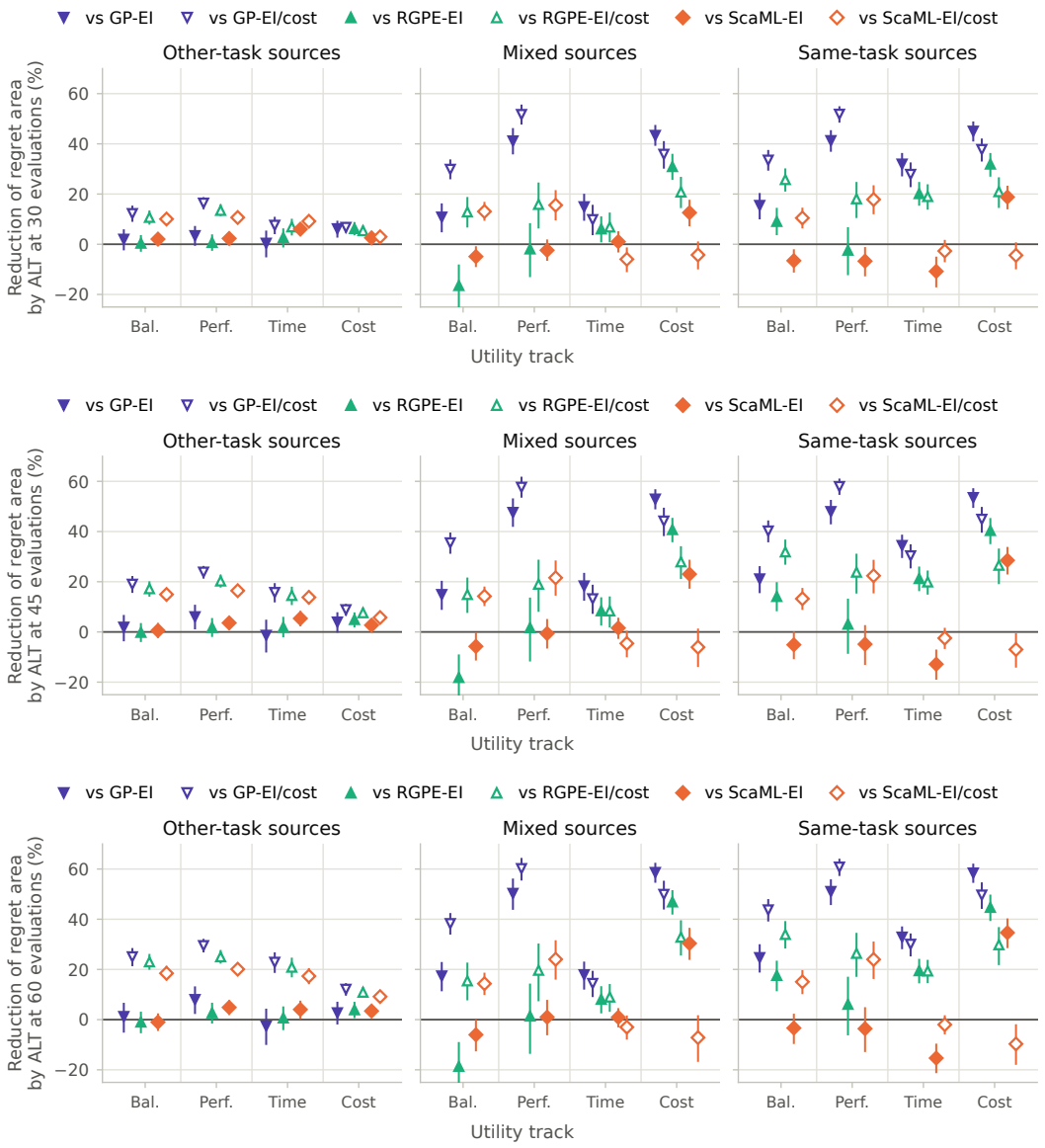

Figure 4: Reduction of the regret area by ALT at 30 (top), 45 (middle) and 60 (bottom) evaluations, in percent of the area of each baseline, with 95 percent paired intervals over 100 seeds. Panels are the source settings, positions the utility tracks, filled markers the EI baselines and open markers their EI-per-dollar variants. ALT reduces the area of random search by 14 to 75 percent in every cell, so random search is left out.

Table 14: Weight of the behavior hypothesis π_n (by setting and number of sources, averaged over tracks, tasks, and seeds) and adaptive exponent α_n (by setting and utility track, averaged over source counts, tasks, and seeds) of ALT, as fitted for the choice of evaluation 10 and of evaluation 60, that is, from 9 and 59 observed target runs.

	Other-task sources			Mixed	Same-task sources	
	1 source	2 sources	3 sources	4 sources	1 source	2 sources
π_n for evaluation 10	0.54	0.29	0.15	0.01	0.16	0.10
π_n for evaluation 60	0.54	0.30	0.11	< 0.01	0.05	0.02

	Bal.	Perf.	Time	Cost
α_n for evaluation 10, other-task sources	0.41	0.30	0.46	0.57
α_n for evaluation 60, other-task sources	0.38	0.23	0.43	0.56
α_n for evaluation 10, mixed sources	0.21	0.05	0.47	0.55
α_n for evaluation 60, mixed sources	0.22	0.02	0.63	0.71
α_n for evaluation 10, same-task sources	0.33	0.11	0.66	0.71
α_n for evaluation 60, same-task sources	0.35	0.02	0.86	0.93