BAYESIANAGENT: GUIDING AGENTIC EVOLUTION WITH BAYESIAN OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Agents that iteratively improve executable solutions must allocate a limited experimental budget across a growing set of candidates. Most existing systems use rule-based selection informed by observed performance and experimental history. However, these rules do not explicitly learn to predict the gain from the next expansion and quantify uncertainty in that prediction. We introduce BayesianAgent, a Bayesian optimization approach that learns where to invest further experiments from completed outcomes. Its central intuition is that an experiment can inform the improvement potential of related candidates, including those with little direct evidence. A Gaussian process learns expansion gains and their uncertainty, sharing evidence across candidates. Joint Thompson sampling then selects the next candidate to expand, balancing promising refinements with opportunities whose value remains uncertain. We compare all methods using Muse Spark 1.3 under matched experimental budgets. Across 15 AlphaEvolve mathematical optimization tasks and 22 MLE-bench Lite tasks, BayesianAgent outperforms the MLEvolve baseline. It achieves the best AlphaEvolve average rank (1.83), the highest MLE-bench Lite gold rate (45.45%), and a joint-highest medal rate (72.73%) among evaluated methods. Further analysis shows that its predictions rank observed expansion gains more accurately than MLEvolve's historical rewards. These results support learning from experimental outcomes to guide subsequent allocation in agentic evolution.

1 INTRODUCTION

Language-model agents increasingly discover and improve algorithms through repeated code generation and execution. Systems such as AlphaEvolve (Novikov et al., 2025) and MLEvolve (Du et al., 2026) maintain candidate solutions, evaluate their performance, and use the results to guide subsequent modifications. Each expansion generates and evaluates a modified child from an existing parent, consuming resources that could instead support another candidate. Under a limited experimental budget, the agent must therefore decide which existing solution to expand next. This allocation decision determines which opportunities receive further evaluation as evolution proceeds.

In practice, most existing systems allocate experiments through rule-based selection informed by observed performance and experimental history (Jiang et al., 2025; Novikov et al., 2025; Liu et al., 2025; Du et al., 2026). However, these rules do not explicitly learn to predict the gain from the next expansion and quantify uncertainty in that prediction. A strong candidate may offer little further improvement, whereas a weaker candidate may produce a better child after refinement. This motivates learning expansion gains from completed experiments and combining those predictions with parent performance to guide subsequent allocation.

Nevertheless, learning expansion outcomes is difficult because experimental feedback is sparse and the candidate set continually grows. Newly generated solutions often have no expansion history, even when related candidates have already been evaluated. Estimating each candidate independently would leave these observations disconnected from many subsequent decisions. A shared predictive model can use similarities in measured performance, expansion history, and ancestry to inform related candidates. Because these predictions rely on limited evidence, their uncertainty must also be represented when choosing the next expansion. We therefore ask: *How can an evolving agent learn from sparse experimental feedback to predict expansion outcomes and choose its next experiment?*

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.

We introduce **BayesianAgent**, which allocates experiments by learning which candidates are likely to produce better solutions after further refinement. The central idea is to let each completed experiment inform where to allocate the next experiment, including candidates with little direct evidence. A Gaussian process combines experimental feedback and LLM assessments to learn expansion gains and uncertainty, sharing evidence across candidates. Joint Thompson sampling (Russo et al., 2018) compares plausible expansion outcomes to choose the next parent, balancing promising refinements with opportunities whose value remains uncertain. Each new result updates these predictions to guide subsequent experiments. Our contributions are as follows:

- We formulate parent selection through a joint posterior over expected expansion outcomes, linking predicted gains to measured parent performance.
- We develop a shared Gaussian-process surrogate that conditions on experimental feedback using parent performance, expansion history, ancestry, and language-model assessments.
- On 15 AlphaEvolve tasks, BayesianAgent achieves the best average rank (1.83). Across all 22 MLE-bench Lite tasks, it attains the highest gold rate (45.45%) and a joint-highest medal rate (72.73%) among evaluated methods. Both rates surpass those of the previous state-of-the-art baseline, MLEvolve (Du et al., 2026), under matched budgets.

2 Related Work

Agentic Evolution and Solution Selection Agentic evolution improves candidate solutions through repeated generation and evaluation (Zhang et al., 2026). FunSearch (Romera-Paredes et al., 2024) and AlphaEvolve (Novikov et al., 2025) can evaluate large numbers of functions or algorithms against programmatic scorers. Machine-learning engineering agents (Jiang et al., 2025; Liu et al., 2025; Toledo et al., 2025; Li et al., 2026) evaluate computationally expensive training pipelines, limiting the number of candidates tested on benchmarks such as MLE-bench (Chan et al., 2025).

Most existing systems use rule-based selection informed by observed performance and search history. Examples include selecting the best observed solution (Jiang et al., 2025), adding visitation bonuses (Liu et al., 2025), and sampling according to performance (Lange et al., 2026; Toledo et al., 2025). AlphaEvolve (Novikov et al., 2025) maintains diversity through an evolutionary database inspired by MAP-Elites and island-based populations. MLEvolve (Du et al., 2026) organizes solutions in a graph and uses progressive Monte Carlo graph search to shift experimental effort toward elite candidates over time. These rules use feedback to guide exploration and exploitation, but do not explicitly learn a predictive distribution over the gain from the next expansion.

Language-model assessments can also identify promising candidates for further exploration. I-MCTS (Liang et al., 2026) uses an LLM to evaluate solutions before costly rollouts, then progressively shifts its hybrid reward toward measured performance. BayesianAgent combines both sources of evidence: observed performance and expansion history, together with LLM assessments of remaining improvement potential. From these inputs, it learns a predictive distribution over expansion gains to guide selection under uncertainty.

Bayesian Optimization Gaussian-process Bayesian optimization (Rasmussen & Williams, 2006) models expensive evaluations and selects the next query using an acquisition function, such as expected improvement (Jones et al., 1998) or an upper confidence bound (Srinivas et al., 2010). Bayesian optimization has long allocated machine learning experiments over fixed configuration spaces (Snoek et al., 2012) and extends to structured domains such as neural architectures (Ru et al., 2021).

Recent efforts have explored combining large language models with Bayesian optimization. LLAMBO (Liu et al., 2024) uses an LLM as surrogate and sampler, while BOPRO (Agarwal et al., 2025) uses Bayesian optimization to guide solution generation by an LLM. We apply Bayesian optimization to parent selection over an evolving solution graph, learning expansion gains from completed experiments. The model shares evidence across related candidates and guides allocation through joint Thompson sampling.

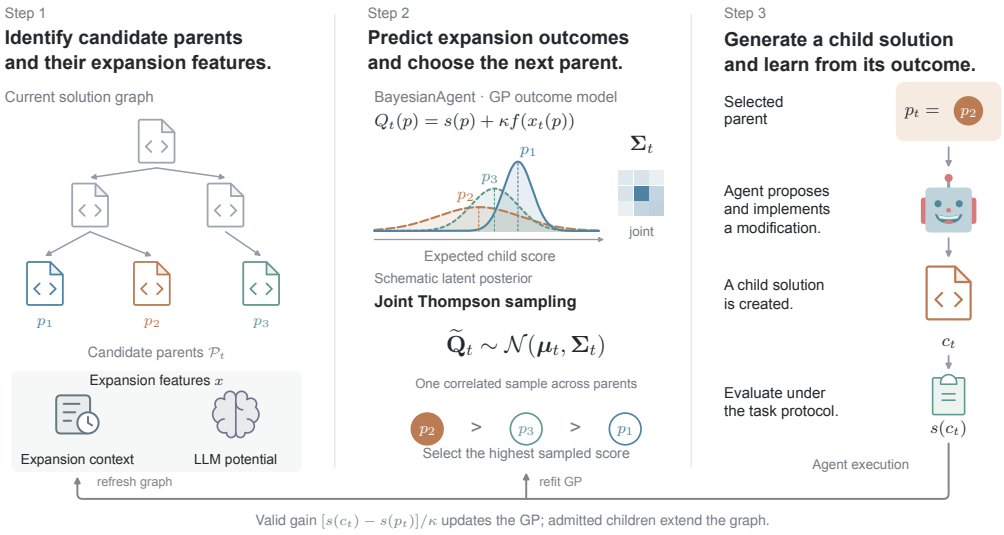

Figure 1: **Allocation and feedback in BayesianAgent.** At each step, the agent must decide which existing solution to refine next. A Gaussian process learns expansion gains from completed outcomes, using expansion context and LLM potential as inputs. Adding parent scores gives a joint posterior over expected child scores, which Thompson sampling uses to select an expansion. Evaluating the generated child provides feedback for the next decision.

3 METHOD

An AI agent iteratively generates, evaluates, and refines candidate solutions to improve performance under a limited experimental budget. As these candidates accumulate into a solution graph, the agent must decide which node to expand next. At decision time t, this graph is $G_t = (V_t, E_t)$, where V_t contains evaluated solutions and E_t records their derivation relations. An evaluation function $s(\cdot)$ assigns each solution a score, with higher values indicating better performance. For minimization tasks, s is the negative of the evaluation metric. Each parent in the eligible set $\mathcal{P}_t \subseteq V_t$ has a proposed operation for generating a child. The agent selects one of these expansions for the next experiment.

The parent's measured score describes its current quality, but leaves the gain from another expansion unknown. BayesianAgent learns this gain from completed experiments, allowing an observed outcome to inform predictions for other, related expansions. It combines these predictions with parent scores to form a joint posterior over expected child quality, then uses Thompson sampling to choose an expansion. Figure 1 summarizes the allocation and feedback loop.

3.1 GAUSSIAN PROCESS MODELING OF EXPANSION OUTCOMES

During agentic evolution, each experiment reveals the outcome of one chosen expansion, leaving most current candidates with little direct evidence of their improvement potential. However, candidates with similar refinement histories or a shared initial solution may respond similarly to further changes. We therefore use a Gaussian process (GP) (Rasmussen & Williams, 2006) to learn expansion gains jointly, sharing observations through a covariance model. Its posterior describes both predicted gains and uncertainty about these predictions.

We model improvement over the parent so that each observation measures what the expansion added to an already evaluated solution. For the ith valid expansion, let p_i and c_i denote its parent and child. To keep gains on a consistent scale throughout evolution, we fix $\kappa > 0$ using the initial solutions. We define the observed normalized gain as

$$y_i = \frac{s(c_i) - s(p_i)}{\kappa}. \tag{1}$$

Valid expansions produce children that satisfy the task's evaluation requirements and yield a finite score. Negative gains are retained because they reveal refinements that reduced solution quality.

The input x_i describes parent p_i and its proposed expansion before child c_i is generated. It combines two sources available at selection: expansion context and LLM potential. The expansion context includes the parent's performance, its gain over its own parent, and its root, the solution that starts its branch. It also records the proposed operation and the gains and attempt counts from previous expansions of that parent using the same operation. LLM potential complements this history with an LLM-generated assessment of remaining improvement potential, based on the parent's solution and evaluation evidence. The assessor rates remaining improvement potential as low, moderate, or high, encoded as $r \in \{-1, 0, 1\}$; a missing assessment is assigned the neutral value zero. Appendix A.1 details the features and rating criteria.

Given these inputs, we model the expected normalized gain conditional on a valid expansion as a latent function $f(x)$. The observed gain y_i may differ from $f(x_i)$; we represent this deviation by an observation error ϵ_i:

$$y_i = f(x_i) + \epsilon_i, \qquad f \sim \mathcal{GP}(\mu_0, k), \qquad \epsilon_i \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma_n^2). \tag{2}$$

Here $\mu_0(x)$ is the prior mean, whose construction from LLM potential is explained below. The kernel $k(x, x')$ is the prior covariance between expected gains at inputs x and x'. The observation errors ϵ_i have variance σ_n^2 and are mutually independent and independent of f; the latent gains remain correlated through k.

The LLM rating indicates improvement potential, but not the size of the expected gain. We model its contribution as θr, where the coefficient θ is shared across candidates and learned from observed gains. To represent our initial uncertainty, we assign $\theta \sim \mathcal{N}(\mu_\theta, \sigma_\theta^2)$, with prior mean μ_θ and variance σ_θ^2. We write $f(x) = \theta r + g(x)$, where $g(x)$ captures variation in expected gains beyond the LLM rating. The residual g has a zero-mean GP prior with covariance $k_{\text{context}} + k_{\text{root}}$, independent of θ. These components describe similar expansion contexts and descendants of a shared root, respectively.[1] Integrating out θ gives the mean and covariance

$$\mu_0(x) = \mu_\theta r, \tag{3}$$

$$k(x, x') = k_{\text{context}}(x, x') + k_{\text{root}}(x, x') + \sigma_\theta^2 r r'. \tag{4}$$

Here r and r' are the potential ratings in x and x', respectively. Observed gains update the shared coefficient, changing how strongly LLM ratings influence predictions across candidates.

The *context kernel* k_{context} shares evidence across roots with similar expansion contexts and proposed operations. It uses a four-dimensional vector z encoding parent performance, gain over its own parent, historical mean expansion gain, and attempt count. The last two summarize gain and attempt histories for the same parent–operation pair. We use Matérn-$3/2$ similarity (Rasmussen & Williams, 2006) to assign greater covariance between expansion gains when these feature vectors are closer. Because similar parents can respond differently to different operations, a categorical factor gives greater weight to matching proposed operations o. For example, independently developed solutions with similar scores and little refinement history may offer comparable improvement opportunities under the same operation. The *root kernel* k_{root} captures inherited model structures or training procedures among descendants of a shared root solution. As descendants are refined, their performance can diverge; we therefore use score similarity to modulate this shared-root covariance. It adds Matérn-$3/2$ covariance in standardized parent performance $v = (s(p) - s_{\text{ref}})/\kappa$ only when inputs share the same root identifier b. Here s_{ref} is the initial-score median, and the additional covariance weakens as the descendants' performance diverges. For example, descendants retaining the same training procedure may share limitations, making one refinement's outcome informative about opportunities within that root. Appendix A.2 specifies both kernels and their parameters.

3.2 Posterior Inference

Posterior inference uses observed expansion gains to revise our expectations of how much current candidates can improve. We collect the recorded feature–gain pairs (x_i, y_i) from completed valid

[1] Following MLEvolve (Du et al., 2026), we initialize three drafts as roots, potentially representing different model families (e.g., MLPs or CNNs). Appendix A.5 provides examples of solutions sharing a root.

expansions in $\mathcal{D}_t$. There are $n = |\mathcal{D}_t|$ such observations and $m = |\mathcal{P}_t|$ current candidate parents.[2] The current input $x_t(p)$ uses the same representation as historical x_i, evaluated for parent p and its proposed expansion at time t. Given $\mathcal{D}_t$, the posterior over the candidates' expected normalized gains $f(x_t(p))$ is jointly Gaussian. Its mean vector $\boldsymbol{\mu}_t^f \in \mathbb{R}^m$ and covariance matrix $\mathbf{C}_t \in \mathbb{R}^{m\times m}$ summarize the predicted gains and their uncertainty. For any two candidate parents $p, p' \in \mathcal{P}_t$, their entries are

$$\begin{aligned}(\boldsymbol{\mu}_t^f)_p &= \mathbb{E}[f(x_t(p)) \mid \mathcal{D}_t],\\ (\mathbf{C}_t)_{p,p'} &= \mathrm{Cov}[f(x_t(p)), f(x_t(p')) \mid \mathcal{D}_t]\,.\end{aligned} \tag{5}$$

Diagonal entries of $\mathbf{C}_t$ describe each candidate's uncertainty; off-diagonal entries describe how their expected gains vary together. To make this update explicit, let X collect the n observed inputs, $\mathbf{y} \in \mathbb{R}^n$ their gains, and X_t the m candidate inputs. For input arrays U, V, define $[K(U,V)]_{ij} = k(u_i, v_j)$, where u_i, v_j are their respective rows. Write $\mu_0(U)$ for the vector of prior means at U. The matrix $A = K(X,X) + \sigma_n^2 I_n \in \mathbb{R}^{n\times n}$ is the covariance of historical observations, including noise; I_n is the identity matrix. The cross-covariance $K(X_t, X) \in \mathbb{R}^{m\times n}$ links current candidates to historical experiments, and $K(X, X_t)$ is its transpose. The matrix $K(X_t, X_t) \in \mathbb{R}^{m\times m}$ describes prior covariance among current candidates. Conditioning under the noise model in Equation 2 gives

$$\begin{aligned}\boldsymbol{\mu}_t^f &= \mu_0(X_t) + K(X_t, X)A^{-1}[\mathbf{y} - \mu_0(X)],\\ \mathbf{C}_t &= K(X_t, X_t) - K(X_t, X)A^{-1}K(X, X_t).\end{aligned} \tag{6}$$

The posterior mean adjusts prior predictions according to how much observed gains exceed or fall short of expectations, with covariance determining how this feedback affects each candidate. The covariance update accounts for uncertainty resolved by those observations, conditional on the fitted kernel parameters (Appendix A.3).

The posterior above predicts gains relative to each parent. A larger gain may still yield a worse child if the parent starts from a lower score. We therefore restore score units with κ and add the parent score $s(p)$ to define the latent expected child score $Q_t(p)$. Let $\mathbf{Q}_t$ and $\mathbf{s}_t$ collect expected child scores and measured parent scores in the same candidate order as $\boldsymbol{\mu}_t^f$ and $\mathbf{C}_t$:

$$\begin{aligned}Q_t(p) &= s(p) + \kappa f(x_t(p)), & \mathbf{Q}_t \mid \mathcal{D}_t &\sim \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t),\\ \boldsymbol{\mu}_t &= \mathbf{s}_t + \kappa\boldsymbol{\mu}_t^f\,, & \boldsymbol{\Sigma}_t &= \kappa^2\mathbf{C}_t.\end{aligned} \tag{7}$$

Here $\boldsymbol{\mu}_t \in \mathbb{R}^m$ and $\boldsymbol{\Sigma}_t \in \mathbb{R}^{m\times m}$ are the posterior mean and covariance of expected child scores. Parent scores shift only the mean; rescaling gains multiplies covariance by κ^2. We use this joint posterior to select the next parent through Thompson sampling.

3.3 Posterior-Guided Expansion Selection

Selecting only the highest posterior mean can overlook uncertain candidates that may yield better solutions. We therefore use Thompson sampling (Russo et al., 2018) to explore plausible alternatives while favoring candidates with higher predicted quality. We sample child scores jointly to preserve correlations from shared evidence, then select the parent with the highest sampled score:

$$\widetilde{\mathbf{Q}}_t \sim \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t), \qquad p_t \in \arg\max_{p\in\mathcal{P}_t} \widetilde{Q}_t(p). \tag{8}$$

To generate correlated draws, we use $\widetilde{\mathbf{Q}}_t = \boldsymbol{\mu}_t + L_t\boldsymbol{\xi}$, where $\boldsymbol{\xi} \sim \mathcal{N}(0, I_m)$ and $L_t L_t^\top = \boldsymbol{\Sigma}_t$. The factor $L_t \in \mathbb{R}^{m\times m}$ converts independent sampling variables $\boldsymbol{\xi} \in \mathbb{R}^m$ into correlated deviations from the posterior mean. Observation errors ϵ_i instead describe deviations of realized gains from their expectations. Execution noise informs inference but is not added to these draws, which compare expected quality rather than individual noisy outcomes.

Joint sampling compares candidates under a common, plausible realization of their expected child scores. For positively correlated candidates, an optimistic draw tends to raise both scores, while a pessimistic draw tends to lower both. Because the scores tend to move together, uncertainty about their absolute quality can coexist with greater confidence in their relative ranking. Holding the candidates' means and individual uncertainties fixed, stronger positive correlation therefore makes reversals of their mean ranking less likely.

[2] Both counts depend on decision time t; we write m and n for m_t and n_t to simplify notation.

Table 1: **Main results on the 15 AlphaEvolve tasks.** Tasks are grouped by problem type. Arrows indicate the preferred direction; bold marks the best score among the compared methods, including ties. Average ranks compare the four methods over all 15 tasks using unrounded scores; ties within 10^{-10} receive mean ranks.

Task	↑/↓	Ours	MLEvolve	Greedy	Agent-rating
Geometric packing / regions					
Hexagon packing	↓	3.944824	3.947878	**3.940341**	3.953391
Circle packing (square)	↑	**2.631092**	2.611866	2.629294	2.630106
Circle packing (rectangle)	↑	**2.359120**	2.348958	2.338491	2.356519
Heilbronn (convex)	↑	**0.030891**	0.030802	0.030769	0.030542
Heilbronn (triangle)	↑	0.036413	**0.036528**	0.036454	0.036235
Kissing number ($d = 11$)	↑	**394**	348	383	348
Additive combinatorics					
Sums and differences 1	↑	1.110246	1.109949	1.113723	**1.122740**
Sums and differences 2	↑	**1.120363**	1.118903	1.112679	1.113825
Autocorrelation / inequalities					
Uncertainty inequality	↓	**0.352099**	**0.352099**	**0.352099**	**0.352099**
First autocorrelation	↓	**1.519079**	1.532056	1.544900	1.544972
Third autocorrelation (var.)	↓	1.485220	1.476867	**1.471381**	1.510232
Third autocorrelation	↓	1.464716	**1.461759**	1.485251	1.482754
Second autocorrelation	↑	**0.890046**	0.868407	0.886555	0.882910
Ratio / overlap optimization					
Max-to-min distance ratio	↓	12.903948	**12.890873**	12.890881	12.902929
Minimum overlap	↓	**0.381114**	0.381597	0.382622	0.382367
Average rank	↓	**1.83**	2.47	2.63	3.07

After selecting p_t, the generation agent produces and evaluates a child solution c_t. A valid outcome extends the graph and updates the GP with the recorded input and observed gain, informing subsequent decisions. Appendix A.4 specifies the treatment of failed and unfinished attempts.

4 Experiments

We compare BayesianAgent with three allocation baselines in the MLEvolve (Du et al., 2026) framework on AlphaEvolve and MLE-bench Lite under benchmark-specific budgets. We then examine the roles of root covariance, LLM potential assessments, and posterior sampling. Appendix F reports supplementary AutoLab (Xu et al., 2026) experiments, including the settings and results.

4.1 Setup

Benchmarks. For mathematical optimization, we use the same 15 AlphaEvolve (Novikov et al., 2025) tasks evaluated by MLEvolve (Du et al., 2026). They span geometric packing, additive combinatorics, autocorrelation inequalities, and distance-ratio and overlap problems. Each task provides an executable verifier and a task-specific objective. For machine-learning engineering, we evaluate all 22 competitions in MLE-bench Lite, the low-complexity split of MLE-bench (Chan et al., 2025), covering image, text, audio, and tabular prediction. Tables 1 and 2 report the complete task-level results for the two benchmarks, respectively.

Methods. **Ours** denotes BayesianAgent. **MLEvolve** (Du et al., 2026) uses its original progressive Monte Carlo graph search (MCGS) policy, which combines UCT traversal with an increasing emphasis on elite solutions. **Parent-metric-greedy** (*Greedy*) selects an eligible parent with the best observed task utility. **Agent-rating-only** (*Agent-rating*) selects the parent with the highest recorded LLM assessment of remaining improvement potential.

Table 2: **MLE-bench Lite results on all 22 tasks.** Arrows indicate the preferred direction; bold marks the best score among the compared methods, including ties. Superscripts G, S, and B denote gold, silver, and bronze medals, respectively. The last row reports gold rate / medal rate (%), both with denominator 22.

Task	↑/↓	Ours	MLEvolve	Greedy	Agent-rating
Aerial Cactus Identification	↑	**1.00000**^{G}	**1.00000**^{G}	**1.00000**^{G}	0.99999
APTOS 2019 Blindness	↑	0.91235	**0.93403**^{G}	0.91445	0.91434
Denoising Dirty Documents	↓	0.00852^{G}	**0.00764**^{G}	0.00908^{G}	0.01315^{G}
Insult Detection	↑	0.95293^{G}	0.95192^{G}	0.94312^{G}	**0.95486**^{G}
Dog Breed Identification	↓	**0.24181**	0.38144	0.33517	0.24642
Dogs vs. Cats	↓	0.00347^{G}	0.02268^{G}	0.00407^{G}	**0.00281**^{G}
Histopathologic Cancer	↑	**0.99769**^{G}	0.99388^{G}	0.99739^{G}	0.99751^{G}
Jigsaw Toxic Comments	↑	**0.98760**^{G}	0.98480	0.98750^{G}	0.98728^{S}
Leaf Classification	↓	0.00006^{S}	0.02072	0.00621^{S}	**0.00002**^{S}
MLSP 2013 Birds	↑	**0.91801**^{S}	0.91108^{S}	0.91007^{S}	0.89833^{B}
NYC Taxi Fare Prediction	↓	8.24182	8.24728	**4.53096**	5.76179
NOMAD 2018 Conductors	↓	0.06086^{S}	0.06007^{S}	**0.05952**^{S}	0.14025
Plant Pathology 2020	↑	**0.99688**^{G}	0.99053^{G}	0.99338^{G}	0.99284^{G}
Random Acts of Pizza	↑	0.69301^{B}	0.61710	**0.70345**^{B}	0.70112^{B}
RANZCR CLiP Catheter/Line	↑	**0.88141**	0.84446	0.48628	0.85187
SIIM–ISIC Melanoma	↑	**0.91188**	0.88367	0.88904	0.86010
Spooky Author Identification	↓	0.23751^{S}	0.27523^{B}	**0.23542**^{S}	0.25511^{S}
Tabular Playground (Dec. 2021)	↑	**0.96325**^{G}	0.96297^{G}	0.96301^{G}	0.96311^{G}
Tabular Playground (May 2022)	↑	0.99409	0.99232	**0.99478**	0.99254
English Text Normalization	↑	**0.99796**^{G}	0.99261^{S}	0.99242^{S}	0.99418^{S}
Russian Text Normalization	↑	**0.98397**^{S}	0.89943	0.98004^{B}	0.97525
ICML 2013 Whale Challenge	↑	0.99309^{G}	0.99258^{G}	0.98908^{S}	**0.99402**^{G}
Gold / Medal rate (%)	↑	**45.45 / 72.73**	40.91 / 59.09	36.36 / **72.73**	31.82 / 59.09

Budgets and implementation. AlphaEvolve runs use an 8-hour experimental budget, at most 500 evolution steps, and a 1,000-second execution cap per candidate; candidates execute on CPUs. MLE-bench Lite runs use a 12-hour cumulative experimental budget and one NVIDIA A100 per task. All experiments use Muse Spark 1.3 (Meta AI Research, 2026) as the specified language model.

Metrics and reporting. For AlphaEvolve, we report verifier-evaluated scores in their original objective units and average within-task ranks across the compared methods. Lower average rank is better; scores within an absolute tolerance of 10^{-10} are treated as ties and receive mean ranks. For MLE-bench Lite (Chan et al., 2025), we report the fraction of tasks earning gold (*gold rate*), followed by the fraction earning at least bronze (*medal rate*). Medals are determined using official held-out submission scores and the benchmark's leaderboard thresholds. Both rates use all 22 tasks as the denominator. Internal validation scores guide evolution but do not determine the reported medals. The GP hyperparameters and fitting bounds are specified in Appendices A.2 and A.3.

4.2 Main Results

AlphaEvolve. Table 1 reports results on the 15 mathematical optimization tasks. BayesianAgent achieves the best average rank of **1.83**, compared with 2.47 for MLEvolve, 2.63 for Greedy, and 3.07 for Agent-rating. It obtains the best score among the four policies on **9 of 15 tasks**, comprising eight outright wins and one shared tie on the uncertainty inequality. These wins span packing, additive combinatorics, autocorrelation, and overlap objectives, rather than being confined to a single problem type. For example, on the 11-dimensional kissing-number task, BayesianAgent reaches 394, compared with 348 for MLEvolve, 383 for Greedy, and 348 for Agent-rating. The advantage is not uniform: Greedy performs best on hexagon packing, while MLEvolve leads on the max-to-min distance ratio.

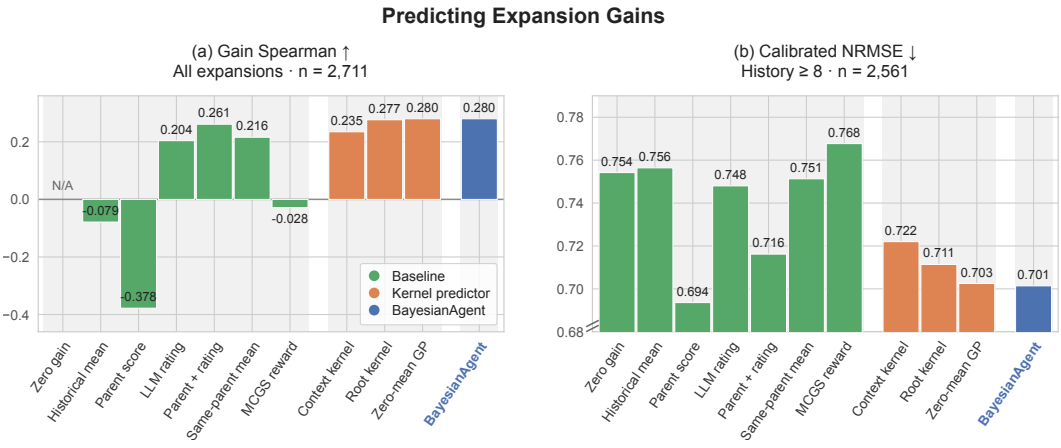

Figure 2: **Predicting expansion gains on 15 AlphaEvolve tasks.** Green bars show baselines; orange bars show context-kernel, root-kernel, and zero-mean full-kernel GPs; blue bars show BayesianAgent. Bars report equally weighted task means. **(a)** Gain Spearman on all 2,711 valid expansions; higher is better. **(b)** Calibrated NRMSE on the common 2,561-expansion subset with at least eight completed historical observations; lower is better. Predictors use the same histories and only feedback available before selection.

MLE-bench Lite. Table 2 reports held-out scores and medal attainment on all 22 competitions. BayesianAgent achieves the highest **gold rate of 45.45%** (10/22), compared with 40.91% (9/22) for MLEvolve, 36.36% (8/22) for Greedy, and 31.82% (7/22) for Agent-rating. Its **medal rate of 72.73%** (16/22) matches Greedy and exceeds the 59.09% (13/22) attained by both MLEvolve and Agent-rating. The comparison with Greedy is particularly informative: BayesianAgent earns two additional gold medals while matching the total number of medal-winning tasks.

4.3 Predicting Expansion Gains

Most existing agentic evolution systems use rule-based selection, allocating experiments according to observed performance, LLM assessments, or historical search statistics. BayesianAgent instead learns a posterior distribution over expansion gains and uses it to guide allocation. We therefore test whether its posterior mean predicts realized gains more accurately and ranks them better than these existing signals. Figure 2 compares predictions of normalized gain y_i (Equation 1) on the same histories from BayesianAgent runs. Green bars show baselines; orange bars show context-kernel, root-kernel, and zero-mean full-kernel GPs; blue bars show BayesianAgent. These GP variants examine which information sources support gain prediction.

Gain Spearman measures rank correlation with y_i; native NRMSE measures root mean squared error in normalized gain units. Affine calibration maps each predictor's scores to gain units using only prior feedback; its evaluation requires at least eight completed valid observations. Appendix C provides predictor definitions, calibration details, and sensitivity analyses. This analysis evaluates predictions on valid expansions selected by BayesianAgent, not alternative online policies or posterior uncertainty calibration.

Across predictor groups, the GP predictors combined positive gain correlations with lower calibrated errors than LLM rating, historical mean, and MCGS reward (Figure 2). These results support predicting gains from candidate features beyond ratings or summary statistics alone.

Historical mean and MCGS reward Q^{MCGS} showed no positive average gain correlation, with Spearman values of -0.0787 and -0.0283, respectively. History itself remained informative: restricting the mean to previous expansions of the same parent yielded a positive correlation of 0.2159. Context and root kernels reached 0.2349 and 0.2767, respectively, consistent with the value of relating past outcomes to the current candidate.

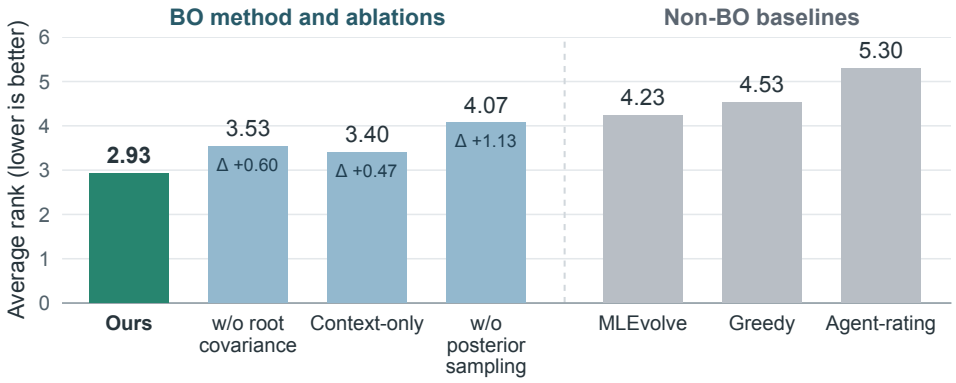

Figure 3: **AlphaEvolve ablations and non-BO baselines.** Bars show average within-task ranks over 15 tasks (lower is better), with all seven methods ranked jointly using unrounded scores and mean ranks for ties within 10^{-10}. Δ denotes the rank increase over the full method.

Parent score illustrates why ranking and calibrated error differ: it correlated negatively with gain (-0.3780), yet achieved the lowest calibrated NRMSE (0.6936). Calibration learned the reversed relationship, so accurate gain estimates need not imply useful rankings from the original score.

BayesianAgent achieved gain Spearman of **0.2800**, exceeding all baselines, including LLM rating (0.2040) and parent-plus-rating regression (0.2612). Its near-identical Spearman to the zero-mean full-kernel GP (0.2801) does not establish an additional benefit from the nonzero prior mean. Its calibrated NRMSE of 0.7014 was also the lowest among the GP predictors (Table 3). The Spearman advantage over MCGS reward held on **14 of 15 tasks**.

4.4 Ablations

We compare three ablations of the BO method on the 15 AlphaEvolve tasks. The **w/o root covariance** variant removes k_{root} from Equation 4 while retaining context and potential-assessment terms. **Context-only** additionally removes the assessment-dependent mean and covariance, leaving a zero-mean GP with the context kernel alone. Both variants retain joint Thompson sampling.

W/o posterior sampling retains the full GP, root covariance, LLM potential assessments, and online updates. It replaces joint Thompson sampling with deterministic selection of the parent having the largest posterior mean of expected child score, $p_t \in \arg\max_{p \in \mathcal{P}_t} (\boldsymbol{\mu}_t)_p$.

Figure 3 ranks the full method, its ablations, and the three non-BO baselines jointly within each task. The full method achieves the lowest average rank of **2.93**, compared with 3.53 without root covariance and 3.40 for Context-only. This ordering does not establish an aggregate benefit from potential assessments without root covariance. Posterior-mean selection obtains 4.07, the largest rank increase among the ablations, despite leading on three tasks. MLEvolve, Greedy, and Agent-rating obtain 4.23, 4.53, and 5.30, respectively. Appendix E provides detailed scores; attribution remains subject to the execution-condition differences in Appendix B. Appendix D provides convergence analysis.

5 Conclusion

We introduced BayesianAgent to learn where to invest experiments in agentic evolution. Its Gaussian process shares evidence across candidates, and joint Thompson sampling combines predicted gains, uncertainty, and parent performance to select expansions. Under matched budgets, it achieves the best AlphaEvolve average rank and the highest gold and joint-highest medal rates on MLE-bench Lite among evaluated methods. Its predictions rank observed gains more accurately than MLEvolve's historical rewards. Future work should assess uncertainty calibration and model failures and longer-term outcomes.

AI USE STATEMENT

In the preparation of this manuscript, GPT and Claude were utilized as writing assistants. The usage was strictly limited to improving grammar, clarity, and overall readability. All scientific ideas, experimental results, and conclusions were conceived and formulated exclusively by the authors. All text polished or modified by the LLM was subsequently reviewed and edited by the authors to ensure that the original scientific meaning was accurately preserved.

ETHICS STATEMENT

All experiments are conducted on publicly available databases. We follow established ethical guidelines in data usage and acknowledge that historical biases present in these resources may be reflected in our results, which is independent of model development.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, Section 3 describes the Gaussian-process surrogate, posterior inference, and joint Thompson sampling used by BayesianAgent. Appendix A specifies the expansion features, score normalization, potential-assessment criteria, covariance functions, prior parameters, hyperparameter-fitting procedure, numerical implementation, and handling of failed and unfinished attempts. Appendix B describes candidate construction and integration with the MLEvolve execution framework. Section 4.1 and Appendix B provide the benchmark tasks, baseline policies, experimental budgets, computational resources, and evaluation protocols, including the rules used to determine MLE-bench Lite medals. Appendix B also documents execution-condition differences relevant to interpreting the comparisons. Appendix C details the expansion-gain prediction analysis, including decision-time information restrictions, historical calibration, and run-selection sensitivity analyses. Appendix D specifies the convergence-reporting procedure. We will release our implementation upon publication to support independent reproduction of the reported results.

REFERENCES

Dhruv Agarwal, Manoj Ghuhan Arivazhagan, Rajarshi Das, Sandesh Swamy, Sopan Khosla, and Rashmi Gangadharaiah. Searching for optimal solutions with LLMs via Bayesian optimization. In *International Conference on Learning Representations (ICLR)*, 2025.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Lilian Weng, and Aleksander Mądry. MLE-bench: Evaluating machine learning agents on machine learning engineering. In *International Conference on Learning Representations (ICLR)*, 2025.

Shangheng Du, Xiangchao Yan, Jinxin Shi, Zongsheng Cao, Shiyang Feng, Zichen Liang, Boyuan Sun, Tianshuo Peng, Yifan Zhou, Xin Li, Jie Zhou, Liang He, Bo Zhang, and Lei Bai. MLEvolve: A self-evolving framework for automated machine learning algorithm discovery. *arXiv preprint arXiv:2606.06473*, 2026.

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.

Donald R. Jones, Matthias Schonlau, and William J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13:455–492, 1998.

Robert Tjarko Lange, Yuki Imajuku, and Edoardo Cetin. ShinkaEvolve: Towards open-ended and sample-efficient program evolution. In *International Conference on Learning Representations (ICLR)*, 2026.

Jin Li, Ahmed Murtadha, Zhiyu Wang, Qiwen Chen, William Chen, Yifei Wu, Guan Wang, Andy L. Siy, Jiayi Yang, Mengsha Huang, Wenhao Li, Yixuan Liu, Shuailin Pan, Mingli Yuan, Sen Song, and Yuhao Sun. Praxist: From experimental artifacts to solution lineages. *arXiv preprint arXiv:2608.25955*, 2026.

Zujie Liang, Feng Wei, Wujiang Xu, Yuxi Qian, Lin Chen, and Xinhui Wu. I-MCTS: Enhancing agentic AutoML via introspective Monte Carlo tree search. In *Findings of the Association for Computational Linguistics: EACL 2026*, pp. 189–210, 2026. doi: 10.18653/v1/2026.findings-eacl.11.

Tennison Liu, Nicolás Astorga, Nabeel Seedat, and Mihaela van der Schaar. Large language models to enhance Bayesian optimization. In *International Conference on Learning Representations (ICLR)*, 2024.

Zexi Liu, Yuzhu Cai, Xinyu Zhu, Yujie Zheng, Runkun Chen, Ying Wen, Yanfeng Wang, Weinan E, and Siheng Chen. ML-master: Towards AI-for-AI via integration of exploration and reasoning. *arXiv preprint arXiv:2506.16499*, 2025.

Meta AI Research. Introducing muse spark 1.3, 2026. URL https://research.meta.ai/blog/introducing-muse-spark-1-3. Blog post / Technical announcement.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. AlphaEvolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, and Alhussein Fawzi. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024.

Binxin Ru, Xingchen Wan, Xiaowen Dong, and Michael A. Osborne. Interpretable neural architecture search via Bayesian optimisation with Weisfeiler-Lehman kernels. In *International Conference on Learning Representations (ICLR)*, 2021.

Daniel J. Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, and Zheng Wen. A tutorial on Thompson sampling. *Foundations and Trends in Machine Learning*, 11(1):1–96, 2018. doi: 10.1561/2200000070. URL https://arxiv.org/abs/1707.02038.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.

Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *International Conference on Machine Learning (ICML)*, 2010.

Edan Toledo, Karen Hambardzumyan, Martin Josifoski, Rishi Hazra, Nicolas Baldwin, Alexis Audran-Reiss, Michael Kuchnik, Despoina Magka, Minqi Jiang, Alisia Maria Lupidi, Andrei Lupu, Roberta Raileanu, Tatiana Shavrina, Kelvin Niu, Jean-Christophe Gagnon-Audet, Michael Shvartsman, Shagun Sodhani, Alexander H. Miller, Abhishek Charnalia, Derek Dunfield, Carole-Jean Wu, Pontus Stenetorp, Nicola Cancedda, Jakob Nicolaus Foerster, and Yoram Bachrach. AI research agents for machine learning: Search, exploration, and generalization in MLE-bench. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. doi: 10.52202/085713-1186.

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.

Jiayi Zhang, Yongfeng Gu, Jianhao Ruan, Maojia Song, Yiran Peng, Zhiguang Han, Jinyu Xiang, Zhitao Wang, Caiyin Yang, Yixi Ouyang, Bang Liu, Chenglin Wu, and Yuyu Luo. Harnessing agentic evolution. *arXiv preprint arXiv:2605.13821*, 2026.

A BayesianAgent Representation and Inference Details

A.1 Expansion Representation

Expansion context. To predict an expansion before its child exists, we describe the parent and the operation proposed for it. Before each decision, the execution system supplies one proposed expansion for each candidate parent $p \in \mathcal{P}_t$. The proposal specifies an operation $o_t(p)$ and any required reference solutions. The input $x_t(p)$ combines the parent's current expansion context with its LLM potential rating. Because each parent has one proposal, selecting a parent also selects that expansion. The context is updated as evolution proceeds, and the operation is represented as a categorical input. Operations and reference selection depend on the execution system; Appendix B describes the MLEvolve instance.

Frozen reference and scale. Historical and current inputs must use the same score normalization so that their features and gains remain comparable as evolution proceeds. Let $\mathcal{S}_0$ be the multiset of scores from valid initial drafts available before the first parent-selection decision. The reference location and scale are fixed once using these scores:

$$s_{\mathrm{ref}} = \mathrm{median}(\mathcal{S}_0), \qquad \kappa = \max\left\{\mathrm{IQR}(\mathcal{S}_0),\, 0.05\,\mathrm{median}_{s\in\mathcal{S}_0}\,|s|,\, 10^{-6}\right\}. \tag{9}$$

Here IQR is the interquartile range, computed using linear interpolation for the quartiles. The second and third terms prevent the scale from collapsing when initial scores have little spread or are close to zero. Subsequent observations do not change either quantity, so new results cannot rescale earlier observations.

Context features. The context vector summarizes the parent's current quality, how it was obtained, and what previous attempts reveal about further improvement. An expansion attempt is complete once execution and any permitted debugging have reached a final outcome. Let $N_{p,o}$ count completed attempts for parent p and operation o, including failures. Let $M_{p,o}$ count the valid attempts among them, and let $S_{p,o}$ sum their unnormalized child-minus-parent score gains. The birth gain $\Delta_{\mathrm{birth}}(p)$ is the score difference between p and the parent that generated it; it equals zero for roots. The four features used in the shared context kernel are

$$\begin{aligned} z_1 &= \tanh\left(\frac{s(p) - s_{\mathrm{ref}}}{\kappa}\right), \qquad z_2 = \tanh\left(\frac{\Delta_{\mathrm{birth}}(p)}{\kappa}\right), \\ z_3 &= \tanh\left(\frac{S_{p,o}}{(M_{p,o}+1)\kappa}\right), \quad z_4 = \frac{\log(1+N_{p,o})}{1+\log(1+N_{p,o})}. \end{aligned} \tag{10}$$

These coordinates encode parent performance, birth gain, historical mean expansion gain with shrinkage, and expansion attempt count, respectively. The transformations keep all four coordinates bounded, compressing large score differences and long attempt histories. The additional denominator count in z_3 pulls the gain estimate toward zero when few valid attempts are available. The last two features use the history of the same parent–operation pair. All counts exclude pending or censored outcomes and retain the original proposal's attribution after debugging. For the ith expansion, we record x_i when the proposal is dispatched, before child c_i is generated. Its eventual gain y_i becomes the regression target; subsequent results do not change this recorded input.

LLM potential. Experimental history may be sparse, so the LLM assessment provides an additional judgment of the parent's remaining improvement potential. Each completed valid node receives an assessment $q(p) \in \{0, 1, 2\}$ when sufficient evidence is available. The values 0, 1, and 2 indicate no clear improvement evidence, plausible incremental improvement, and concrete evidence of substantial feasible improvement, respectively. The assessor returns a missing value when the execution is invalid or the available evidence is insufficient. It uses only the completed node's code, public training or validation diagnostics, task rules, and resource protocol. Private test scores and future experimental results are excluded from this assessment. We use the centered potential rating $r(p) = q(p) - 1$, assigning $r(p) = 0$ when the assessment is unavailable. The assessment enters allocation when the completed node is subsequently considered as a parent; experimental gains provide the regression targets.

Root assignment. Root identity records shared derivation, allowing the model to relate descendants that may retain common structures or training procedures. Each initial draft serves as a root, and descendants inherit its identifier through their primary parents. Fusion retains its primary parent's root identifier, while aggregation establishes a new root.

A.2 Covariance Structure and Prior Parameters

The covariance combines similarity in expansion context, shared derivation, and uncertainty about the contribution of the LLM rating. We collect these inputs as $x = (z, o, r, b, v)$, where b is the root identifier and $v = (s(p) - s_{\mathrm{ref}})/\kappa$ is standardized parent performance. For a current candidate parent p, the operation coordinate in $x_t(p)$ is $o = o_t(p)$. The root kernel uses v to distinguish descendants whose performance has diverged. Unlike the bounded context coordinate $z_1 = \tanh(v)$, this coordinate retains large performance differences without compression.

As in Section 3.1, we write $f(x) = \theta r + g(x)$, separating the rating contribution from the remaining variation in expected gains. We assign $\theta \sim \mathcal{N}(\mu_\theta, \sigma_\theta^2)$ and an independent prior $g \sim \mathcal{GP}(0, k_{\mathrm{context}} + k_{\mathrm{root}})$. The shared coefficient contributes mean $\mu_\theta r$ and covariance $\sigma_\theta^2 r r'$, giving Equations 3 and 4 after integration over θ.

Both component kernels use distance to reduce covariance as candidates become less similar. For Euclidean inputs w, w' and length scale $\ell > 0$, define the Matérn-$3/2$ correlation (Rasmussen & Williams, 2006, Equation (4.17)) as

$$M_{3/2}(w, w'; \ell) = \left(1 + \frac{\sqrt{3}\|w - w'\|_2}{\ell}\right) \exp\left(-\frac{\sqrt{3}\|w - w'\|_2}{\ell}\right). \tag{11}$$

The length scale controls how quickly correlation decreases with distance between the inputs. Let σ_f and τ denote the context and root amplitudes, with respective length scales ℓ and ℓ_b. Similar parents may respond differently to different operations, so the context kernel also compares their proposed operations. The factor $\gamma \in [0, 1]$ specifies the relative covariance for different operations, compared with matching operations at the same context distance. The two component kernels are

$$k_{\mathrm{context}}(x, x') = \sigma_f^2 M_{3/2}(z, z'; \ell)\big[\gamma + (1 - \gamma)\mathbf{1}\{o = o'\}\big], \tag{12}$$

$$k_{\mathrm{root}}(x, x') = \tau^2 \mathbf{1}\{b = b'\} M_{3/2}(v, v'; \ell_b). \tag{13}$$

Here $\mathbf{1}\{\cdot\}$ equals one when its condition holds and zero otherwise. The context kernel uses only (z, o), the root kernel uses only (b, v), and the assessment covariance term uses only r. Thus, nearby context vectors can share evidence across roots, with covariance multiplied by γ when operations differ. The root term adds covariance only for a shared root, and this contribution decreases as the parents' performance diverges. BayesianAgent fixes $\gamma = 0.25$, $\tau = 1.5$, $\ell_b = 1$, and $\mu_\theta = \sigma_\theta = 0.25$. The initial shared-kernel parameters are $(\ell, \sigma_f, \sigma_n) = (1, 1, 0.25)$.

A.3 Posterior Inference and Sampling

Hyperparameter fitting. As valid observations accumulate, we fit the context length scale, context amplitude, and observation-noise level to the observed gains. Before eight valid expansion observations are available, inference uses the initial values given above. Thereafter, each changed training set triggers a maximum a posteriori fit of their logarithms, $\eta = (\log \ell, \log \sigma_f, \log \sigma_n)^\top \in \mathbb{R}^3$. Let X collect the n observed expansion inputs, $\mathbf{y} \in \mathbb{R}^n$ their normalized gains, and $K_\eta \in \mathbb{R}^{n \times n}$ their kernel matrix. Writing $\eta_0 = (0, 0, \log 0.25)^\top$, $A_\eta = K_\eta + \sigma_n^2 I_n$, and $\mathbf{e} = \mathbf{y} - \mu_0(X)$, the objective is

$$\mathcal{L}(\eta) = \tfrac{1}{2}\mathbf{e}^\top A_\eta^{-1}\mathbf{e} + \tfrac{1}{2}\log|A_\eta| + \tfrac{n}{2}\log(2\pi) + \tfrac{1}{2}\|\eta - \eta_0\|_2^2. \tag{14}$$

The first three terms are the negative log marginal likelihood, measuring how the model accounts for the observed gains. The final term is a Gaussian prior penalty on η, discouraging large departures from the initial log parameters η_0. The fit restarts from η_0 and uses L-BFGS-B with analytic gradients. Bounds are $\ell \in [0.25, 4]$, $\sigma_f \in [0.1, 5]$, and $\sigma_n \in [0.05, 2]$. Optimization is bounded by 25 iterations, 150 objective evaluations, and 30 seconds per fit. The root parameters and the prior parameters of θ remain fixed during fitting; conditioning on outcomes still updates the inferred rating contribution. Posterior uncertainty is conditional on the fitted shared-kernel hyperparameters rather than integrated over them.

Posterior conditioning and sampling. Conditioning uses completed expansions to update the predicted gains of the current candidate parents. At decision time t, let X_t collect the current inputs of the m candidate parents in $\mathcal{P}_t$. The arrays X and X_t are ordered collections of n historical and m current inputs, respectively. Each input is a tuple (z, o, r, b, v) containing numerical and categorical coordinates; operation and root labels enter the kernel through equality tests. For ordered input collections U, V, the matrix $K(U, V)$ has entries $[K(U, V)]_{ij} = k(u_i, v_j)$, and $\mu_0(U)$ stacks the prior means. With $A = K(X, X) + \sigma_n^2 I_n \in \mathbb{R}^{n\times n}$, the posterior moments of normalized gains are

$$\boldsymbol{\mu}_t^f = \mu_0(X_t) + K(X_t, X)A^{-1}[\mathbf{y} - \mu_0(X)], \tag{15}$$

$$\mathbf{C}_t = K(X_t, X_t) - K(X_t, X)A^{-1}K(X, X_t). \tag{16}$$

Thus $\boldsymbol{\mu}_t^f \in \mathbb{R}^m$ and $\mathbf{C}_t \in \mathbb{R}^{m\times m}$; all posterior vectors and matrices use the same ordering of candidate parents in $\mathcal{P}_t$. The cross-covariance $K(X_t, X) \in \mathbb{R}^{m\times n}$ determines how the n observed gains inform the m current predictions. For a single observation, a gain above its prior expectation raises predictions at inputs with positive prior covariance; a below-expectation gain lowers them. Thus, even a positive observed gain can reduce predicted gains elsewhere if it falls below expectation.

Selection compares expected child scores, so we first restore the original score units and then add each parent's measured score. Let $\mathbf{s}_t$ collect the measured parent scores and $\mathbf{Q}_t$ the expected child scores, with $(\mathbf{s}_t)_p = s(p)$ and $(\mathbf{Q}_t)_p = Q_t(p)$. The child-score posterior in Equation 7 then has moments

$$\boldsymbol{\mu}_t = \mathbf{s}_t + \kappa\boldsymbol{\mu}_t^f, \qquad \boldsymbol{\Sigma}_t = \kappa^2\mathbf{C}_t. \tag{17}$$

To compare candidates while preserving their posterior dependence, we transform a vector of independent standard normal draws into one joint score sample:

$$\widetilde{\mathbf{Q}}_t = \boldsymbol{\mu}_t + L_t\boldsymbol{\xi}, \qquad \boldsymbol{\xi} \sim \mathcal{N}(0, I_m), \qquad L_tL_t^\top = \boldsymbol{\Sigma}_t. \tag{18}$$

Here $L_t \in \mathbb{R}^{m\times m}$ and $\boldsymbol{\xi}, \widetilde{\mathbf{Q}}_t \in \mathbb{R}^m$, so one joint draw assigns a sampled expected child score to each candidate. Multiplication by L_t produces the required covariance $\boldsymbol{\Sigma}_t$ between sampled scores; the parent with the largest sampled score is selected. Posterior inference uses Cholesky solves with diagonal jitter for numerical stability. Joint sampling uses a symmetric eigendecomposition, clipping only negative eigenvalues attributable to numerical roundoff. Sampling targets expected child quality, so it adds no independent observation noise to the latent posterior covariance.

A.4 Outcome Handling and Model Updates

To keep training examples consistent with the selection decision, each observed gain is paired with the input recorded before its expansion began. Only candidates that pass the task checks and yield a finite evaluation metric are admitted to the solution graph. A completed valid expansion contributes one normalized gain observation to the training set. Valid negative gains are retained, and new valid observations update the posterior; hyperparameters are fitted according to Appendix A.3. Completed failed attempts contribute to the attempt-history feature but receive no numerical quality label. Pending and budget-censored attempts remain outside the completed history and do not supply regression targets. When debugging resolves a proposal, its final outcome retains the original proposal's parent, operation, and recorded expansion context. Parallel execution follows the execution system's pending policy; the GP uses only completed feedback without synthetic pending outcomes.

Modeling scope. The acquisition targets expected child quality conditional on a valid outcome. It does not explicitly model failure probability, execution cost, or longer-term returns.

A.5 Examples from Prior Experiments

The following cases illustrate the similarities represented by the context and root kernels, followed by recorded examples of posterior updating.

Refinements inherited from one initial solution. In NOMAD transparent-conductor prediction, one initial solution used a multilayer perceptron (MLP) with crystal pair-distance features to jointly predict formation energy and bandgap. Its descendants retained this representation while refining the

training procedure. Validation used root mean squared logarithmic error (RMSLE), averaged across the two targets. Standardizing the log-transformed targets during training reduced the recorded validation RMSLE from 0.06051 to 0.05731. A subsequent three-seed ensemble reduced it further to 0.05606 on the same validation split, grouped by crystal space group. The validation metric and partition remained fixed across these refinements. This lineage illustrates how an initial model can define a continuing training direction, motivating covariance between gains within a root. Root identity follows derivation rather than model class, so separately initialized MLPs would remain distinct roots.

Similar contexts across roots. In New York City Taxi Fare Prediction, two parents from different roots combined a meter-based fare estimate with boosted-tree residual prediction. Their validation root mean squared errors were 3.0507 and 3.0545, and both had small positive birth gains and no completed local refinement attempts. Their recorded context vectors differed by only 0.0027 in Euclidean distance, and both proposed the improvement operation. Favoring recent training examples improved the first parent's recorded error to 3.0407; filtering implausible unit fares from training improved the second's to 3.0224. Thus, similar performance and refinement histories accompanied remaining improvement opportunities in distinct roots, although the successful modifications addressed different data limitations. These observations motivate context-based sharing without establishing that one outcome caused a useful prediction update for the other.

Distinct refinement directions in image classification. Different initial solutions on the same task can retain distinct modeling choices, motivating separate roots. Leaf Classification provides an example with two independently initialized solutions. One combined Gaussian class densities for margin, shape, and texture features, whereas another added a frozen SigLIP2 image-encoder view. Both used five outer cross-validation folds, with fusion weights and temperature calibration selected through four inner folds. In the first lineage, retaining 64 rather than 32 principal components per feature block reduced recorded log loss from 0.0340 to 0.0216. In the second, restricting calibration temperature to values of at least one reduced recorded log loss from 0.0274 to 0.0102. These refinements preserved the parents' feature views while addressing feature truncation and overconfident calibration, respectively. All three examples use internal validation feedback reused during evolution; they establish neither independent test improvements nor causal benefits of the covariance components.

Recorded posterior responses. These examples show how an observed gain changes the model's predictions, beyond the structural similarities illustrated above. The first target-standardization refinement in NOMAD also provides direct evidence of inference through the root component. Its normalized gain was 1.0571, exceeding the prior expectation of 0.2500 by 0.8071. The root contribution below is the part of the predicted normalized gain associated with k_{root}. At the observed parent's standardized performance, the inferred root contribution increased from zero to 0.5380, while its variance decreased from 2.2500 to 0.7500. At performance coordinates two normalized units away, the same root's mean contribution increased only to 0.0752. The corresponding contributions for other roots remained unchanged, although their total predictions could respond through other kernel terms. These recorded component posteriors agree with Gaussian conditioning and illustrate the performance-local nature of root sharing.

The Leaf Classification case illustrates why an improvement can nevertheless lower predictions for another candidate. The image-augmented lineage subsequently increased the retained principal components from 32 to 64, reducing log loss from 0.010208 to 0.009389. Despite this improvement, the normalized gain of 0.2463 fell below the GP's preceding expectation of 0.8419. An unchanged candidate in the Gaussian-only lineage consequently had its predicted normalized gain reduced from 0.6247 to 0.5988. This update followed a single additional observation, with the previous observations, candidate input, normalization, and hyperparameters held fixed. Reconditioning the recorded posterior reproduced the change, using a positive conditional cross-covariance of 0.1292. Because the roots differed and the observed input had a neutral rating, only k_{context} contributed to their direct prior covariance. The conditional cross-covariance nevertheless depended on the full model and preceding observations. The candidate remained unselected in the adjacent joint draws, demonstrating prediction updating without establishing a selection benefit.

B Implementation Details

We implement BayesianAgent within the MLEvolve framework (Du et al., 2026), whose agents generate, execute, and evaluate candidate solutions. The framework maintains the solution graph and determines the available expansion operations; BayesianAgent uses completed experimental feedback to allocate these expansion opportunities. The surrogate representation and inference procedures are specified in Appendix A.

B.1 Evolution and Execution

Candidate construction and execution. Evolution begins with three initial drafts, each defining a root in the solution graph. At an allocation decision, eligible parents are evaluated solutions that remain available for expansion under the framework's validity and branching rules. For each parent, MLEvolve specifies one feasible operation: improvement of the current solution, evolution informed by its refinement history, or fusion with reference solutions. Any fusion references are determined before allocation. Each parent in $\mathcal{P}_t$ therefore identifies a concrete proposed expansion, so selecting p_t also fixes the operation and its reference solutions. The corresponding generation agent carries out this expansion, preserving the proposal on which the allocation decision was based. Initial drafting, debugging, and aggregation follow MLEvolve's scheduling rules and are handled separately from this parent-selection step.

Feedback and potential assessment. Allocation uses only information available before the selected expansion is executed. A completed valid expansion supplies an observed gain, and the review of its solution also assesses the remaining feasible improvement potential. The assessment uses the solution, public training or validation diagnostics, task requirements, and the stated resource budget. It is retained for subsequent parent selection, so evaluating the candidate set does not require a new language-model assessment of every parent. Private test scores and future outcomes are excluded from this feedback. Appendix A.1 defines the three potential levels and the neutral treatment of missing assessments; Appendix A.4 specifies how failed, pending, and budget-censored attempts are handled.

B.2 Experimental Configuration

Task coverage and computational budgets. All methods are evaluated on the same 15 AlphaEvolve mathematical tasks and 22 MLE-bench Lite competitions, listed in Tables 1 and 2, respectively. AlphaEvolve runs execute on CPUs with an 8-hour experimental budget, a maximum of 500 evolution steps, and a 1,000-second execution cap per candidate. MLE-bench Lite runs receive a 12-hour cumulative experimental budget and one NVIDIA A100 per task. These are budgets for the complete evolutionary run; the number of evaluated candidates depends on the time required to generate and execute individual solutions.

Language model and execution conditions. Muse Spark 1.3 with thinking effort set to high is the specified language model for solution generation and feedback. Some AlphaEvolve runs changed language-model providers or CPU allocations during execution, and potential assessment in BayesianAgent used an earlier description of the available resources. Such differences can affect experiment throughput and assessments of feasible improvement. The AlphaEvolve comparisons, including the component ablations, therefore remain subject to these execution-condition differences when attributing performance gains to the allocation policy or surrogate design.

B.3 Evaluation and Reporting

MLE-bench Lite. Internal validation feedback guides evolution, while the final submission is graded on the benchmark's held-out test data. Medals are determined from this held-out score using the competition-specific leaderboard thresholds supplied by MLE-bench (Chan et al., 2025). A score equal to a threshold qualifies for that medal, and each task receives the highest medal for which it qualifies. Let N_{gold} be the number of tasks earning gold and $N_{\mathrm{bronze+}}$ the number earning at least bronze, including silver and gold. The gold rate and medal rate are $100\,N_{\mathrm{gold}}/22$ and

$100\, N_{\text{bronze+}}/22$, respectively. Both rates use the full set of 22 competitions as the denominator, so each task contributes equally to the aggregate evaluation.

AlphaEvolve. We report verifier-evaluated objective values for the selected solutions in their original task units and optimization directions. For allocation, minimization objectives are negated to obtain the higher-is-better utility used in Section 3; this transformation does not change the reported values. Task-level ranks are computed from unrounded scores, with values within an absolute tolerance of 10^{-10} treated as ties and assigned their mean rank. We average these ranks over all 15 tasks, using the four allocation policies for Table 1 and a common set of seven methods for Figure 3 and Table 9. The latter comparison includes the full method, three ablations, and three non-BO baselines; its ranks are recomputed jointly from the task-level scores. This rank-based aggregation gives each task equal weight without averaging objectives with different units or scales. A task win is assigned to every method whose score is tied for the best value within the corresponding comparison.

C Expansion-Gain Prediction Analysis

C.1 Evaluation and Historical Information

We evaluate 2,711 valid expansions from the original BayesianAgent runs on all 15 AlphaEvolve tasks. For each expansion, every predictor uses the same completed observations available to the surrogate when that expansion was selected. Feedback from unfinished expansions and outcomes arriving after selection is excluded. Historical features and predictions retain their decision-time values, preventing later information from changing earlier forecasts. Negative gains remain in the evaluation; invalid outcomes and unexecuted candidates have no target in this analysis. When debugging repairs a proposal, the gain remains relative to its original expansion parent.

For task j, let $\mathcal{I}_j$ index its evaluated expansions and let h_i be a predictor's decision-time score for expansion i. For BayesianAgent, $h_i = (\boldsymbol{\mu}^f_{t_i})_{p_i}$, where t_i is the decision at which parent p_i was selected. For predictors expressed in normalized gain units, the task-level metrics are

$$\rho_j = \text{Spearman}\big((h_i)_{i\in\mathcal{I}_j}, (y_i)_{i\in\mathcal{I}_j}\big), \qquad \text{NRMSE}_j = \sqrt{\frac{1}{|\mathcal{I}_j|}\sum_{i\in\mathcal{I}_j}(h_i - y_i)^2}. \tag{19}$$

We average these metrics with equal task weights, rather than pooling expansions across tasks. All nonconstant scores have defined correlations on all 15 tasks in the primary analysis. Constant-score correlations are undefined and are not replaced by zero. MCGS reward Q^{MCGS} and parent score are ranking signals without native gain units, so their native NRMSE is not reported. We do not subtract parent utility from the MCGS reward score.

C.2 Predictors and Kernel Diagnostics

Simple predictors. Zero gain predicts $h_i = 0$, equivalent to predicting that the child inherits its parent's score. Historical mean predicts the average normalized gain in the available history. Parent score uses $v = (s(p) - s_{\text{ref}})/\kappa$, a positive affine transformation of the measured parent score. Its negative correlation with gain does not imply an absence of predictive information. For example, reversing its direction gives gain Spearman 0.3780 in the original runs; this is a directional diagnostic rather than a fitted predictor. The LLM rating predictor uses $0.25r$, matching the prior mean in Equation 3. The positive factor does not affect its Spearman correlation but determines its native prediction error. MCGS reward Q^{MCGS} is the decision-time historical reward score maintained on the same BayesianAgent trajectories. It is distinct from the expected child score $Q_t(p)$ modeled in Section 3.2. It is not a reconstructed execution of the full MCGS policy.

Parent-plus-rating regression. We fit a ridge regression of normalized gain on parent performance v and LLM rating r. Both inputs are standardized using only the available training history, with a unit scale for numerically constant features. Writing $\widetilde{w}_i \in \mathbb{R}^2$ for these standardized inputs and $\beta \in \mathbb{R}^2$ for their regression weights, the fit minimizes

$$\sum_{i\in\mathcal{H}_t}(y_i - a - \widetilde{w}_i^{\mathsf{T}}\beta)^2 + \|\beta\|_2^2, \tag{20}$$

Table 3: **Predicting expansion gains on 15 AlphaEvolve tasks.** Gain Spearman and native NRMSE use all 2,711 valid expansions; calibrated NRMSE uses the common 2,561-expansion subset with at least eight historical observations. Metrics are averaged with equal task weights, with preferred directions indicated in the headers. Predictors share the same histories and use only feedback available before selection. Kernel diagnostics retain the parameters fitted at that decision and do not rerun the online ablations. N/A denotes an undefined constant-score correlation or a score without native gain units.

Predictor	Gain Spearman ↑ All expansions	Native NRMSE ↓ All expansions	Calibrated NRMSE ↓ History ≥ 8
Zero gain	N/A	0.7702	0.7542
Historical mean	-0.0787	0.7892	0.7564
Parent score	-0.3780	N/A	0.6936
LLM rating	0.2040	0.8090	0.7480
Parent + rating regression	0.2612	0.7587	0.7162
Same-parent historical mean	0.2159	0.7499	0.7513
MCGS reward Q^{MCGS}	-0.0283	N/A	0.7677
Context-kernel predictor	0.2349	0.7221	0.7220
Root-kernel predictor	0.2767	0.7193	0.7114
Zero-mean GP, full kernel	0.2801	0.7230	0.7025
Ours	0.2800	0.7234	0.7014

where $\mathcal{H}_t$ indexes the common available history at decision t. The scalar intercept a is unpenalized; the penalty coefficient is fixed at one under this summed-squared-error convention. Without history, the prediction is zero; parameters are not chosen using subsequent evaluation outcomes.

Same-parent historical mean. Let $\mathcal{H}_{t,p} \subseteq \mathcal{H}_t$ contain completed expansions of the current parent, across operations, and let $\bar{y}_t$ be the overall historical mean. We use one overall-mean pseudo-observation to shrink sparse parent-specific estimates:

$$h_t(p) = \frac{\sum_{i \in \mathcal{H}_{t,p}} y_i + \bar{y}_t}{|\mathcal{H}_{t,p}| + 1}. \tag{21}$$

This predictor falls back to the overall mean for an unseen parent and to zero when no history exists. Previous observations of the same parent were available for 1,394 of the 2,711 evaluated expansions (51.4%).

Kernel predictors. The context-kernel and root-kernel predictors have zero prior means and retain only k_{context} and k_{root}, respectively. Each recomputes its own posterior using the common historical targets and its retained kernel. The root kernel includes within-root similarity in parent performance; it is not merely a constant effect for each root identifier. The zero-mean GP with the full kernel retains context, root, and rating covariance, including $\sigma_\theta^2 r r'$. It removes the prior mean but does not remove all LLM assessment information. All three diagnostics retain the hyperparameters, noise level, and numerical regularization available to BayesianAgent at the corresponding decision. They do not independently refit hyperparameters or generate new evolutionary trajectories. They therefore differ from the online strategy ablations in Section 4.4.

C.3 Historical Calibration and Sensitivity

For each nonconstant predictor, we calibrate a current score h as $\widehat{y}_t(h) = a_t + b_t h$ using ordinary least squares on $\mathcal{H}_t$. The scalar coefficients a_t and b_t are the fitted intercept and slope, respectively. The calibration inputs are predictions made at those earlier decisions using their own available histories. We never substitute predictions of past training labels recomputed by the current model. The slope is unconstrained, allowing its direction to be learned from earlier feedback. No history gives a zero prediction; a single observation or constant historical score gives the historical target mean. For numerical stability, historical scores are treated as constant when their range is at most

Table 4: **Sensitivity to short calibration histories.** The full-sample calibration column retains all 2,711 expansions, including extreme extrapolations from short histories. The remaining columns use the common 2,381 expansions with at least twenty historical observations. Gain correlations use native scores, and all metrics receive equal task weights.

Predictor	Calibrated NRMSE ↓ All expansions	Gain Spearman ↑ History ≥ 20	Native NRMSE ↓ History ≥ 20	Calibrated NRMSE ↓ History ≥ 20
Zero gain	0.7702	N/A	0.7420	0.7420
Historical mean	0.7892	-0.0680	0.7435	0.7435
Parent score	0.7766	-0.3759	N/A	0.6907
LLM rating	0.7823	0.2084	0.7938	0.7329
Parent + rating regression	19.0373	0.2828	0.6926	0.6996
Same-parent historical mean	23.1426	0.2500	0.7060	0.7339
MCGS reward Q^{MCGS}	0.8029	0.0129	N/A	0.7477
Context-kernel predictor	10.7888	0.2438	0.6913	0.7062
Root-kernel predictor	14.7907	0.2760	0.6915	0.6941
Zero-mean GP, full kernel	15.7137	0.2721	0.6958	0.6937
Ours	0.7787	0.2706	0.6962	0.6924

$64\epsilon_{\mathrm{mach}} \max(1, \max_{i \in \mathcal{H}_t} |h_i|)$. Here ϵ_{mach} is float64 machine precision, distinct from the GP observation errors ϵ_i. We do not otherwise regularize or clip the calibration fit or its predictions. Zero gain and historical mean are retained as direct controls without an additional calibration fit.

Short histories can produce severe extrapolation even when the underlying predictor has modest error. On the uncertainty inequality, several predictors had nearly constant earlier forecasts but substantially different scores at histories of four or five observations. The resulting affine fits produced predicted normalized gains with magnitudes in the thousands. These errors are retained in the full-sample results in Table 4; they are not evidence of equally poor native predictions. The common subsets requiring at least eight or twenty observations contain 2,561 and 2,381 expansions, respectively. Both thresholds were fixed before examining the calibration results. They change the evaluation subset without changing earlier predictions or refitting them using future data.

Run-selection sensitivity. The primary prediction analysis uses the original runs rather than selecting runs using final task scores. To assess alignment with the main task-score tables, we also evaluate the run selection used for those tables. This selection replaces four runs, on hexagon packing, maximum-to-minimum distance ratio, minimum overlap, and sums and differences 2. The remaining eleven runs are shared, so the resulting 2,785 expansions do not constitute an independent replication. The replacements reached the experimental budget but ended with provider or cleanup failures, without an independent final reevaluation. We retain valid completed expansions whose predictions and outcomes can be matched under the same temporal rules. Table 5 reports all predictors under this alternative selection. The comparison with MCGS reward Q^{MCGS} remains favorable: gain Spearman is 0.3029 versus 0.0061, with higher correlation on 14 of 15 tasks.

C.4 Task-Level Results and Interpretation

Tables 6–8 report the original-run results by task, including every predictor in Table 3. The first two tables use all valid evaluated expansions; the calibrated table uses the common history-at-least-eight subset. Correlations describe realized gains among actions selected by BayesianAgent, not all alternatives that could have been executed. Within-trajectory dependence and overlapping information sources limit causal attribution to individual components. These comparisons do not establish a causal benefit from transferring evidence across candidates or the calibration of posterior uncertainty.

In the task-level tables, Z denotes zero gain; H, historical mean; P, parent score; L, LLM rating; PL, parent-plus-rating regression; and SP, same-parent mean. Q denotes MCGS reward Q^{MCGS}; C and R, the context- and root-kernel predictors; F, the zero-mean full-kernel GP; and O, BayesianAgent.

Table 5: **Sensitivity to the runs selected for the task-score tables.** This selection contains 2,785 valid expansions across the same fifteen tasks, sharing eleven runs with the primary analysis and replacing four. The final column uses its common subset with at least eight historical observations. The alternative selection is not an independent replication; extreme full-sample calibration errors are retained.

Predictor	Gain Spearman ↑ All expansions	Native NRMSE ↓ All expansions	Calibrated NRMSE ↓ All expansions	Calibrated NRMSE ↓ History ≥ 8
Zero gain	N/A	0.7901	0.7901	0.7907
Historical mean	-0.0760	0.7997	0.7997	0.7906
Parent score	-0.3821	N/A	0.7766	0.7224
LLM rating	0.2132	0.8330	0.7902	0.7798
Parent + rating regression	0.2917	0.7590	18.9607	0.7372
Same-parent historical mean	0.2294	0.7627	23.1484	0.8043
MCGS reward Q^{MCGS}	0.0061	N/A	0.8087	0.7991
Context-kernel predictor	0.2805	0.7400	10.7992	0.7554
Root-kernel predictor	0.2890	0.7392	14.6795	0.7478
Zero-mean GP, full kernel	0.2973	0.7461	15.6669	0.7435
Ours	0.3029	0.7471	0.7755	0.7449

Table 6: **Gain Spearman by task.** Native scores on all valid evaluated expansions; higher is better.

Task	Z	H	P	L	PL	SP	Q	C	R	F	O
Circle packing (rectangle)	N/A	-0.012	-0.469	0.275	0.441	0.383	-0.134	0.404	0.314	0.371	0.406
Circle packing (square)	N/A	-0.130	-0.479	0.348	0.392	0.339	-0.012	0.430	0.496	0.452	0.453
First autocorrelation	N/A	-0.125	-0.241	0.144	0.063	0.111	0.142	0.221	0.226	0.234	0.211
Heilbronn (convex)	N/A	-0.003	-0.542	0.381	0.469	0.380	-0.050	0.427	0.436	0.441	0.448
Heilbronn (triangle)	N/A	-0.018	-0.502	0.500	0.414	0.388	-0.088	0.418	0.427	0.467	0.466
Hexagon packing	N/A	0.035	-0.340	0.083	-0.010	0.143	-0.095	-0.044	0.242	0.282	0.252
Kissing number ($d = 11$)	N/A	-0.110	-0.204	0.095	-0.073	0.038	-0.053	-0.052	-0.210	-0.232	-0.186
Max/min distance ratio	N/A	-0.002	-0.482	0.146	0.308	0.230	-0.027	0.097	0.264	0.191	0.144
Minimum overlap	N/A	-0.029	-0.268	0.067	0.209	0.273	-0.121	0.297	0.245	0.294	0.286
Second autocorrelation	N/A	-0.101	-0.346	0.264	0.378	0.299	0.077	0.408	0.284	0.304	0.332
Sums and differences 1	N/A	-0.147	-0.157	0.027	0.117	0.057	-0.045	-0.026	0.214	0.170	0.184
Sums and differences 2	N/A	-0.155	-0.572	0.110	0.526	0.233	0.033	0.353	0.473	0.488	0.490
Third autocorrelation	N/A	-0.019	-0.262	0.098	0.108	0.137	0.087	0.108	0.156	0.182	0.162
Third autocorrelation (var.)	N/A	-0.258	-0.359	0.159	0.147	-0.011	-0.056	0.078	0.154	0.104	0.097
Uncertainty inequality	N/A	-0.106	-0.447	0.362	0.431	0.239	-0.081	0.404	0.430	0.453	0.456

D CONVERGENCE ANALYSIS

AlphaEvolve (Novikov et al., 2025) uses the kissing number problem for its ablation analysis. Following this choice, we select the 11-dimensional instance as a representative case study from our suite of 15 mathematical tasks. The objective is to maximize the number of non-overlapping unit spheres that can touch a central unit sphere. To complement this task-level view, we track the average rank across all 15 tasks: at each time, we rank the six methods by their best observed score on each task, then average these ranks with equal task weights. This aggregate captures relative performance without introducing a task-specific score normalization.

Figure 4(a) shows that BayesianAgent makes rapid early progress on the kissing number problem and achieves further gains as evolution proceeds. After two hours, it reaches a verifier score of 385, already exceeding the greedy policy's final score of 383. It finishes at 394, ahead of the variant without root covariance at 388, context-only at 349, and MLEvolve and rating-only at 348. Figure 4(b) shows a complementary trend across the full task suite. Although the methods exchange positions

Table 7: **Native NRMSE by task.** Direct gain predictions on all valid evaluated expansions; lower is better.

Task	Z	H	P	L	PL	SP	Q	C	R	F	O
Circle packing (rectangle)	1.854	1.877	N/A	1.878	1.821	1.805	N/A	1.761	1.830	1.811	1.808
Circle packing (square)	0.837	0.866	N/A	0.834	1.139	0.744	N/A	0.729	0.777	0.777	0.779
First autocorrelation	0.644	0.653	N/A	0.681	0.653	0.638	N/A	0.617	0.594	0.605	0.609
Heilbronn (convex)	0.973	0.921	N/A	1.022	0.839	0.851	N/A	0.856	0.846	0.856	0.856
Heilbronn (triangle)	0.539	0.542	N/A	0.571	0.484	0.478	N/A	0.491	0.509	0.510	0.505
Hexagon packing	0.427	0.569	N/A	0.473	0.525	0.525	N/A	0.420	0.352	0.362	0.355
Kissing number ($d = 11$)	0.770	0.759	N/A	0.860	0.782	0.813	N/A	0.782	0.778	0.801	0.803
Max/min distance ratio	0.480	0.484	N/A	0.482	0.518	0.452	N/A	0.475	0.475	0.478	0.480
Minimum overlap	0.868	0.877	N/A	0.913	0.798	0.786	N/A	0.805	0.792	0.787	0.791
Second autocorrelation	0.697	0.730	N/A	0.742	0.571	0.611	N/A	0.600	0.552	0.557	0.548
Sums and differences 1	0.056	0.056	N/A	0.250	0.054	0.055	N/A	0.056	0.066	0.057	0.071
Sums and differences 2	0.413	0.442	N/A	0.418	0.348	0.404	N/A	0.387	0.390	0.397	0.395
Third autocorrelation	0.914	0.925	N/A	0.950	0.943	1.059	N/A	0.920	0.918	0.919	0.923
Third autocorrelation (var.)	0.640	0.679	N/A	0.667	0.618	0.652	N/A	0.628	0.623	0.639	0.635
Uncertainty inequality	1.440	1.458	N/A	1.397	1.290	1.374	N/A	1.304	1.288	1.289	1.291

Table 8: **Calibrated NRMSE by task.** All predictors use the same evaluated expansions with at least eight historical observations; lower is better.

Task	Z	H	P	L	PL	SP	Q	C	R	F	O
Circle packing (rectangle)	1.880	1.885	1.833	1.894	1.838	1.858	1.888	1.809	1.904	1.823	1.855
Circle packing (square)	0.822	0.834	0.748	0.818	0.825	0.752	0.862	0.748	0.744	0.748	0.747
First autocorrelation	0.636	0.631	0.628	0.637	0.633	0.626	0.636	0.625	0.599	0.603	0.602
Heilbronn (convex)	0.991	0.935	0.840	0.892	0.867	0.876	0.939	0.876	0.861	0.863	0.860
Heilbronn (triangle)	0.539	0.531	0.482	0.465	0.482	0.481	0.534	0.495	0.495	0.484	0.478
Hexagon packing	0.198	0.239	0.122	0.239	0.248	0.259	0.307	0.198	0.094	0.109	0.083
Kissing number ($d = 11$)	0.759	0.734	0.735	0.734	0.740	0.757	0.736	0.737	0.742	0.739	0.752
Max/min distance ratio	0.494	0.497	0.529	0.504	0.536	0.530	0.499	0.501	0.499	0.502	0.501
Minimum overlap	0.843	0.842	0.765	0.857	0.767	0.777	0.842	0.811	0.766	0.757	0.762
Second autocorrelation	0.669	0.697	0.558	0.686	0.564	0.608	0.731	0.631	0.541	0.548	0.538
Sums and differences 1	0.052	0.051	0.050	0.051	0.052	0.052	0.051	0.051	0.050	0.050	0.051
Sums and differences 2	0.411	0.413	0.304	0.411	0.346	0.405	0.413	0.419	0.392	0.402	0.373
Third autocorrelation	0.930	0.940	0.942	0.958	0.942	1.194	0.942	0.943	0.955	0.940	0.941
Third autocorrelation (var.)	0.642	0.654	0.585	0.665	0.596	0.681	0.665	0.648	0.644	0.666	0.662
Uncertainty inequality	1.446	1.463	1.284	1.410	1.304	1.412	1.469	1.337	1.383	1.306	1.316

early in evolution, BayesianAgent maintains the best average rank from approximately five hours onward and finishes at 2.57, followed by context-only at 3.03 and the variant without root covariance at 3.17. Together, these trajectories show that BayesianAgent achieves the best final performance among the evaluated settings on the selected case study and the best final average rank across the 15-task suite. Execution-history differences relevant to interpreting these comparisons are documented in Appendix B.

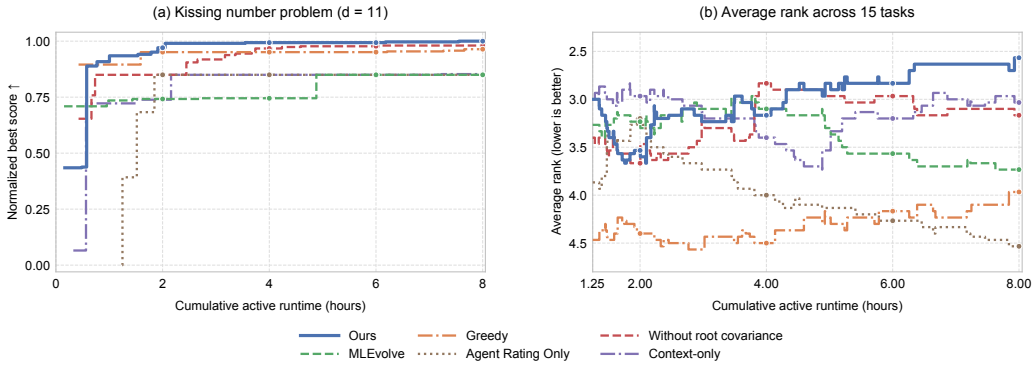

Figure 4: **Convergence behavior on a representative task and across the mathematical task suite. (a)** Best observed verifier score on the 11-dimensional kissing number problem, normalized as $(s - 88)/(394 - 88)$ using the minimum initial score and maximum final score across the six methods; higher is better. **(b)** Mean rank among all six methods across 15 tasks; lower ranks are better and appear higher on the axis. Ties within 10^{-10} in the original objective units receive average ranks. The rank curves begin at 1.25 hours, when all task–method pairs have a valid observation. Both panels use cumulative active runtime, with steps at recorded updates and circles at two-hour intervals.

Table 9: **Detailed AlphaEvolve scores for the ablation comparison.** Scores are the best verifier-evaluated values in the shared experiment snapshot. Arrows indicate the preferred direction; bold marks the best score among all seven methods, including ties. Average ranks are computed jointly over the seven methods using unrounded scores on all 15 tasks, with ties within 10^{-10} assigned mean ranks.

(a) BO method and ablations

Task		**Ours**	w/o root covariance	Context-only	w/o posterior sampling
Hexagon packing	↓	3.944824	4.000000	3.944206	4.037085
Circle packing (square)	↑	2.631092	2.619710	**2.633016**	2.630852
Circle packing (rectangle)	↑	2.359120	2.364233	2.357306	**2.364239**
Heilbronn (convex)	↑	**0.030891**	0.030872	0.030817	0.030701
Heilbronn (triangle)	↑	0.036413	0.036082	0.036513	0.036374
Kissing number ($d = 11$)	↑	**394**	388	349	380
Sums and differences 1	↑	1.110246	1.118775	**1.125678**	1.112391
Sums and differences 2	↑	1.120363	1.113774	**1.122167**	1.113046
Uncertainty inequality	↓	**0.352099**	**0.352099**	**0.352099**	**0.352099**
First autocorrelation	↓	**1.519079**	1.520947	1.519227	1.526182
Third autocorrelation (var.)	↓	1.485220	1.471575	1.522194	**1.465710**
Third autocorrelation	↓	1.464716	1.466098	1.471765	1.469785
Second autocorrelation	↑	0.890046	0.887309	0.869092	**0.902493**
Max-to-min distance ratio	↓	12.903948	**12.890436**	12.891695	12.894203
Minimum overlap	↓	**0.381114**	0.382023	0.382029	0.382141
Average rank	↓	**2.93**	3.53	3.40	4.07

(b) Full method and non-BO baselines

Task		**Ours**	MLEvolve	Greedy	Agent-rating
Hexagon packing	↓	3.944824	3.947878	**3.940341**	3.953391
Circle packing (square)	↑	2.631092	2.611866	2.629294	2.630106
Circle packing (rectangle)	↑	2.359120	2.348958	2.338491	2.356519
Heilbronn (convex)	↑	**0.030891**	0.030802	0.030769	0.030542
Heilbronn (triangle)	↑	0.036413	**0.036528**	0.036454	0.036235
Kissing number ($d = 11$)	↑	**394**	348	383	348
Sums and differences 1	↑	1.110246	1.109949	1.113723	1.122740
Sums and differences 2	↑	1.120363	1.118903	1.112679	1.113825
Uncertainty inequality	↓	**0.352099**	**0.352099**	**0.352099**	**0.352099**
First autocorrelation	↓	**1.519079**	1.532056	1.544900	1.544972
Third autocorrelation (var.)	↓	1.485220	1.476867	1.471381	1.510232
Third autocorrelation	↓	1.464716	**1.461759**	1.485251	1.482754
Second autocorrelation	↑	0.890046	0.868407	0.886555	0.882910
Max-to-min distance ratio	↓	12.903948	12.890873	12.890881	12.902929
Minimum overlap	↓	**0.381114**	0.381597	0.382622	0.382367
Average rank	↓	**2.93**	4.23	4.53	5.30

E Detailed AlphaEvolve Ablation Results

Table 9 gives the task-level scores underlying Figure 3. The two panels separate the BO method and its ablations from the non-BO baselines, with the full method repeated as a reference. All average ranks use the same seven-method comparison; the repeated reference is counted only once.

F SUPPLEMENTARY AUTOLAB EXPERIMENTS

Experimental setup. We extend the allocation comparison to seven tasks from AutoLab (Xu et al., 2026), covering image filtering, relational joins, concurrent key–value storage, attention and Fourier-transform kernels, instruction scheduling, and hashing. Candidate execution and evaluation use CPUs, and each run receives a 2-hour budget for agent iteration. All methods use Muse Spark 1.3 and three random seeds (42, 43, and 44).

Methods and reporting. We compare BayesianAgent, MLEvolve with its original MCGS allocation policy, parent-metric-greedy selection, and LLM-guided selection. LLM-guided selection asks the language model to choose a parent directly from summaries of eligible solutions, their measured performance and refinement histories, and the remaining budget. Each run is summarized by its best valid verifier reward within the budget, with higher values indicating better performance. Table 10 reports the mean and sample variance across the three seeds; the final row averages the task means with equal weight.

Table 10: **Supplementary AutoLab results on seven CPU tasks.** Each run uses a 2-hour agent-iteration budget. Entries are mean (sample variance) of verifier reward across three seeds; higher means are better. Bold marks the largest mean per task, including ties. The last row reports the equally weighted mean over the seven tasks.

Task	Ours	MLEvolve	Greedy	LLM-guided
Gaussian blur	0.6246 (0.000162)	0.6495 (0.001422)	**0.6601** (0.002848)	0.6548 (0.000083)
Hash join	**0.6775** (0.000026)	0.6626 (0.000045)	0.6676 (0.000074)	0.6733 (0.000186)
Concurrent key–value store	0.8180 (0.002441)	**0.8205** (0.003372)	0.7988 (0.008020)	0.7960 (0.005231)
Flash attention	**0.7337** (0.004179)	0.7202 (0.000778)	0.7037 (0.008756)	0.6603 (0.001205)
FFT (Rust)	**0.5847** (0.000007)	0.5737 (0.000020)	0.5629 (0.000158)	0.5716 (0.000844)
VLIW scheduling	**1.0000** (0.000000)	**1.0000** (0.000000)	**1.0000** (0.000000)	**1.0000** (0.000000)
SHA-256 throughput	0.1521 (0.000779)	**0.1566** (0.004813)	0.1133 (0.001267)	0.1119 (0.000218)
Seven-task mean	**0.6558**	0.6547	0.6438	0.6383

BayesianAgent obtains the highest seven-task mean of 0.6558, with a small difference from MLEvolve at 0.6547. It leads on hash join, flash attention, and FFT, while Greedy leads on Gaussian blur and MLEvolve leads on concurrent key–value storage and SHA-256 throughput. All four methods attain a reward of 1.0000 on VLIW scheduling.