GP-HARNESS: SEQUENTIAL BAYESIAN–EVOLUTIONARY OPTIMISATION OF LLM HARNESSES

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model agents rely on harnesses that organise planning, tool use, verification, and failure recovery around a fixed model. Automatic harness search can generate more candidates than can be evaluated on long-horizon tasks, making the choice of which candidates to evaluate a central challenge. We introduce GP-HARNESS, which combines evolutionary candidate generation with Bayesian optimisation to guide this choice. Starting from eight adapted open-source harnesses, programmatic mutation and recombination generate candidate pools. A Gaussian-process surrogate over graph features uses previous evaluations to predict candidate fitness and uncertainty, and expected improvement selects one candidate for evaluation in each round. Within each benchmark, all harnesses use a shared executor, with the task model, tool interfaces, evaluator, and per-task execution limits held fixed. With 20 new-candidate evaluations per run, GP-HARNESS achieves 64.6% mean success on a held-out 24-task Terminal-Bench 2.0 subset and a 43.3% mean resolved rate on the official 300-instance SWE-bench Lite test set across six search seeds, including runs that fall back to an initial harness. These results improve on the initial harness selected by its performance on the search tasks by 10.4 and 8.0 percentage points, respectively, and on budget-matched surrogate-free evolution by 4.9 and 4.6 points. Online ablations favour graph features over dimension-matched text features and uncertainty-aware acquisition over posterior-mean selection.

1 INTRODUCTION

For large language model (LLM) agents, the *harness* comprises the code and prompts that organise planning, tool use, context management, verification, and failure recovery. These choices shape how the model interacts with its environment, making harness design an optimisation target even when the model and available tools remain fixed (Yao et al., 2023; Yang et al., 2024). Software-engineering agents illustrate this design space through different procedures for locating relevant code, making edits, running tests, and responding to errors (Yang et al., 2024; Xia et al., 2025; Wang et al., 2025). Recent methods have begun to automate the search for better harnesses (Lee et al., 2026; Luo et al., 2026; Lin et al., 2026a; Zhang et al., 2026b). Assessing a proposed change, however, requires running the resulting agent through task environments, often involving many model calls and tool executions (Merrill et al., 2026; Jimenez et al., 2024). Candidate generation can therefore outpace the available evaluation budget.

Deciding which unexecuted harness warrants complete benchmark evaluation presents a fundamental research challenge: candidate performance is highly uncertain prior to execution, yet exhaustive testing across long-horizon benchmarks is computationally prohibitive. While recent automated agent-design pipelines recognise this evaluation bottleneck, existing strategies leave the selection problem fundamentally underconstrained. Existing approaches address different parts of this search problem. Execution feedback can guide subsequent proposals (Lee et al., 2026), evaluations on task subsets can screen candidates before full evaluation (Luo et al., 2026), and novelty filters can remove similar proposals before execution (Lange et al., 2026). Related agent-design work also uses learned performance predictors to guide search (Li et al., 2026). Our focus is the allocation of complete evaluations within an ongoing harness search. Previous evaluations provide candidate-level observations that can inform this decision, but new candidates remain uncertain until they are executed. We therefore study whether a probabilistic surrogate, updated from the evaluations collected

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.

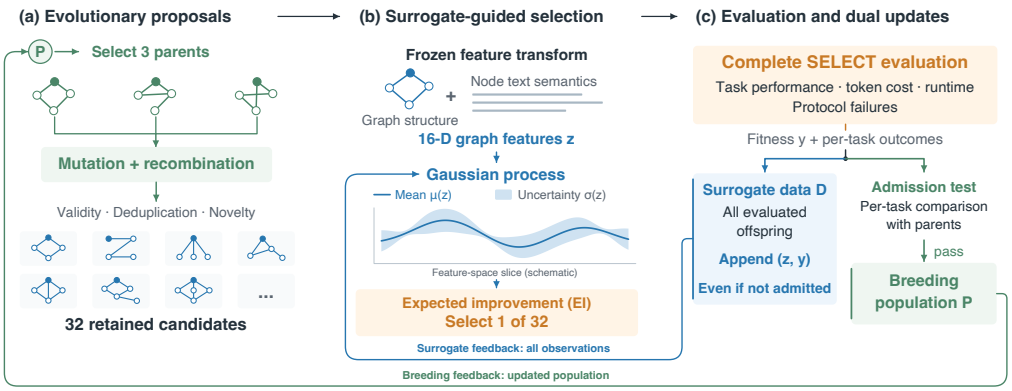


Figure 1: Overview of GP-HARNESS. (a) Three parents generate 32 filtered candidates through mutation and recombination. (b) A GP over frozen graph and text features guides EI to select one candidate for complete evaluation. (c) All evaluated offspring update the surrogate dataset, while only admitted offspring enter the breeding population. Blue and green feedback paths update the GP and parent selection, respectively.

during a run, can use predicted fitness and uncertainty to select promising candidates without first executing every proposal.

To address this allocation problem, we introduce GP-HARNESS, a framework that combines evolutionary candidate generation with sequential Bayesian optimisation in Figure 1. Programmatic mutation and recombination generate a pool of candidates from a breeding population without proposer-model calls. A Gaussian-process surrogate fitted to previous evaluations predicts each candidate's fitness and uncertainty, and expected improvement selects one candidate for a complete evaluation (Jones et al., 1998; Snoek et al., 2012). The observed result updates the surrogate before the next round, while offspring that meet the breeding criteria can also become parents. This allows every evaluated candidate to inform future selection, including those not retained for breeding.

The surrogate needs features that describe both the content and organisation of a harness. We use an attributed graph whose nodes describe workflow states, edges describe declared transitions, and attributes capture prompts, skills, and tool requirements. The workflow is supplied as instructions to a shared agent executor; a separate state-machine interpreter does not enforce its transitions. We compare graph features with features extracted from the serialised harness text, using the same feature dimension and GP configuration. Preprocessing transformations are fitted on fixed unlabelled harness corpora and frozen before search. This comparison assesses the complete representation pipelines, not the effect of graph topology in isolation.

We evaluate GP-HARNESS independently on Terminal-Bench 2.0 and SWE-bench Lite. Holding the task model (DeepSeek-V4-Flash-0731), execution interfaces, and per-task constraints strictly constant across methods, each search run receives a constrained budget of only 20 complete candidate evaluations. Under this restricted budget, GP-HARNESS achieves a 64.6% mean success rate on held-out Terminal-Bench 2.0 tasks and a 43.3% resolved rate on official SWE-bench Lite instances across six paired seeds, outperforming initial founder harnesses by 10.4 and 8.0 percentage points, respectively, and budget-matched surrogate-free evolution by 4.9 and 4.6 points. Online ablations favour graph features over dimension-matched text features and uncertainty-aware acquisition over posterior-mean selection on both benchmarks.

Our contributions are threefold:

- We develop GP-HARNESS, which uses a Gaussian-process surrogate to choose which evolution-generated harnesses to evaluate. Each evaluation informs later choices, even when the offspring is not retained for breeding.

- We compare graph and text representations under the same GP configuration, and test whether using predictive uncertainty improves candidate selection.

- We show that GP-guided search achieves higher mean held-out performance than the initial-harness baseline and surrogate-free evolution on two benchmarks under matched candidate-evaluation budgets, including runs that fall back to an initial harness.

2 Related Work

Harness optimisation. Prior work automates the design of the system around a fixed model at three granularities. Prompt optimisers evolve instructions through reflection or LLM-driven mutation (Agrawal et al., 2026; Zehle et al., 2025); workflow and agent-system optimisers search over code, topologies, and module compositions (Hu et al., 2025; Zhang et al., 2025b; Zhou et al., 2026; Zhang et al., 2025a; Shang et al., 2025); and recent methods edit complete coding-agent harnesses (Lee et al., 2026; Luo et al., 2026; Lin et al., 2026a; Zhang et al., 2026b) or evolve programs and agent scaffolds from an archive (Zhang et al., 2026a; Novikov et al., 2025; Lange et al., 2026). Across all three, candidates are proposed by a language model, and the evaluation budget is managed by evaluating every candidate, by screening or racing on cheap task subsets, or by filtering proposals for novelty. Where a learned predictor is used, it returns point scores from an in-context LLM, a neural encoder, or a value model (Shang et al., 2025; Trirat et al., 2026; Li et al., 2026). We instead treat evaluation allocation as the central decision: a programmatic generator proposes harnesses, and a Gaussian-process surrogate whose uncertainty enters the acquisition chooses which one to evaluate.

Bayesian optimisation for language-based search. Gaussian-process surrogates have reached language artefacts in three forms: selecting from a fixed prompt pool under multi-fidelity evaluation (Schneider et al., 2025); steering an LLM's proposals through the acquisition, over frozen embeddings of past solutions or over features the model itself elicits (Agarwal et al., 2025; Lin et al., 2026b); and pairing a Bayesian backend with an LLM that proposes or controls (Cissé et al., 2025; Brunzema et al., 2026). In each case the surrogate either picks from a static pool or shapes what is generated next; none decides which of several newly generated candidates receives an expensive evaluation. Two findings shape our design: LLMs used directly as optimisers ignore feedback and are outperformed by Gaussian processes (Gupta et al., 2025), and code embeddings fail to separate near-identical programs (Agarwal et al., 2025). GP-Harness therefore lets an evolutionary generator supply a fresh candidate frontier each round, scores it with a Gaussian process on frozen structural features of the harness, and selects a single harness by expected improvement; we compare graph-structured and text features at matched dimension.

3 Method

Proposing a new harness is cheap, whereas evaluating it requires running the agent through long-horizon tasks. We therefore cast harness search as Bayesian optimisation over a combinatorial space: a Gaussian-process surrogate decides which harness receives the next expensive evaluation, and a programmatic evolutionary generator supplies the finite candidate set over which the acquisition is maximised, as in Bayesian optimisation over graphs and other combinatorial domains (Oh et al., 2019; Ru et al., 2021). The generator makes no model calls and only determines which candidates are available.

A harness W specifies prompts, declared states and transitions, tools, skills, and recovery logic. FastHarness serialises this specification into instructions for a shared agent executor. Within each benchmark, the task model, tools, backend, evaluator, data partitions, and per-task limits are fixed; only the harness changes.

At round t, GP-Harness follows the closed loop

$$\mathcal{P}_{t-1} \xrightarrow{\text{generation}} \mathcal{C}_t \xrightarrow{\text{acquisition}} W_t \xrightarrow{\text{evaluation}} y_t \longrightarrow (\mathcal{D}_t, \mathcal{P}_t), \tag{1}$$

where $\mathcal{C}_t$ is the candidate set, $\mathcal{D}_t$ contains every evaluated harness and trains the surrogate, and $\mathcal{P}_t$ is the set of parents from which candidates are generated. Every evaluation updates $\mathcal{D}_t$, whereas only admitted offspring enter $\mathcal{P}_t$, so an offspring that is not retained as a parent still informs later acquisition decisions.

3.1 Surrogate Model and Acquisition

The N_0 founders initialise both $\mathcal{P}_0$ and $\mathcal{D}_0 = \{(\psi_r(W), y(W)) : W \in \mathcal{P}_0\}$. At each subsequent round, the generator produces a candidate set $\mathcal{C}_t = \{W_{t,1}, \ldots, W_{t,M}\}$, of which only one receives a complete SELECT evaluation. We model latent SELECT fitness with a Gaussian process,

$$f \sim \mathcal{GP}(m, k_\theta), \qquad y_i = f(z_i) + \epsilon_i, \qquad \epsilon_i \sim \mathcal{N}(0, \sigma_n^2), \tag{2}$$

with a constant mean and an isotropic Matérn-$5/2$ kernel, giving posterior mean μ_{t-1} and standard deviation σ_{t-1}. Our default acquisition is expected improvement (EI):

$$\begin{aligned} W_t &\in \arg\max_{W \in \mathcal{C}_t} \mathbb{E}\big[(f(z_W) - y_{t-1}^{\max} - \xi)_+ \mid \mathcal{D}_{t-1}\big], \\ \mathcal{D}_t &= \mathcal{D}_{t-1} \cup \{(\psi_r(W_t), y(W_t))\}. \end{aligned} \tag{3}$$

The GP is refitted after each evaluation, before the next candidate set is scored. Kernel fitting, closed-form EI, and the UCB and posterior-mean variants are given in Appendix A.6.

3.2 Frozen Harness Representations

The surrogate is useful only if a few observed harnesses can inform predictions about unseen candidates. We map each harness to a fixed representation $z_W = \psi_r(W) \in \mathbb{R}^d$ and compare $r \in \{\mathrm{emb}, \mathrm{graph}\}$.

Text representation. We deterministically serialise the complete harness, encode it with a fixed pre-trained text encoder, $x_W = \phi_{\mathrm{text}}(\mathrm{serialise}(W))$, and apply ℓ_2 normalisation, standardisation, and PCA:

$$e_W = \frac{x_W}{\|x_W\|_2}, \qquad z_W^{\mathrm{emb}} = P_{\mathrm{emb}} \mathcal{N}_{\mathrm{emb}}(e_W) \in \mathbb{R}^d. \tag{4}$$

Graph representation. We represent the declared states and transitions as an attributed directed graph whose raw descriptor g_W combines directed Weisfeiler–Lehman features (Shervashidze et al., 2011), propagated state-prompt embeddings, pooled skill and tool embeddings, and global workflow statistics:

$$z_W^{\mathrm{graph}} = P_{\mathrm{graph}} \mathcal{N}_{\mathrm{graph}}(g_W) \in \mathbb{R}^d. \tag{5}$$

Both routes share the output dimension and surrogate family. Standardisation and PCA are fitted once on a fixed unlabelled harness corpus before any labelled evaluation and then frozen with the encoder and feature extractor, so no SELECT, VALID, or TEST outcome can alter the representation space. The comparison concerns complete representation pipelines rather than topology alone (Appendix A.5).

3.3 Candidate Generation

At round t, fitness-ranked parents in $\mathcal{P}_{t-1}$ produce proposals through mutation, which may alter prompts, states, transitions, tools, skills, recovery, or termination logic, and recombination, which inherits components from multiple parents. Generation continues until M proposals pass fixed validity, duplicate, and novelty checks; these form $\mathcal{C}_t$. After evaluation, W_t is compared with its primary parent on paired SELECT task outcomes, and the admission decision $B_t(W_t) \in \{0, 1\}$ updates the parent set:

$$\mathcal{P}_t = \begin{cases} \mathcal{P}_{t-1} \cup \{W_t\}, & B_t(W_t) = 1, \\ \mathcal{P}_{t-1}, & B_t(W_t) = 0. \end{cases} \tag{6}$$

Admission controls where future candidates are generated; the surrogate learns from every evaluation and alone decides which candidate is evaluated. Admission and novelty rules are detailed in Appendices A.4 and C.1.

3.4 Search Objective and Evaluation Protocol

For split $s \in \{\mathrm{sel}, \mathrm{val}, \mathrm{test}\}$, let $R_s(W)$ denote the mean task score, with repetitions averaged within tasks before tasks are weighted equally. Search and formal confirmation use

$$\widetilde{F}_s(W) = R_s(W) - \lambda_c C_s(W) - \lambda_t T_s(W) - \lambda_f Q_s(W), \tag{7}$$

where C_s, T_s, and Q_s are normalised token use, runtime, and protocol-failure rate, with all constants fixed before search. Each harness contributes one surrogate label $y(W) = \widetilde{F}_{\mathrm{sel}}(W)$ from its first complete SELECT evaluation.

SELECT is the only split used during optimisation. Once the budget is exhausted, the offspring with highest SELECT fitness is frozen. It is promoted only if it strictly exceeds the strongest SELECT founder, exceeds every founder on penalised VALID fitness, and has a saved passing novelty report; otherwise the SELECT-best founder is returned. VALID cannot alter any search state, and TEST is opened only after all seed-level outputs are fixed and reports the unpenalised score R_{test}. Proposition 2 explains why confirmation is necessary; details are in Appendix A.8.

3.5 Surrogate Guidance Helps Through Ranking Accuracy and Uncertainty

Evaluating every generated candidate would cost MG complete evaluations over G rounds; GP-Harness spends G. We ask how much a single surrogate-guided choice per round recovers, separating an exploitation component, governed by how well the surrogate ranks the candidate set, from an information component contributed by posterior uncertainty. Let $e_M = \mathbb{E}[\max_{i \le M} Z_i]$ for independent standard normal Z_i, so that $e_3 \approx 0.85$ and $e_{32} \approx 2.07$. Proofs are given in Appendix B.

Proposition 1. *Model the M candidates in $\mathcal{C}_t$ as independent draws for which latent fitness $f(z_W)$ and a selection score $s(W)$ are jointly Gaussian with correlation r_t, where f has standard deviation σ_t across the candidate set. Let $W_t^s \in \arg\max_{W \in \mathcal{C}_t} s(W)$ and let W_t^u be drawn uniformly from $\mathcal{C}_t$. Then*

$$\mathbb{E}\big[f(z_{W_t^s})\big] - \mathbb{E}\big[f(z_{W_t^u})\big] = r_t \, \sigma_t \, e_M. \tag{8}$$

The gain is linear in surrogate accuracy and candidate-set spread but grows only as $\sqrt{2 \log M}$ in the pool size. Representations act through r_t, novelty checks act through σ_t, and posterior-mean selection is the case $s = \mu_{t-1}$.

Corollary 1. *Suppose each candidate descends from one of P parents and $f(z_W) = b_{p(W)} + w_W$, with independent parent effects $b_p \sim \mathcal{N}(m, \sigma_B^2)$ and within-parent effects $w_W \sim \mathcal{N}(0, \sigma_W^2)$. Any selection rule independent of $\{w_W\}$ improves on uniform selection by at most $\sigma_B e_P$. Since $\sigma_B \le \sigma_t$, a candidate-level score whose gain is given by Eq. equation 8 outperforms every such rule whenever $r_t > e_P / e_M$.*

Surrogate-free evolution, which ranks candidates by primary-parent fitness, is such a rule. With $P = 3$ and $M = 32$ the threshold is $e_3/e_{32} \approx 0.41$, and it is lower when within-parent variation is substantial. The graph GP's prequential rank correlations, 0.48 and 0.44 (Table 3), lie above this level, although within one standard error; being measured on acquisition-selected candidates against noisy labels, they likely understate r_t. Posterior-mean selection realises only the exploitation component, so the further gain of EI in Table 2 can be read as the value of posterior uncertainty. Its weak association with realised error (Table 3) suggests that this value comes from directing evaluations towards unexplored candidates rather than from calibrated predictions; a regret bound for the UCB variant, which requires calibration, is given in Appendix B.

The final recommendation selects the best of G noisy labels and is subject to the same order-statistic effect.

Proposition 2. *Let $y_i = f_i + \epsilon_i$ for $i = 1, \dots, G$, with independent $f_i \sim \mathcal{N}(m, \sigma_f^2)$ and $\epsilon_i \sim \mathcal{N}(0, \sigma_\epsilon^2)$. For $i^\star \in \arg\max_i y_i$,*

$$\mathbb{E}[y_{i^\star}] - \mathbb{E}[f_{i^\star}] = \frac{\sigma_\epsilon^2}{\sqrt{\sigma_f^2 + \sigma_\epsilon^2}} \, e_G. \tag{9}$$

Offspring labels use one execution per task and founder labels three, so the SELECT-best offspring carries more optimism than the SELECT-best founder and the SELECT eligibility test favours offspring. VALID confirmation removes this advantage: the frozen offspring's VALID estimate is unbiased, while the founder side retains a maximum-of-eight optimism, making the comparison conservative.

4 EXPERIMENTS

We evaluate GP-HARNESS on Terminal-Bench 2.0 and SWE-bench Lite under a fixed budget of 20 complete new-candidate SELECT evaluations per run, starting from eight adapted open-source founders per benchmark. Section 4.1 evaluates the workflow returned by the full search-and-confirmation procedure; Section 4.2 diagnoses its search mechanisms in relation to Section 3.5. Section 4.3 summarises the protocol and scope, with full details in Appendix A.

4.1 HELD-OUT PERFORMANCE

Table 1 compares complete search-and-confirmation procedures over six paired seeds; a run whose offspring fails confirmation contributes its founder fallback. GP-HARNESS reaches 64.6% success on the held-out Terminal-Bench 2.0 subset and 43.3% resolved on SWE-bench Lite, 10.4 and 8.0 points above the SELECT-selected founder. It promotes an evolved offspring in five of six runs on each benchmark (Appendix C.1). Including fallback runs measures the outcome of the procedure as a whole, rather than conditioning the comparison on successful promotion.

Table 1: Held-out TEST performance under a matched candidate-evaluation budget. Search methods receive 20 complete new-candidate SELECT evaluations per run. Entries are means ± between-seed standard errors over six paired search seeds (Appendix A.10); a run that returns its founder contributes the founder's TEST score. TB2 uses a fixed 24-task held-out subset, and SWE-bench Lite uses the official 300-instance test set. Prior methods are evaluated as adaptations to the shared FastHarness specification and executor. Bold and underlining indicate the highest and second-highest means within each benchmark.

Method	Candidate evals.	TB2 held-out success (%)	SWE-bench Lite resolved (%)
SELECT-selected founder	0	54.2± 5.9	35.3± 2.1
Random offspring selection	20	56.9± 6.3	36.8± 3.1
Surrogate-free evolution	20	59.7± 6.8	38.7± 3.0
Meta-Harness	20	61.1± 6.8	40.5± 2.9
HarnessBank	20	<u>61.8</u>± 4.9	<u>41.2</u>± 3.2
AHE	20	59.7± 9.0	40.0± 3.5
HarnessCompass	20	61.1± 7.3	41.0± 3.3
GP-HARNESS	20	**64.6**± 4.4	**43.3**± 3.1

The internal controls show the same ordering of mean performance on both benchmarks: founder < random offspring selection < surrogate-free evolution < GP-HARNESS. Random selection adds 2.7 and 1.5 points over the founder, parent-fitness ranking a further 2.8 and 1.9, and candidate-level GP guidance a further 4.9 and 4.6. Because the search controls share the generator and evaluation budget, these comparisons assess candidate-selection policies together with the subsequent trajectories they induce. Among prior-method adaptations, HarnessBank is strongest; GP-HARNESS exceeds it by 2.8 and 2.1 points. The between-seed standard errors describe uncertainty in each method's mean, not in the paired difference between methods. We therefore interpret the reported ordering descriptively; the marginal errors alone do not establish the significance of the margin over HarnessBank. All prior methods are evaluated as FastHarness adaptations under the shared protocol rather than as their native systems (Section 4.3).

4.2 MECHANISM ANALYSIS

The variants in Table 2 share founders, budget, generator, pool size, and proposal-stage filters, but follow separate online trajectories. SELECT reports the best offspring's penalised fitness, the acquisition objective, and is subject to in-sample selection bias. Diagnostic VALID evaluates that same offspring on separate tasks using raw scores. It provides an out-of-sample check, but is not numerically comparable to SELECT because only SELECT includes penalties.

Learned allocation and uncertainty. GP-guided search has higher SELECT and VALID means than random offspring selection and surrogate-free evolution on both benchmarks. With graph fea-

Table 2: Online ablations of harness representation and candidate selection. Entries are means ± between-seed standard errors over three paired search seeds (the first three of the six main seeds), with 20 complete new-candidate SELECT evaluations per run. SELECT reports the penalised SELECT fitness of each run's best offspring; VALID reports the raw task score (%) of that same offspring on validation tasks. All runs are included regardless of promotion, without founder replacement. Bold indicates the highest mean in each column.

	Terminal-Bench 2.0		SWE-bench Lite	
Variant	SELECT	VALID	SELECT	VALID
GP-HARNESS (Graph + EI)	**.701**± .033	**66.7**± 6.7	**.438**± .030	**45.5**± 6.1
Text + EI	.667± .041	61.1± 6.9	.401± .033	40.4± 6.3
Graph + UCB	.682± .036	63.3± 8.8	.417± .031	42.4± 6.1
Graph + posterior mean	.658± .030	60.0± 5.8	.392± .029	39.4± 6.1
Surrogate-free evolution	.644± .034	57.8± 6.9	.381± .030	38.4± 7.6
Random offspring selection	.622± .039	54.4± 6.9	.365± .035	36.4± 6.1

Table 3: Prequential surrogate diagnostics on the six GP-HARNESS runs per benchmark. Before each selected candidate is evaluated, a graph GP and a text GP are refitted on the same history and predict that candidate. ρ_{rank}: Spearman correlation between predicted mean and observed SELECT fitness; NLPD: negative log-predictive density (lower is better); $\rho_{\mathrm{unc}}^{\mathrm{latent}}$: Spearman correlation between latent posterior standard deviation and absolute error. Metrics are computed per run and reported as means ± between-seed standard errors.

Benchmark	Predictor	$\rho_{\mathrm{rank}} \uparrow$	NLPD↓	$\rho_{\mathrm{unc}}^{\mathrm{latent}}$
Terminal-Bench 2.0	Graph GP	**0.48**± 0.09	−**1.01**± 0.13	0.16± 0.14
	Text GP	0.39± 0.10	−0.91± 0.14	0.09± 0.11
SWE-bench Lite	Graph GP	**0.44**± 0.08	−**1.19**± 0.12	0.14± 0.09
	Text GP	0.36± 0.09	−1.10± 0.13	0.10± 0.10

tures fixed, both uncertainty-aware acquisitions achieve higher VALID means than posterior-mean selection: EI by 6.7 and 6.1 points, and UCB by 3.3 and 3.0. The posterior-mean variant retains learned fitness predictions but removes the explicit uncertainty term from candidate selection. The six variants have the same ordering of means in all four columns, across two independently searched benchmarks and both measures. With only three seeds, however, this agreement is descriptive evidence, not a substitute for uncertainty estimates of paired differences. The results favour uncertainty-aware selection in these runs, but do not establish a statistically reliable advantage of EI over UCB.

Structured representation. Dimension-matched text features lower diagnostic VALID by 5.6 and 5.1 points, with the same direction of change in SELECT. Both routes feed 16-dimensional inputs to the same GP, but do not retain identical information: the text encoder truncates long serialisations, whereas the graph route encodes each state separately (Section 4.3). The comparison therefore concerns complete encoding pipelines rather than topology alone; the following diagnostics compare their prediction quality on shared search trajectories.

Surrogate diagnostics. The online ablations entangle representation effects with the different candidate pools each variant realises. Table 3 removes this confound without new benchmark executions: before each selected candidate's outcome enters the surrogate dataset, a graph GP and a text GP refitted on the same history predict the same candidate. Graph features rank the selected candidates better and attain lower NLPD on both benchmarks; each gap lies within about one between-seed standard error, but the direction agrees across metrics and benchmarks and matches the online ablation. Predictive uncertainty is only weakly associated with realised error (0.16 and 0.14 for the graph GP, not distinguishable from zero at six seeds), so the uncertainty term acts mainly by directing evaluations towards unexplored candidates rather than by calibrating individual predictions.

Search dynamics. Figure 2 shows the running best offspring SELECT fitness for the same three runs used in Table 2. Graph + EI does not lead throughout the search, but achieves the highest final mean on both benchmarks. The curves contain plateaus and changes in the ordering of methods, indicating that endpoint performance alone does not describe their behaviour at smaller budgets. A plateau in a run's curve means that no newly evaluated candidate improves its best observed

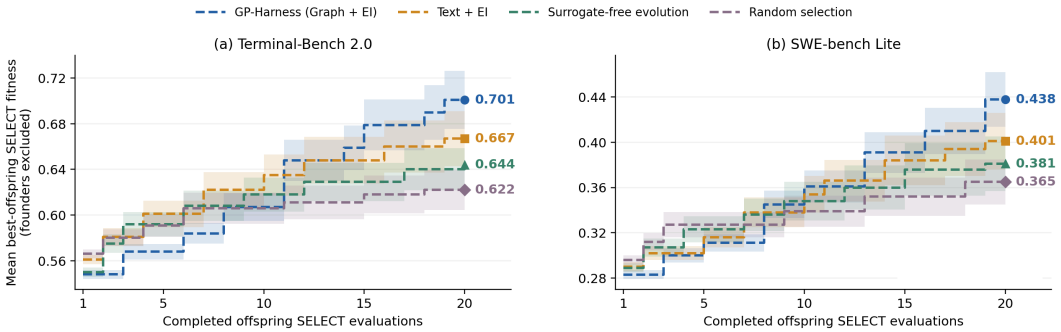


Figure 2: Search trajectories on Terminal-Bench 2.0 and SWE-bench Lite. For each run, we compute the highest offspring penalised SELECT fitness observed after each completed candidate evaluation, excluding founders. Lines show means across the same three paired search runs used for the corresponding variants in Table 2; shaded bands show ± 1 between-seed standard error. At evaluation 20, both the means and standard errors match the corresponding SELECT entries in Table 2.

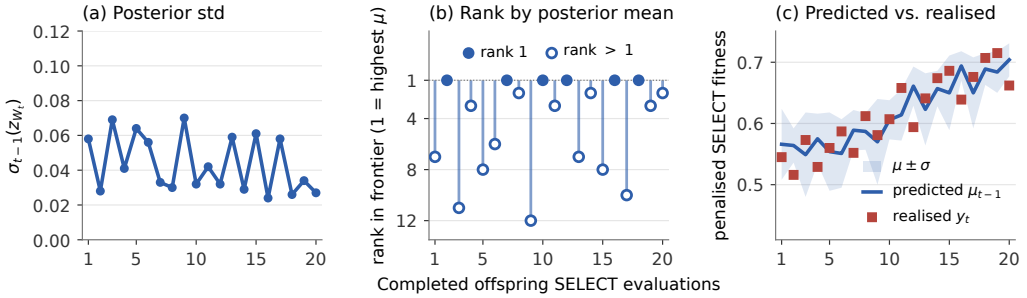


Figure 3: Acquisition behaviour of GP-HARNESS (Graph + EI) on Terminal-Bench 2.0 for one of the three search runs of Table 2 (seed 2). (a) Pre-evaluation posterior standard deviation of the selected candidate's latent fitness, excluding observation noise. (b) Rank of the selected candidate by posterior mean among the 32 candidates; filled markers denote the highest-predicted-mean candidate. (c) Pre-evaluation predicted mean with its $\pm\sigma$ band against the observed penalised SELECT fitness of the selected candidate. The band in (c) shows one posterior standard deviation for latent fitness; it excludes observation noise and is not an observation-predictive interval. No between-seed statistics are shown.

offspring fitness; it does not mean that no candidate is evaluated or that the surrogate receives no new information. These trajectories characterise the evaluated 20-candidate setting; they do not establish a minimum budget required for surrogate guidance.

Acquisition behaviour. Figure 3 examines candidate selection in one Terminal-Bench 2.0 run (seed 2) from Table 2. In the displayed trajectory, EI selects both highest-predicted-mean candidates with posterior standard deviations around 0.03 and lower-ranked candidates with larger standard deviations around 0.06. These selections occur throughout the budget rather than following a strict transition from exploration to exploitation. The observed fitness values describe individual candidate evaluations; their cumulative maximum gives this run's contribution to Figure 2, not the three-run mean. Rank 1 indicates agreement with posterior-mean selection, but does not imply that uncertainty played no role in the EI decision. Likewise, the latent posterior standard deviation is distinct from the predictive uncertainty of a noisy observation. This single-run view illustrates selection behaviour; it does not establish calibration or isolate the causal benefit of exploration.

4.3 Experimental Protocol and Scope

Benchmarks and partitions. Terminal-Bench 2.0 (Merrill et al., 2026) uses fixed partitions of 10 SELECT, 10 VALID, and 24 TEST tasks; SWE-bench Lite (Jimenez et al., 2024) splits its 23 development instances by repository into 12 SELECT and 11 VALID, with the official 300-instance test set as TEST. Partitions were fixed before any search. Founders are recalibrated under the experimental task model rather than taken from published leaderboards. SELECT supplies the observations used during search, VALID is used for confirmation and diagnostic evaluation, and TEST measures the held-out performance of the final returned workflow.

Matched search conditions. For the main comparison, all seven search methods use DeepSeek-V4-Flash-0731 as the task model with common tools, execution limits, partitions, founders, and six paired seeds. GP-Harness and the internal controls generate candidates with a programmatic genetic generator that operates directly on the harness specification and makes no model calls; all-mpnet-base-v2 is used only for representation features, and its fixed input length truncates long serialisations of the full harness. Founder calibration, parent comparisons, proposal generation, screening, VALID, and TEST fall outside the 20-evaluation cap and are recorded separately (Appendix A.9). Founder SELECT uses three executions per task, offspring SELECT one, VALID three, and TEST one; search and formal confirmation use penalised fitness, while diagnostic VALID and TEST report raw scores. Our method and controls use three fitness-ranked parents and 32 retained candidates per round; the default GP uses 16-dimensional graph features and EI with $\xi = 0.01$, with standardisation and PCA fitted on 64 unlabelled harnesses and then frozen.

Baselines. The founder baseline returns the SELECT-selected founder without search. Random offspring selection keeps our parent selection, generator, pool size, and filters but selects uniformly from the frontier; surrogate-free evolution ranks candidates by their primary parent's SELECT fitness without candidate-level predictions. Meta-Harness (Lee et al., 2026), HarnessBank (Luo et al., 2026), AHE (Lin et al., 2026a), and HarnessCompass (Zhang et al., 2026b) are adapted to FastHarness and retain their own proposal, feedback, and screening mechanisms, including proposer model calls; our 32-candidate pool and filters are not imposed on them (Appendix A.7).

Scope and limitations. Results concern the harness space exposed by the FastHarness representation and generator under a single task model. Six main and three ablation seeds provide limited information about run-to-run variation, particularly for small differences between methods. The graph–text comparison does not isolate topology. The main comparison matches candidate evaluations rather than total compute. Our programmatic generator makes no proposer-model calls, whereas the adapted prior methods incur such calls outside the candidate-evaluation cap. Programmatic generation and feature extraction still require computation; screening and parent-comparison executions are also recorded separately (Appendix A.9). The Terminal-Bench result uses a 24-task subset with a single TEST execution per task. The search trajectories describe the evaluated budget (Figure 2); they do not establish how the relative performance of the methods changes at larger budgets.

5 Conclusion

We introduced GP-Harness, a framework for optimising agent harnesses around a fixed task model by combining evolutionary candidate generation with surrogate-guided evaluation selection. Under a matched candidate-evaluation budget, independent searches on a held-out Terminal-Bench 2.0 subset and SWE-bench Lite improve mean held-out performance over the founder baseline and surrogate-free evolution. Online ablations favour attributed graph representations and uncertainty-aware acquisition in the evaluated settings. These findings highlight the value of selecting which harnesses to evaluate, alongside generating new designs. The conclusions remain specific to the evaluated workflows, execution backend, and task partitions; broader evaluations and component-level interventions are needed to assess generality and identify which structural changes drive the gains.

AI USE STATEMENT

We used generative AI tools to assist with writing and improve the clarity and readability of the manuscript. No generative AI tools were used in conducting the research reported in this paper. The authors reviewed all AI-assisted content and take full responsibility for the final content of this work, including all text, claims, results, and artifacts.

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 *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=aVfDR17xDV.

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 *The Fourteenth International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/0e9e708b6f48e14fd0ac29e167413f76-Abstract-Conference.html.

Paul Brunzema, Louis Tiao, Nhat Le, Kevin De Angeli, Yao Xuan, and Djordje Gligorijevic. Agentic bayesian optimization through surrogate-augmented autoresearch, 2026. URL https://arxiv.org/abs/2608.00316. arXiv:2608.00316.

Abdoulatif Cissé, Xenophon Evangelopoulos, Vladimir V. Gusev, and Andrew I. Cooper. Language-based bayesian optimization research assistant (BORA). In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence*, 2025. doi: 10.24963/ijcai.2025/553. URL https://doi.org/10.24963/ijcai.2025/553.

Herbert A. David and Haikady N. Nagaraja. *Order Statistics*. Wiley, 3 edition, 2003.

Rushil Gupta, Jason Hartford, and Bang Liu. LLMs for bayesian optimization in scientific domains: Are we there yet?, 2025. URL https://arxiv.org/abs/2509.21403. arXiv:2509.21403.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=D01WR1yVW2.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

Donald R. Jones, Matthias Schonlau, and William J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492, 1998. doi: 10.1023/A:1008306431147. URL https://doi.org/10.1023/A:1008306431147.

Robert Tjarko Lange, Yuki Imajuku, and Edoardo Cetin. ShinkaEvolve: Towards open-ended and sample-efficient program evolution. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2509.19349.

Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. Meta-Harness: End-to-end optimization of model harnesses, 2026. URL https://arxiv.org/abs/2603.28052. arXiv:2603.28052.

Yu Li, Lehui Li, Zhihao Wu, Qingmin Liao, Jianye Hao, Kun Shao, and Fengli Xu. AgentSwift: Efficient LLM agent design via value-guided hierarchical search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 31843–31851, 2026. doi: 10.1609/aaai.v40i38.40453. URL https://doi.org/10.1609/aaai.v40i38.40453.

Jiahang Lin, Shichun Liu, Chengjun Pan, Lizhi Lin, Shihan Dou, Zhiheng Xi, Xuanjing Huang, Hang Yan, Zhenhua Han, Tao Gui, and Yu-Gang Jiang. Agentic Harness Engineering: Observability-driven automatic evolution of coding-agent harnesses, 2026a. URL https://arxiv.org/abs/2604.25850. arXiv:2604.25850.

Zhiyuan Jerry Lin, Benjamin Letham, Samuel Dooley, Maximilian Balandat, and Eytan Bakshy. Embedding by elicitation: Dynamic representations for bayesian optimization of system prompts, 2026b. URL https://arxiv.org/abs/2605.19093. arXiv:2605.19093.

Xiaotian Luo, Dizhan Xue, Fengxingyu Wang, Chuanrui Hu, and Yafeng Deng. HarnessBank: Semantic gene-bank search with gated verification for agent-harness self-evolution, 2026. URL https://arxiv.org/abs/2607.13683. arXiv:2607.13683.

Mike A. Merrill, Alexander G. Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Yeon Shin, Thomas Walshe, E. Kelly Buchanan, et al. Terminal-Bench: Benchmarking agents on hard, realistic tasks in command line interfaces. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=a7Qa4CcHak.

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, 2025. URL https://arxiv.org/abs/2506.13131. arXiv:2506.13131.

Changyong Oh, Jakub M. Tomczak, Efstratios Gavves, and Max Welling. Combinatorial Bayesian optimization using the graph cartesian product. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

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

Lennart Schneider, Martin Wistuba, Aaron Klein, Jacek Golebiowski, Giovanni Zappella, and Felice Antonio Merra. Hyperband-based bayesian optimization for black-box prompt selection. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 53413–53438, 2025. URL https://proceedings.mlr.press/v267/schneider25b.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 *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=mPdmDYIQ7f.

Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M. Borgwardt. Weisfeiler–Lehman graph kernels. *Journal of Machine Learning Research*, 12:2539–2561, 2011. URL https://www.jmlr.org/papers/v12/shervashidze11a.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://proceedings.neurips.cc/paper/2012/hash/05311655a15b75fab86956663e1819cd-Abstract.html.

Niranjan Srinivas, Andreas Krause, Sham M. Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proceedings of the 27th International Conference on Machine Learning*, pp. 1015–1022, 2010.

Patara Trirat, Wonyong Jeong, and Sung Ju Hwang. Multi-view encoders for performance prediction in LLM-based agentic workflows. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=7oeKDZsmWp.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Yanjun Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, Robert Brennan, Hao Peng, Heng Ji, and Graham Neubig. OpenHands: An open platform for AI software developers as generalist agents. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2407.16741.

Chunqiu Steven Xia, Yinlin Deng, Soren Dunn, and Lingming Zhang. Demystifying LLM-based software engineering agents. *Proceedings of the ACM on Software Engineering*, 2(FSE):801–824, 2025. doi: 10.1145/3715754. URL https://doi.org/10.1145/3715754.

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.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Tom Zehle, Moritz Schlager, Timo Heiß, and Matthias Feurer. CAPO: Cost-aware prompt optimization. In *Proceedings of the Fourth International Conference on Automated Machine Learning*, volume 293 of *Proceedings of Machine Learning Research*, 2025. URL https://proceedings.mlr.press/v293/zehle25a.html.

Guibin Zhang, Luyang Niu, Junfeng Fang, Kun Wang, Lei Bai, and Xiang Wang. Multi-agent architecture search via agentic supernet. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, 2025a. URL https://arxiv.org/abs/2502.04180.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin Gödel Machine: Open-ended evolution of self-improving agents. In *The Fourteenth International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=pUpzQZTvGY.

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 *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=z5uVAKwmjf.

Luan Zhang, Ruochen Zhou, Dandan Song, Zhengyu Chen, Yuhang Tian, Jun Yang, Huipeng Ma, Chenhao Li, Guangyuan Feng, Xudong Li, Yizhou Jin, and Yan Xu. HarnessCompass: Guiding automatic harness evolution toward generalizable and effective agent harnesses, 2026b. URL https://arxiv.org/abs/2608.01918. arXiv:2608.01918.

Han Zhou, Xingchen Wan, Ruoxi Sun, Hamid Palangi, Shariq Iqbal, Ivan Vulić, Anna Korhonen, and Sercan Ö. Arık. Multi-agent design: Optimizing agents with better prompts and topologies. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2502.02533.

Table 4: Evaluation configuration for the main comparison and component ablations. A repetition is one execution of one task for one harness. Diagnostic VALID evaluations use the same task set and repetition count as formal VALID confirmation.

Setting	Terminal-Bench 2.0	SWE-bench Lite
SELECT tasks	10	12
VALID tasks	10	11
TEST tasks	24	300
Primary TEST metric	Success	Resolved
Founder SELECT repetitions	3	3
Candidate SELECT repetitions	1	1
VALID repetitions per task	3	3
TEST repetitions per task/output	1	1
Main paired search seeds	6	6
Online ablation seeds	3	3

A Experimental Details

This appendix specifies the benchmark partitions, founder populations, harness representations, search configuration, and evaluation procedures used in Section 4. Terminal-Bench 2.0 and SWE-bench Lite define independent optimisation problems, each with its own founders, observations, feature transformation, surrogate, search trajectory, and final outputs. Fitness labels and fitted transformations are not transferred between benchmarks.

A *candidate evaluation* is a complete evaluation of one new harness on the prescribed SELECT batch. We use *offspring* for candidates produced by evolutionary methods. This distinction allows the same candidate-budget definition to cover both evolutionary and agentic harness optimisers. For the final-selection rule in Section 3.4, let $W^{\star}_{\mathrm{off}}$ denote the SELECT-frozen offspring, $W^{\star}_{\mathrm{founder},s}$ the founder maximising $\widetilde{F}_s$ on split s, and $W^{\star}$ the returned output after confirmation and fallback. For non-evolutionary baselines, the same provisional-output notation refers to their best evaluated candidate.

A.1 Benchmarks and Evaluation Partitions

Each benchmark is partitioned into SELECT, VALID, and TEST. SELECT supplies feedback for candidate generation, surrogate fitting, and parent admission. VALID is used for formal confirmation and separate post-search component diagnostics after the relevant candidates have been frozen. TEST evaluates the final returned outputs. Neither VALID nor TEST outcomes can update candidate generation, the representation, the surrogate, the parent set, or search hyperparameters.

For Terminal-Bench 2.0 (Merrill et al., 2026), we use 10 SELECT, 10 VALID, and 24 TEST tasks. The reported TEST results concern this fixed 24-task held-out subset, not the complete public benchmark. For SWE-bench Lite (Jimenez et al., 2024), the 23 development instances are partitioned by repository into 12 SELECT and 11 VALID instances. The official 300-instance test set is reserved for final evaluation. Repository membership is disjoint between SELECT and VALID; this does not imply that TEST repositories are disjoint from development repositories.

Repeated executions are first averaged within each task, and tasks are then weighted equally. Founder SELECT calibration is shared across the matched search methods. This sharing concerns the initial SELECT observations; it does not by itself specify reuse of founder VALID or TEST executions. An imported SELECT observation remains one measurement even when it initialises several search runs, and parent comparisons do not overwrite that label. Task manifests record the instance identifiers, dataset revisions, partition membership, and evaluator versions.

A.2 Models, Execution Conditions, and Information Access

For all seven search methods in Table 1, the task model is DeepSeek-V4-Flash-0731. Within each benchmark, they share the tool interfaces, execution backend, and per-task execution limits. We distinguish the task executor, which solves benchmark tasks under a harness, from a candidate pro-

Table 5: Founder populations and source repositories. Source revisions and adapted harness definitions identify the particular versions used in an experiment.

Founder	Source repository
Terminal-Bench 2.0	
II-Agent	Intelligent-Internet/ii-agent
Letta Code	letta-ai/letta
Grok CLI	superagent-ai/grok-cli
Codex CLI	openai/codex
Terminus 2	harbor-framework/harbor
OpenHands Terminal	OpenHands/OpenHands
mini-SWE-agent Terminal	SWE-agent/mini-swe-agent
Gemini CLI	google-gemini/gemini-cli
SWE-bench Lite	
Live-SWE-agent	OpenAutoCoder/live-swe-agent
Trae Agent	bytedance/trae-agent
mini-SWE-agent	SWE-agent/mini-swe-agent
Lingxi	nimasteryang/Lingxi
JoyCode Agent	jd-opensource/joycode-agent
OpenHands	OpenHands/OpenHands
SWE-agent	SWE-agent/SWE-agent
Agentless	OpenAutoCoder/Agentless

poser, which creates or edits harnesses, and the experiment orchestrator, which launches runs and collects results.

FastHarness represents states, prompts, transition conditions, skills, and tool requirements as a workflow specification. It serialises that specification into text, prepends an instruction to follow the workflow, and combines it with the benchmark task instruction for the shared agent executor. The terminal adapter delegates execution to MiniSweAgent. The agent selects and issues tool calls within the benchmark environment; there is no separate state-machine interpreter that maintains the declared workflow state or enforces each transition.

Founder systems are adapted to this prompt-conditioned interface rather than run with all of their native runtime capabilities. Accordingly, graph features describe the declared workflow, not a guaranteed execution path. Task outcomes measure behaviour under the shared executor.

Candidate proposers may inspect SELECT scores, candidate source, and the SELECT execution records permitted by their method. They cannot inspect VALID or TEST outcomes. Task-executor and proposer configurations are recorded separately. Benchmark executions initiated during proposal generation or screening are included in resource accounting rather than treated as free proposal computation.

A.3 Founder Populations and Adaptation Fidelity

Each benchmark uses eight open-source inference-time founders. Founder discovery considers public implementations and benchmark relevance, but published leaderboard scores are never used as optimisation labels. Every founder is evaluated under the fixed experimental model and benchmark-specific execution conditions.

Terminal founders are drawn from the Terminal-Bench ecosystem. SWE founders may be identified through the SWE-bench Verified ecosystem, but all scores used here are measured on the specified SWE-bench Lite partitions. Founder discovery and experimental evaluation thus use distinct sources. All reported founder scores are measured on the partitions used here.

Adaptation maps founder components to FastHarness states, transition descriptions, prompts, skills, and tool-capability requirements. The adapted versions share an execution interface; they are not evaluations of each founder's native runtime. Adaptation records identify the retained components, interface changes, omitted capabilities, and termination behaviour.

For each benchmark, the founder population is

$$\mathcal{P}_0 = \{W_1^{(0)}, \ldots, W_8^{(0)}\}. \tag{10}$$

Table 6: Search and surrogate configuration. Prior optimisers retain their own proposal mechanisms; the 32-candidate pool is not imposed on every baseline.

Setting	Value
Founders per benchmark	8
Main paired seeds	$\{0, 1, 2, 3, 4, 5\}$
Online ablation seeds	$\{0, 1, 2\}$
Parents per round	3
Candidate pool size	32
Evaluated candidates per round	1
Candidate-evaluation budget	20
Mutation probability parameter	0.5
Unlabelled projection corpus size	64
Text encoder	all-mpnet-base-v2
Raw text dimension	768
Graph propagation rounds	2
PCA output dimension	16
GP kernel	Matérn-$5/2$
GP length scale	Isotropic, adaptive
Initial observation-noise variance	10^{-3}
Acquisition	Expected improvement
EI improvement margin	0.01
UCB coefficient	1.5

All founders are admitted to the initial parent set. After the representation transform has been fitted and frozen, their SELECT evaluations initialise

$$\mathcal{D}_0 = \left\{ \left(\psi_r(W_i^{(0)}), \widetilde{F}_{\mathrm{sel}}(W_i^{(0)}) \right) \right\}_{i=1}^{8}. \tag{11}$$

Each founder contributes one GP label obtained by aggregating its initial SELECT repetitions.

A.4 Search Budget and Candidate Generation

The search budget is $G = 20$ complete candidate SELECT evaluations, excluding shared founder calibration. The proposal and admission rules below apply to GP-Harness, its representation and acquisition variants, and the two internal search controls. Each round retains a pool of $M = 32$ offspring after validity, duplicate, and novelty checks, then evaluates one candidate. Candidates rejected before benchmark execution do not consume a complete candidate evaluation, although their generation and checking costs are included in resource accounting. Parent re-evaluations and partial benchmark screening are recorded separately as harness–task executions.

Proposal generation. These methods select the three eligible parents with the highest fixed SELECT fitness, using deterministic identifier-based tie breaking. Mutation modifies the harness definition, including prompts, states, transitions, skills, validation, recovery, and termination. Recombination inherits components from multiple parents. The primary parent is the first identifier in the candidate's ordered parent list. The genetic generator places the source of the base workflow first, followed by other contributors with duplicate identifiers removed. Low-cost checks reject invalid and protocol-violating candidates before surrogate scoring. The generator additionally rejects content and role-labelled topology duplicates and applies the novelty thresholds in Section C.1. These checks operate during generation and therefore affect the candidate pool and subsequent search trajectory. A passing report is stored with each retained candidate and read again during final confirmation; it is not recomputed against the final archive. The mutation operators may propose edits that the filters reject. In particular, changing prompts while preserving inferred node roles, start and wrap-up indicators, and graph connections leaves the topology fingerprint unchanged. Such a candidate is rejected, even when its text is different. A prompt change can, however, alter an inferred role, so these checks must not be interpreted as excluding every prompt-only edit.

Parent admission. Each evaluated offspring is compared with its primary parent on all tasks in the configured SELECT phase, not a retrospectively chosen subset. The evaluator pairs parent and offspring outcomes by task and repetition index, using one repetition per SELECT task under the

stated protocol. Pairing does not imply that the two model executions share a random state. Completed records may be read from the evaluator's cache rather than executed again; cached records are not additional repetitions.

Let $r_{h,d,k}$ be the raw score and $q_{h,d,k} \in \{0, 1\}$ the protocol-failure indicator for harness h, task d, and paired repetition k. Define the protocol-adjusted score and task difference by

$$\begin{aligned} a_{h,d,k} &= (1 - q_{h,d,k}) r_{h,d,k}, \\ \Delta_d &= \frac{1}{K_d} \sum_{k=1}^{K_d} \big(a_{\text{child},d,k} - a_{\text{parent},d,k}\big). \end{aligned} \tag{12}$$

Here K_d is the number of complete paired repetitions for task d. A task is a win when $\Delta_d > 10^{-9}$, a loss when $\Delta_d < -10^{-9}$, and a tie otherwise. Thus admission uses protocol-adjusted task scores, not the token- and runtime-penalised fitness in Equation 7.

Let n be the number of distinct task records after averaging repetitions, m the number of distinct task-family labels, and n_+ and n_- the win and loss counts. The admission rule is

$$B_t(W_t) = \mathbf{1}\{n \geq 3,\ m \geq 2,\ n_+ \geq 1,\ n_+ > n_-\}. \tag{13}$$

Task families are fixed by the benchmark manifest, not inferred from outcomes. SWE-bench's manifest uses repository identifiers as families; Terminal-Bench uses the task's manifest family field. The evidence implementation groups repetitions by task identifier, family, and split. Missing family labels are mapped to `unclassified`, which the admission counter treats as one label, not as evidence of a known category. Ties count as neither wins nor losses and repetitions do not increase the number of independent tasks. Admitted offspring become eligible for future parent selection. For the GP-guided variants, every completed offspring evaluation supplies a GP label, regardless of admission. Paired parent re-evaluations neither overwrite the parent's original label nor provide additional GP labels. Parent admission is separate from final promotion.

Sequential updates and paired seeds. For the GP-guided variants, the selected candidate's first complete SELECT label is added before the next acquisition decision, and the GP is refitted to the updated dataset. After G rounds, the surrogate contains $8 + G$ labels and has scored GM candidate entries across the generated pools. These counts exclude parent re-evaluations and other task executions. Paired seeds align configured randomisation across variants, but different selections can produce different parent sets, candidate pools, and fitted posteriors.

A.5 Frozen Harness Representations

The representation transforms are fitted using a benchmark-specific unlabelled corpus

$$\mathcal{U} = \{U_1, \ldots, U_{64}\} \tag{14}$$

generated from the founder pool before labelled founder evaluation. The corpus contains valid harness definitions but no SELECT, VALID, or TEST outcomes. Feature standardisation and PCA are fitted on this corpus and then frozen throughout optimisation. The encoder, feature schema, and graph extractor are also fixed.

Text representation. The harness is serialised deterministically, including its overview, starting and terminal states, required capabilities, state identifiers and descriptions, transitions, conditions, deliverables, and skills. State and skill identifiers are ordered deterministically, and transition ordering follows the fixed serialisation procedure. The all-mpnet-base-v2 encoder produces an L2-normalised embedding:

$$e_W = \text{L2Norm}\left[\phi_{\text{text}}\left(\text{serialise}(W)\right)\right] \in \mathbb{R}^{768}. \tag{15}$$

The final text feature is

$$z_W^{\text{emb}} = P_{\text{emb}} \mathcal{N}_{\text{emb}}(e_W) \in \mathbb{R}^{16}, \tag{16}$$

where $\mathcal{N}_{\text{emb}}$ and P_{emb} are the standardisation transform and PCA projection fitted on $\mathcal{U}$. The full harness is supplied to serialisation; the content retained by the encoder also depends on its input-length and truncation configuration. The text–graph comparison therefore concerns the complete representation pipelines, not topology alone.

Graph representation. The harness is represented as an attributed directed graph $G_W = (V_W, E_W)$ of declared states and transitions. Each node has a state-type label τ_v and a state description p_v. Its initial content vector is

$$h_v^{(0)} = \mathrm{L2Norm}\left[\phi_{\mathrm{text}}(p_v)\right]. \tag{17}$$

Content propagation uses the union of incoming and outgoing neighbours:

$$\mathcal{N}(v) = \mathcal{N}_{\mathrm{in}}(v) \cup \mathcal{N}_{\mathrm{out}}(v). \tag{18}$$

Two propagation rounds compute

$$h_v^{(\ell+1)} = \frac{h_v^{(\ell)} + \sum_{u \in \mathcal{N}(v)} h_u^{(\ell)}}{1 + |\mathcal{N}(v)|}, \qquad \ell \in \{0, 1\}. \tag{19}$$

Incoming and outgoing structure remains distinguished in the directed structural features.

For each state type τ, propagated vectors are mean-pooled:

$$h_\tau^{\mathrm{state}}(W) = \begin{cases} \dfrac{1}{|V_\tau|} \displaystyle\sum_{v \in V_\tau} h_v^{(2)}, & |V_\tau| > 0, \\ \mathbf{0}, & |V_\tau| = 0, \end{cases} \tag{20}$$

where $V_\tau = \{v \in V_W : \tau_v = \tau\}$. The raw descriptor is

$$\begin{aligned} g_W = h_W^{\mathrm{WL}} \oplus \left(\bigoplus_{\tau \in \mathcal{T}} h_\tau^{\mathrm{state}}(W) \right) \\ \oplus\, h_W^{\mathrm{skill}} \oplus h_W^{\mathrm{tool}} \oplus h_W^{\mathrm{global}}. \end{aligned} \tag{21}$$

Here, h_W^{WL} contains directed Weisfeiler–Lehman features (Shervashidze et al., 2011). Skill and required-tool embeddings are mean- and max-pooled. Global features include graph size, branching, terminal-state counts, selected state-type counts, cycle presence, and start-to-wrap-up distance. The final feature is

$$z_W^{\mathrm{graph}} = P_{\mathrm{graph}} \mathcal{N}_{\mathrm{graph}}(g_W) \in \mathbb{R}^{16}. \tag{22}$$

The two routes use separate standardisation and PCA transforms. Their common output dimension controls the number of GP input features.

A.6 Gaussian-Process Surrogate and Acquisition

The surrogate models latent SELECT fitness:

$$f(z) \sim \mathcal{GP}(m, k_\theta(z, z')), \qquad y_i = f(z_i) + \epsilon_i, \qquad \epsilon_i \sim \mathcal{N}(0, \sigma_n^2). \tag{23}$$

We use a constant mean and an isotropic Matérn-$5/2$ kernel:

$$\begin{aligned} k_\theta(z, z') = \sigma_f^2 \left(1 + \sqrt{5}\rho + \frac{5}{3}\rho^2\right) \exp(-\sqrt{5}\rho), \\ \rho = \frac{\|z - z'\|_2}{\ell}. \end{aligned} \tag{24}$$

At initialisation, the length scale is the median non-zero pairwise distance between founder features. The output variance is the empirical variance of founder fitness, lower-bounded by 10^{-4}, and the observation-noise variance is 10^{-3}.

Covariance parameters are fitted by deterministic coordinate search on the exact log marginal likelihood, with six optimisation iterations and warm starts from the preceding fit. Positive parameters are represented in log space and bounded to $[-16, 16]$. For each covariance setting, the constant mean is estimated by generalised least squares. Cholesky factorisation starts with jitter 10^{-9} and increases it geometrically when necessary, for at most eight attempts.

Before selecting candidate W_t, the posterior conditioned on $\mathcal{D}_{t-1}$ provides latent mean $\mu_{t-1}(z)$ and standard deviation $\sigma_{t-1}(z)$. Acquisition uses latent uncertainty without adding observation noise. Founder labels average three executions per task, whereas offspring labels use one. The GP nevertheless uses a single homoscedastic noise parameter; it does not explicitly model the different replication counts.

Expected improvement. Let $y_{t-1}^{\max} = \max_{(z_i,y_i)\in\mathcal{D}_{t-1}} y_i$ be the best observed SELECT fitness before round t and define

$$\Delta_{t-1}(W) = \mu_{t-1}(z_W) - y_{t-1}^{\max} - \xi, \qquad \xi = 0.01. \tag{25}$$

For $\sigma_{t-1}(z_W) > 0$,

$$\begin{aligned} u_{t-1}(W) &= \frac{\Delta_{t-1}(W)}{\sigma_{t-1}(z_W)}, \\ \alpha_{t-1}^{\mathrm{EI}}(W) &= \Delta_{t-1}(W)\Phi(u_{t-1}(W)) + \sigma_{t-1}(z_W)\phi(u_{t-1}(W)). \end{aligned} \tag{26}$$

For zero posterior variance, the limiting value is

$$\alpha_{t-1}^{\mathrm{EI}}(W) = \max\{\Delta_{t-1}(W), 0\}. \tag{27}$$

Here ϕ and Φ are the standard normal density and cumulative distribution functions, respectively. This limit is also used for numerically negligible standard deviations. Candidate ties are resolved by the fixed harness identifier.

Acquisition variants. The ablations use

$$\alpha_{t-1}^{\mathrm{UCB}}(W) = \mu_{t-1}(z_W) + 1.5\,\sigma_{t-1}(z_W) \tag{28}$$

and

$$\alpha_{t-1}^{\mathrm{Mean}}(W) = \mu_{t-1}(z_W). \tag{29}$$

The variants share the GP specification and refitting procedure, not the same realised posterior: each variant fits its own observations along its online search trajectory. The posterior-mean variant is the case $s = \mu_{t-1}$ of Proposition 1, and therefore isolates the exploitation component of surrogate-guided selection.

A.7 Baselines and Method-Specific Adaptations

Internal controls. The founder baseline returns the founder with the highest $\widetilde{F}_{\mathrm{sel}}$ without candidate search. Random offspring selection retains fitness-ranked parents, the evolutionary generator, candidate-pool size, and validity and novelty filters, but selects uniformly from the retained pool. Surrogate-free evolution instead ranks candidates by their primary parent's fixed SELECT fitness, with seeded random tie breaking. Neither control uses candidate features or a candidate-level performance surrogate; both satisfy the independence condition of Corollary 1.

Prior methods. Meta-Harness (Lee et al., 2026) uses a history-conditioned agentic proposer with access to previous harness code, scores, and execution traces. HarnessBank (Luo et al., 2026) uses a semantic gene bank, reinvention, recombination, and gated screening. The FastHarness adaptations retain their method-specific proposal and selection mechanisms.

AHE (Lin et al., 2026a) associates edits with observable components, experience, and prediction–outcome records. HarnessCompass (Zhang et al., 2026b) combines task-agnostic edit constraints, proactive executor feedback, component-wise optimisation, and consolidation.

Shared comparison conditions. The comparison includes GP-Harness, random offspring selection, surrogate-free evolution, Meta-Harness, HarnessBank, AHE, and HarnessCompass. Within each benchmark, all seven methods share the task model, tool interfaces, execution backend, per-task limits, task partitions, and initial eight founders. Each uses the same six paired search seeds, $\{0, 1, 2, 3, 4, 5\}$, and receives 20 complete candidate SELECT evaluations.

Every search method uses the common SELECT selection, formal VALID confirmation, saved novelty-report requirement, and SELECT-selected founder fallback in Section A.8. The TEST mean includes all six returned outputs, including fallbacks. The strongest-founder baseline performs no search and returns the SELECT-selected founder directly.

The proposal-stage filtering described in Section A.4 applies to GP-Harness and the two internal search controls. A common final requirement for a saved novelty report does not, by itself, imply

Table 7: Proposal and selection mechanisms distinguishing the prior-method baselines. Candidate-evaluation budgets are separate from proposal computation and screening costs.

Method	Mechanism
Meta-Harness	History-conditioned proposals using prior code, scores, and execution traces.
HarnessBank	Semantic gene-bank organisation, reinvention, recombination, and gated screening.
AHE	Observable components and experience, with edit predictions and outcome records.
HarnessCompass	Task-agnostic editing, proactive feedback, component-wise optimisation, and consolidation.

identical proposal-stage filtering or archive contents for the four prior-method adaptations. These are separate properties from their shared output rule.

The common evaluation conditions do not require identical candidate proposers, feedback use, internal screening, or proposal-model calls. Methods retain those mechanisms. Additional screening executions and model calls are included in resource accounting outside the cap on complete candidate evaluations. The main table reports these evaluated adaptations rather than scores copied from their original publications.

Candidate proposers receive SELECT information only. Their model calls and any benchmark-based screening are accounted for separately from the complete candidate-evaluation budget. The adaptations retain method-specific proposal and selection mechanisms within the common harness interface.

A.8 Fitness, Final Selection, and TEST Aggregation

Task-level aggregation. For split s, task i, and repetition k, let $r_{i,k}^{(s)}(W)$ be the benchmark score. For $K_i^{(s)}$ repetitions, define

$$\bar{r}_i^{(s)}(W) = \frac{1}{K_i^{(s)}} \sum_{k=1}^{K_i^{(s)}} r_{i,k}^{(s)}(W), \qquad R_s(W) = \frac{1}{n_s} \sum_{i=1}^{n_s} \bar{r}_i^{(s)}(W). \tag{30}$$

For binary outcomes, each $r_{i,k}^{(s)}(W)$ is zero or one. This aggregation gives equal weight to each task rather than to each repeated execution.

Penalised fitness. For every execution, define

$$\begin{aligned} c_{i,k}^{(s)}(W) &= \min\left\{1, \frac{\mathrm{tokens}_{i,k}^{(s)}(W)}{10^6}\right\}, \\ t_{i,k}^{(s)}(W) &= \min\left\{1, \frac{\mathrm{time}_{i,k}^{(s)}(W)}{900\,\mathrm{s}}\right\}, \\ q_{i,k}^{(s)}(W) &\in \{0,1\}, \end{aligned} \tag{31}$$

where q marks a harness-attributable protocol failure. Applying the same within-task and across-task averaging gives $C_s(W)$, $T_s(W)$, and $Q_s(W)$. The fitness is

$$\widetilde{F}_s(W) = R_s(W) - 0.05C_s(W) - 0.05T_s(W) - 0.10Q_s(W). \tag{32}$$

SELECT search uses $\widetilde{F}_{\mathrm{sel}}$, and formal confirmation uses $\widetilde{F}_{\mathrm{val}}$. TEST performance and diagnostic VALID performance are reported using the unpenalised metrics R_{test} and R_{val}.

Provisional selection. Define the SELECT-selected founder and best evaluated candidate by

$$\begin{aligned} W^{\star}_{\mathrm{founder,sel}} &\in \arg\max_{W \in \mathcal{P}_0} \widetilde{F}_{\mathrm{sel}}(W), \\ W^{\star}_{\mathrm{off}} &\in \arg\max_{W \in \mathcal{C}_{\mathrm{eval}}} \widetilde{F}_{\mathrm{sel}}(W), \end{aligned} \tag{33}$$

where $\mathcal{C}_{\text{eval}}$ contains all candidates evaluated during search, excluding founders. For evolutionary methods, this set includes offspring that were not admitted to the parent set. Identifier-based tie breaking is fixed before selection. The candidate enters formal confirmation only if

$$\widetilde{F}_{\text{sel}}(W^{\star}_{\text{off}}) > \widetilde{F}_{\text{sel}}(W^{\star}_{\text{founder,sel}}). \tag{34}$$

Otherwise, the run returns $W^{\star}_{\text{founder,sel}}$ without formal VALID confirmation.

VALID confirmation and returned output. The eligible candidate is frozen before VALID evaluation. Define

$$W^{\star}_{\text{founder,val}} \in \arg\max_{W \in \mathcal{P}_0} \widetilde{F}_{\text{val}}(W). \tag{35}$$

The candidate passes the performance criterion when

$$\widetilde{F}_{\text{val}}(W^{\star}_{\text{off}}) > \widetilde{F}_{\text{val}}(W^{\star}_{\text{founder,val}}). \tag{36}$$

Because offspring labels use one execution per task and founder labels use three, Eq. equation 34 favours offspring; the VALID comparison in Eq. equation 36 removes this advantage and is conservative with respect to promotion (Appendix B). Let $\nu_{\text{gen}}(W) \in \{0, 1\}$ be the saved proposal-stage novelty decision for W, evaluated against $\mathcal{R}_{\text{gen}}(W)$. Final confirmation requires an available passing report and does not recompute it against the completed archive (Section C.1). The returned output is

$$W^{\star} = \begin{cases} W^{\star}_{\text{off}}, & \widetilde{F}_{\text{sel}}(W^{\star}_{\text{off}}) > \widetilde{F}_{\text{sel}}(W^{\star}_{\text{founder,sel}}), \\ & \widetilde{F}_{\text{val}}(W^{\star}_{\text{off}}) > \widetilde{F}_{\text{val}}(W^{\star}_{\text{founder,val}}), \\ & \nu_{\text{gen}}(W^{\star}_{\text{off}}) = 1, \\ W^{\star}_{\text{founder,sel}}, & \text{otherwise.} \end{cases} \tag{37}$$

VALID cannot select a replacement candidate or a different founder, and cannot alter any search state. Passing the penalised VALID comparison does not necessarily imply a strict increase in the unpenalised VALID task metric.

TEST aggregation. For every search method in Table 1, all seed-level outputs are fixed before TEST is opened. Each output is evaluated without reselection or adaptation. For $S = 6$ seeds, the reported mean is

$$\widehat{R}_{\text{test}} = \frac{1}{S} \sum_{j=1}^{S} R_{\text{test}}(W^{\star}_{j}). \tag{38}$$

All six seeds are included, including runs that return a founder. The statistic therefore measures the complete procedure, not performance conditional on successful promotion.

Diagnostic ablations. For every variant and internal control reported in Table 2, the offspring with the highest SELECT fitness from each of the three runs is frozen and evaluated on VALID, regardless of SELECT eligibility or formal promotion. This protocol also applies to random offspring selection and surrogate-free evolution. All variants and controls use the same VALID tasks and repetition counts. The diagnostic mean is

$$\widehat{R}^{\text{diag}}_{\text{val}} = \frac{1}{3} \sum_{j=1}^{3} R_{\text{val}}(W^{\star}_{\text{off},j}). \tag{39}$$

No run is excluded for failing SELECT eligibility or promotion, and no selected offspring is replaced by a founder in this mean. The diagnostic statistic has a different purpose from formal confirmation and cannot trigger further search or alter selection, promotion, fallback, or TEST outputs. Different statistical uses do not necessarily correspond to different executions. If the same VALID records supply both statistics, the raw score and penalised fitness are computed from those records and their execution cost is counted once. If distinct records are used, only the formal confirmation records determine promotion; diagnostic records cannot replace them. Any additional diagnostic execution is outside the complete candidate-evaluation budget.

A.9 Failure Handling and Resource Accounting

Malformed actions, invalid tool calls, exhausted execution budgets, harness-attributable timeouts, and unsuccessful termination remain method outcomes. Provider outages and corrupted environments are distinguished from task failures. Infrastructure retries are recorded at the operation level: dependency installation, provider requests, and complete benchmark executions are separate operations. An unresolved infrastructure failure is recorded as missing rather than silently converted to a failed task.

Missing task outcomes and founder fallbacks are different cases. A fallback is a valid seed-level output and remains in the main aggregate. Any missing or excluded task outcome requires its cause and effective denominator to be reported. Paired analyses retain method pairing and identify the task outcomes included in each comparison.

Resource accounting separates:

- shared founder calibration;
- complete candidate SELECT evaluations;
- partial candidate screening;
- parent re-evaluations for admission;
- formal VALID confirmation;
- post-search diagnostic evaluations;
- final TEST evaluations;
- proposal computation and infrastructure retries.

The records distinguish harness–task executions, proposal and task-model tokens, and wall-clock time. Execution reuse is distinct from inclusion in an aggregate. Reusing a record does not create a new model execution, but every returned seed-level output still contributes to the TEST mean. Shared calibration and cached evaluation records must be counted once in physical execution costs, rather than once for every downstream use. Independently executed repetitions are counted separately, even when the harness is unchanged. The candidate budget measures benchmark sample efficiency under the stated accounting convention, not equal total compute or time. Displayed task metrics are converted to percentages where indicated; the SELECT fitness columns retain their unscaled values.

A.10 Statistical Summaries

The main comparison uses six search seeds as repeated executions of the optimisation procedure. Task executions within a seed are not independent repetitions of the entire search. For each method, let $x_j = R_{\text{test}}(W_j^\star)$. The reported mean is

$$\bar{x} = \frac{1}{S}\sum_{j=1}^{S} x_j. \tag{40}$$

For seed-level quantities, the between-seed standard error is

$$\mathrm{SE}(\bar{x}) = \sqrt{\frac{1}{S(S-1)}\sum_{j=1}^{S}(x_j - \bar{x})^2}. \tag{41}$$

For the search curves in Figure 2, x_j is replaced by $C_j(t)$, the best offspring SELECT fitness observed up to candidate-evaluation index t as defined below, and $S = 3$. This standard error describes variation over search seeds conditional on the fixed evaluation tasks.

For paired methods A and B, the seed-level difference is

$$d_j = R_{\text{test}}(W_j^{\star,A}) - R_{\text{test}}(W_j^{\star,B}). \tag{42}$$

For baselines that do not depend on the search seed, $W_j^{\star,B}$ is the same harness for every j. Mean differences concern the complete returned-output procedures, including founder fallbacks. Table 8

reports $\bar{d}$ with its between-seed standard error for the three comparisons discussed in Section 4. All six differences favour GP-HARNESS; only the SWE-bench Lite gain over the SELECT-selected founder exceeds two standard errors, and the remaining five lie within 1.5 standard errors of zero, so the ordering in Table 1 is consistent across seeds but the margins over surrogate-free evolution and HarnessBank are not resolved at this number of seeds.

Table 8: Paired seed-level TEST differences, GP-HARNESS minus baseline, in percentage points of R_{test}. Means with between-seed standard errors, conditional on the fixed TEST tasks; positive values favour GP-HARNESS.

Benchmark	Baseline	$\bar{d}$ (pp)
Terminal-Bench 2.0	SELECT-selected founder	$+10.4 \pm 7.4$
	Surrogate-free evolution	$+4.9 \pm 8.1$
	HarnessBank	$+2.8 \pm 6.6$
SWE-bench Lite	SELECT-selected founder	$+8.0 \pm 3.7$
	Surrogate-free evolution	$+4.6 \pm 4.3$
	HarnessBank	$+2.1 \pm 4.5$

Discovery is a strict SELECT-fitness improvement over the strongest SELECT founder. Promotion additionally requires the penalised VALID comparison and an available passing proposal-stage novelty report. Both are reported as counts out of all six search seeds. The first-gain statistic is the median evaluation index among seeds with a discovery and is interpreted together with that discovery count. The online ablation tables average all three seeds under the diagnostic protocol in Section A.8.

Search-curve and ablation quantities. For seed j, define the founder-inclusive running best by

$$B_j(t) = \max\left\{ \max_{W \in \mathcal{P}_{0,j}} \widetilde{F}_{\text{sel}}(W), \max_{1 \le i \le t} \widetilde{F}_{\text{sel}}(W_{j,i}) \right\}, \tag{43}$$

where at $t = 0$ only the founder term is used, and the offspring-only running best by

$$C_j(t) = \max_{1 \le i \le t} \widetilde{F}_{\text{sel}}(W_{j,i}). \tag{44}$$

A discovery in seed j corresponds to $C_j(G)$ exceeding the founder term of $B_j(G)$. The search curves in Figure 2 average $C_j(t)$ over the three online-ablation seeds, with shaded bands giving the between-seed standard error above computed over those seeds; their endpoints $C_j(G)$ are the SELECT column of Table 2, which averages the same quantity over the same runs. Thus, $B_j(G) \ge C_j(G)$ whenever they refer to the same run. Each run is identified by its benchmark manifest, method configuration, seed, founder labels, representation transform, and execution records.

B PROOFS FOR THE SELECTION ANALYSIS

Throughout, $e_M = \mathbb{E}[\max_{i \le M} Z_i]$ for independent $Z_i \sim \mathcal{N}(0,1)$. Numerical integration gives $e_3 = 3/(2\sqrt{\pi}) \approx 0.846$, $e_8 \approx 1.424$, $e_{16} \approx 1.766$, $e_{20} \approx 1.868$, $e_{32} \approx 2.070$, and $e_{64} \approx 2.344$. The sequence (e_M) is increasing and satisfies $e_M \le \sqrt{2 \log M}$ (David & Nagaraja, 2003).

Evaluation accounting. Over G rounds with M retained candidates per round, a generational algorithm that evaluates every candidate before selection uses MG complete candidate evaluations, whereas GP-HARNESS uses G. Founder evaluations are common to both and omitted. A steady-state or subsampled evolutionary method can also evaluate one candidate per round, but must then choose it without candidate-level predictions; Proposition 1 quantifies the value of making that choice with a surrogate.

B.1 VALUE OF SURROGATE-GUIDED SELECTION

Let (f_j, s_j), $j = 1, \dots, M$, be independent bivariate Gaussian pairs with means (m_f, m_s), standard deviations (σ_t, σ_s), and correlation r_t. Gaussian regression gives the decomposition

$$f_j = m_f + r_t \sigma_t \frac{s_j - m_s}{\sigma_s} + \eta_j, \qquad \eta_j \sim \mathcal{N}\big(0, \sigma_t^2 (1 - r_t^2)\big), \tag{45}$$

where each η_j is independent of s_j. Because the pairs are independent, the vector $(\eta_1, \ldots, \eta_M)$ is independent of $(s_1, \ldots, s_M)$.

Proof of Proposition 1. The selected index $J = \arg\max_j s_j$ is a function of $(s_1, \ldots, s_M)$ alone; ties occur with probability zero. Hence $\mathbb{E}[\eta_J] = \mathbb{E}\big[\mathbb{E}[\eta_J \mid s_1, \ldots, s_M]\big] = 0$, and Eq. equation 45 gives

$$\mathbb{E}[f_J] = m_f + r_t \sigma_t \, \mathbb{E}\left[\max_{j \le M} \frac{s_j - m_s}{\sigma_s}\right] = m_f + r_t \sigma_t e_M,$$

since the standardised scores are independent standard normals. A uniformly drawn index U is independent of all pairs, so $\mathbb{E}[f_U] = m_f$. Subtracting the two expectations proves Eq. equation 8. □

Relating r_t to the diagnostics. Table 3 reports Spearman correlations ρ_{rank}. For a bivariate Gaussian pair, the Pearson and Spearman correlations satisfy $r = 2\sin(\pi\rho_{\mathrm{rank}}/6)$, so the graph-GP values 0.48 and 0.44 correspond to $r \approx 0.50$ and $r \approx 0.46$. Two effects make these values conservative estimates of the r_t in Proposition 1. First, they correlate predictions with noisy labels $y = f + \epsilon$ rather than with latent fitness; if ϵ is independent of the score, $\mathrm{corr}(s, y) = \mathrm{corr}(s, f)\, \sigma_f / \sqrt{\sigma_f^2 + \sigma_\epsilon^2}$. Second, they are computed on acquisition-selected candidates, whose range of predicted values is narrower than that of the full candidate set.

B.2 Parent-Level Selection Rules

Proof of Corollary 1. Let W^R be chosen by a rule, possibly randomised, that is independent of $\{w_W\}$. Then $\mathbb{E}[f(z_{W^R})] = \mathbb{E}[b_{p(W^R)}] + \mathbb{E}[w_{W^R}]$. Conditional on the parent effects and the rule's internal randomness, W^R is fixed and w_{W^R} has mean zero, so the second term vanishes. The first term is at most $\mathbb{E}[\max_{p \le P} b_p] = m + \sigma_B e_P$; if fewer than P parents are represented, the bound still holds because (e_k) is increasing. A uniform choice is independent of all effects, so $\mathbb{E}[f(z_{W^u})] = m$, and the improvement is at most $\sigma_B e_P$.

For the comparison, a candidate-level score whose gain is $r_t \sigma_t e_M$ exceeds this bound whenever $r_t > (\sigma_B/\sigma_t)(e_P/e_M)$. Since $\sigma_t^2 = \sigma_B^2 + \sigma_W^2 \ge \sigma_B^2$, the condition $r_t > e_P/e_M$ is sufficient. □

Random offspring selection and surrogate-free evolution, which ranks candidates by the SELECT fitness of their primary parent and breaks ties at random, both satisfy the independence condition. The comparison imports the candidate-level gain from Proposition 1, which treats candidates as independent draws, whereas candidates sharing a parent are correlated under the parent model. The threshold is therefore a stylised guide rather than an exact criterion. With $P = 3$ and $M = 32$, $e_3/e_{32} \approx 0.409$.

B.3 Optimism of the Observed Incumbent

Proof of Proposition 2. Write $\tau^2 = \sigma_f^2 + \sigma_\epsilon^2$. The pairs (f_i, y_i) are independent bivariate Gaussians with $\mathrm{Var}(y_i) = \tau^2$ and $\mathrm{Cov}(f_i, y_i) = \sigma_f^2$, so

$$\mathbb{E}[f_i \mid y_1, \ldots, y_G] = m + \frac{\sigma_f^2}{\tau^2}(y_i - m).$$

The index $i^\star$ is a function of $(y_1, \ldots, y_G)$, hence $\mathbb{E}[f_{i^\star}] = m + (\sigma_f^2/\tau^2)(\mathbb{E}[y_{i^\star}] - m)$. Moreover $\mathbb{E}[y_{i^\star}] = m + \tau e_G$. Therefore

$$\mathbb{E}[y_{i^\star}] - \mathbb{E}[f_{i^\star}] = \left(1 - \frac{\sigma_f^2}{\tau^2}\right) \tau e_G = \frac{\sigma_\epsilon^2}{\tau}\, e_G.$$

□

Replicated labels and confirmation. For a fixed task set, averaging K independent executions per task divides the execution-noise variance by K. Founder SELECT labels use $K = 3$ and offspring labels use $K = 1$. As an illustration, if $\sigma_f = \sigma_\epsilon$ for single executions, Eq. equation 9 gives an optimism of about $1.32\,\sigma_\epsilon$ for the best of $G = 20$ offspring, compared with about $0.41\,\sigma_\epsilon$ for the best of eight founders with $K = 3$. The eligibility test in Eq. equation 34 is therefore biased towards offspring. VALID confirmation uses tasks and executions independent of those that determined the frozen offspring, so its VALID estimate is unbiased for that offspring, while the founder side of Eq. equation 36 is a maximum over eight founders and retains a positive optimism. The confirmation comparison is therefore conservative with respect to promotion.

B.4 Frontier Regret for the UCB Variant

The following result specialises the GP-UCB analysis of Srinivas et al. (2010) to the finite candidate sets generated along the search trajectory. It applies to the UCB acquisition in Eq. equation 28, not to the default EI acquisition. Let $W_t^\star \in \arg\max_{W \in \mathcal{C}_t} f(z_W)$ and $r_t^{\mathrm{fr}} = f(z_{W_t^\star}) - f(z_{W_t})$.

Proposition 3. *Consider the UCB rule $W_t \in \arg\max_{W \in \mathcal{C}_t} \mu_{t-1}(z_W) + \sqrt{\beta_t}\,\sigma_{t-1}(z_W)$ and suppose that, with probability at least $1-\delta$, $|f(z_W) - \mu_{t-1}(z_W)| \le \sqrt{\beta_t}\,\sigma_{t-1}(z_W)$ holds simultaneously for all $t \le G$ and all $W \in \mathcal{C}_t$. Then, on this event, $r_t^{\mathrm{fr}} \le 2\sqrt{\beta_t}\,\sigma_{t-1}(z_{W_t})$ for every t. If in addition $k(z,z) \le \kappa^2$ and β_t is non-decreasing, then*

$$R_G^{\mathrm{fr}} := \sum_{t=1}^{G} r_t^{\mathrm{fr}} \le 2\sqrt{\frac{2\kappa^2 G \beta_G \gamma_G}{\log(1+\kappa^2/\sigma_n^2)}} = O\Big(\sqrt{G\beta_G\gamma_G}\Big),$$

where γ_G is the maximum information gain from G observations.

Proof. On the confidence event, $f(z_{W_t^\star}) \le \mu_{t-1}(z_{W_t^\star}) + \sqrt{\beta_t}\sigma_{t-1}(z_{W_t^\star})$. Since W_t maximises the UCB score on $\mathcal{C}_t$, the right-hand side is at most $\mu_{t-1}(z_{W_t}) + \sqrt{\beta_t}\sigma_{t-1}(z_{W_t})$. The lower confidence bound gives $f(z_{W_t}) \ge \mu_{t-1}(z_{W_t}) - \sqrt{\beta_t}\sigma_{t-1}(z_{W_t})$, and subtracting yields the per-round bound.

For the cumulative bound, $\beta_t \le \beta_G$ and Cauchy–Schwarz give

$$R_G^{\mathrm{fr}} \le 2\sqrt{\beta_G}\sum_{t=1}^{G}\sigma_{t-1}(z_{W_t}) \le 2\sqrt{G\beta_G\sum_{t=1}^{G}\sigma_{t-1}^2(z_{W_t})}.$$

For Gaussian observation noise, $\frac{1}{2}\sum_{t=1}^{G}\log\big(1+\sigma_n^{-2}\sigma_{t-1}^2(z_{W_t})\big)$ equals the information gain of the selected observations conditional on the founder observations. By submodularity of the information gain, this is at most the unconditional gain and hence at most γ_G. Because $0 \le \sigma_{t-1}^2(z) \le \kappa^2$ and $x \mapsto \log(1+\sigma_n^{-2}x)$ is concave, $\sigma_{t-1}^2 \le \kappa^2\log(1+\sigma_n^{-2}\sigma_{t-1}^2)/\log(1+\kappa^2/\sigma_n^2)$, so $\sum_t \sigma_{t-1}^2(z_{W_t}) \le 2\kappa^2\gamma_G/\log(1+\kappa^2/\sigma_n^2)$. Substitution completes the proof. □

A uniform-selection contrast. Consider a candidate set containing one candidate of fitness $a+\Delta$ and $M-1$ candidates of fitness a. Uniform selection gives $\mathbb{E}[r_t^{\mathrm{fr}}] = (1-1/M)\Delta$, so $\mathbb{E}[R_G^{\mathrm{fr}}] = G(1-1/M)\Delta = \Theta(G)$. This is a stylised contrast rather than a general lower bound for evolutionary algorithms.

Scope of the analysis. Propositions 1 and 2 use stylised Gaussian models: candidates that share parents are correlated, and penalised fitness is bounded rather than Gaussian. Their role is to identify which measurable quantities govern the value of surrogate-guided selection and the optimism of the final recommendation, not to predict exact gains. Proposition 3 is conditional on a calibration event for fixed kernel hyperparameters, whereas our hyperparameters are refitted after every evaluation and the diagnostics in Table 3 indicate weak calibration of individual predictions. None of the results guarantees that the generated candidate sets contain a globally optimal harness.

Table 9: Ordered substring rules for state-type assignment. Literal strings retain the spelling used by the implementation.

Type	Matching substrings, checked in the listed row order
wrapup	wrapup, final_select, finalise, finalize
handoff	handoff, submit, submission
setup	setup, start, initialise, initialize, intake
planning	plan, scope, intent, decompose
search	search, scan, retrieve, fetch, browse, source
localisation	localise, localize, locate, inspect
execution	run, execute, eval, evaluate, mutate, apply
validation	valid, score, test, check, critique, review
repair	repair, retry, fix, recover
aggregation	aggregate, merge, pool, pareto, summar
writing	write, report, draft, package

C Structural and Reproducibility Evidence

C.1 Structural Novelty and Lineage

Timing and reference history. Novelty assessment is part of candidate generation, before acquisition and benchmark evaluation. At the start of a generation round, the generator takes a snapshot of the available archive and adds the selected parents. The archive includes founders and previously retained proposals, including those not selected for evaluation; it is not restricted to completed SELECT evaluations. Only accessible harness definitions with a `task-overview.md` file enter the comparison. For candidate W, write this reference set as $\mathcal{R}_{\mathrm{gen}}(W)$.

A report stores the decision, reference count, closest-reference identifiers, overlap and distance statistics, content and topology fingerprints, and threshold configuration. Final confirmation reads the stored decision $\nu_{\mathrm{gen}}(W)$; a missing or non-passing report does not satisfy this requirement. It does not recompute distances against candidates generated later. Thus, the novelty claim concerns the recorded generation-time comparison, not a fresh comparison against the completed search archive.

Content and topology fingerprints. The content fingerprint hashes file paths and raw contents in sorted path order using SHA-256. Files named `reflection.md` and files inside `runs` or `__pycache__` directories are excluded. This check preserves filenames and byte-level differences; it is not semantic text normalisation.

The topology fingerprint is computed from the parsed directed graph. Each node starts with a role signature containing its inferred state type and indicators for the starting and wrap-up states. Directed Weisfeiler–Lehman refinement uses the current node label and separately sorted incoming and outgoing neighbour-label multisets. After $\max\{3, |V_W|\}$ rounds, the sorted node-label and directed edge-label multisets, together with the node count, are hashed. Matching fingerprints are rejected. This is a fixed duplicate-detection rule, not a complete graph-isomorphism test.

Content and topology checks also exclude duplicates among candidates already retained in the same pool. The threshold comparisons below use the generation's archive-and-parent reference snapshot rather than all pairwise distances within the new pool.

State types and node correspondence. State types are assigned by an ordered substring rule. The lower-cased state name is checked first. If no rule matches, the same rules are applied to the first 300 characters of the lower-cased state description. The first matching row in Table 9 determines the type; otherwise it is `other`. The role signature additionally records whether the node is the starting or wrap-up state.

There is no optimised node-to-node alignment. The state-file comparison below matches exact filenames, whereas structural comparison aggregates role counts. A rename leaves the structural descriptor unchanged only if the inferred roles and start/wrap-up markers remain unchanged. Description edits can also change a role when the name does not match any type rule.

State-file change. Let $S(W)$ be the filenames of the Markdown files directly inside `states`, and let $H_W(n)$ be the SHA-256 hash of file n's raw bytes. The unchanged-file overlap with reference U is

$$o(W,U) = \frac{|\{n \in S(W) \cap S(U) : H_W(n) = H_U(n)\}|}{\max\{|S(W)|, |S(U)|, 1\}}. \tag{46}$$

The change ratio is

$$\rho(W) = 1 - \max_{U \in \mathcal{R}_{\text{gen}}(W)} o(W,U), \qquad \rho(W) \geq 0.15. \tag{47}$$

The denominator accounts for unequal state counts. Filename changes and byte-level edits, including whitespace edits, can reduce overlap, so this quantity measures file change rather than semantic change.

State-description similarity. Non-empty state descriptions are lower-cased, whitespace-normalised, and split on whitespace. Let $T(p)$ be the set of consecutive three-word sequences in description p. For a non-empty description shorter than three words, its complete word sequence forms one set element. Similarity is Jaccard overlap,

$$J(p,q) = \frac{|T(p) \cap T(q)|}{|T(p) \cup T(q)|}, \tag{48}$$

with similarity one if both sets are empty. Let $\mathcal{D}_{\mathcal{R}}$ contain all non-empty state descriptions from all references. For each candidate description p, compute

$$s(p) = \max_{q \in \mathcal{D}_{\mathcal{R}}} J(p,q). \tag{49}$$

The maximum is defined as zero when the reference-description collection is empty. At least one candidate description must have $s(p) \leq 0.85$. Descriptions are compared against all reference descriptions, without requiring matched state names or a separate file-change classification.

Role-count structural distance. For role signature r, let $a_W(r)$ count nodes with that role. For an ordered role pair (r,s), let $b_W(r,s)$ count declared directed transitions from role r to role s whose targets resolve to states inside the harness. Parallel declared transitions contribute separately. The distance is

$$\begin{aligned} d_{\text{role}}(W,U) = &\sum_r |a_W(r) - a_U(r)| \\ &+ \sum_{r,s} |b_W(r,s) - b_U(r,s)|. \end{aligned} \tag{50}$$

The novelty threshold requires

$$\min_{U \in \mathcal{R}_{\text{gen}}(W)} d_{\text{role}}(W,U) \geq 3. \tag{51}$$

This is an L1 distance between count descriptors, not an exact graph-edit distance. A one-count increase or decrease contributes one; moving a count between two role bins contributes two if other counts are unchanged. Changing a node role can also alter incident edge-role counts. The calculation uses net count differences, not a sum over an edit script, and distinct graphs can have zero distance.

Decision and interpretation. A candidate passes only if both fingerprint checks and all three thresholds pass. These tests use no task outcomes but influence which candidates reach the surrogate and evaluator. In terms of Proposition 1, they act on the candidate-set spread σ_t: a near-duplicate of a reference harness is likely to have similar latent fitness and therefore adds little spread from which the acquisition can select. All GP-evaluated candidates have already passed proposal-stage screening; five of six runs on each benchmark subsequently meet final confirmation. The remaining run returns its SELECT-selected founder and remains in TEST aggregation.

The checks establish file, description, and role-based differences relative to the generation-time references. Because state types can depend on names and descriptions, passing the gate does not by itself prove that a transition was added or removed. Nor does it establish global novelty, enforced runtime transitions, or a causal effect of an individual edit. Lineage records identify source parents, inherited components, and state and transition edits separately from the numerical novelty scores.

C.2 Reproducibility Records and Baseline Fidelity

The result records connect each reported aggregate to its benchmark configuration, harness definitions, and task outcomes. The relevant records comprise:

- task manifests, dataset revisions, and evaluator versions;
- founder source revisions and adapted harness definitions;
- task-executor and candidate-proposer configurations;
- search seeds and completed candidate-evaluation counts;
- frozen representation corpora and fitted transformations;
- candidate definitions, parentage, and acquisition decisions;
- task-level outcomes, failures, and execution costs;
- VALID fitness, novelty decisions, and returned outputs;
- TEST outcomes, including founder fallbacks;
- aggregation procedures and figure-generation settings.

Harness identifiers and configuration records associate an evaluation with the particular program, model, and task set that produced it. The diagnostic ablation records distinguish selected offspring from the fallback-inclusive outputs used in the main comparison.

For Meta-Harness, HarnessBank, AHE, and HarnessCompass, adaptation records distinguish the proposer, available feedback, persistent search state, editable components, screening policy, and changes from the original system. The baseline comparison concerns these adapted implementations under the evaluation conditions reported in this paper.