GENERATE AND SELECT: A FRAMEWORK FOR BAYESIAN OPTIMIZATION WITH LLM PROPOSALS

Anonymous authors
Paper under double-blind review

ABSTRACT

Optimizing structured objects, such as molecules and symbolic expressions, is difficult because the candidate domain is vast, often implicit, and contains many invalid or low-quality objects. Large language models (LLMs) can construct promising candidates, but generation alone does not allocate scarce black-box evaluations. We introduce generate-and-select (GAS), a modular framework that separates candidate generation from evaluation selection within an adaptive feedback loop. At each iteration, a black-box LLM generates r candidates from previous observations, and Bayesian optimization (BO) with a Gaussian-process (GP) surrogate selects q candidates from a cumulative pool. Assuming generation supplies improvement opportunities, we bound Bayesian simple regret relative to the best generator-reachable value in terms of generator improvement and cumulative missed improvements in selection. Expected regret vanishes exactly when average expected missed improvement vanishes, and a GP Thompson sampling (GP–TS) specialization depends on information gain over the generator-reachable domain. Across four structured and synthetic tasks, GAS matches or outperforms direct LLM optimizers and BO–LLM methods, with statistically significant gains over the strongest baseline on three. It achieves comparable or better performance than the closest BO–LLM baseline while using over two orders of magnitude fewer LLM tokens. Together, these results establish GAS as a strong, modular, and efficient framework for BO with LLM-generated candidates.

1 INTRODUCTION

Many scientific and design problems require optimizing structured objects, such as molecules, programs, or symbolic expressions, under a limited evaluation budget. Bayesian optimization (BO) is well suited to expensive black-box objectives (Frazier, 2018; Garnett, 2023), but requires a candidate set or parameterized region over which to optimize an acquisition function. In structured domains, syntactic, semantic, or feasibility constraints may make this search space impossible to enumerate or optimize over directly. Candidate construction then becomes part of the optimization problem.

Large language models (LLMs) provide one interface for constructing structured candidates. They can use task instructions, semantic knowledge, and previous observations to propose objects in varied representations. Generation alone, however, does not determine which proposals merit a costly evaluation. Direct LLM optimizers typically evaluate generated proposals without separately ranking them using a probabilistic surrogate (Yang et al., 2024; Brahmachary et al., 2025; Lange et al., 2024). BO–LLM hybrids introduce uncertainty-aware selection, but may also use the LLM for prediction or use acquisition optimization to guide generation (Liu et al., 2024; Agarwal et al., 2025). Motivated by this, we ask: *How can we combine the generative flexibility of LLMs with the uncertainty-aware, sample-efficient search of BO?*

We respond by introducing *generate-and-select* (GAS; Figure 4), a modular framework in which a black-box LLM constructs the search space and conventional BO allocates evaluations within it. At iteration t, the LLM generates r candidates from previous observations and adds them to a cumulative pool. A Gaussian-process (GP) acquisition rule selects q candidates for evaluation, while unselected candidates remain available as the surrogate changes. This decouples the proposal and evaluation budgets: when generation is inexpensive, choosing $r > q$ lets BO consider more candidates without additional objective evaluations.

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.

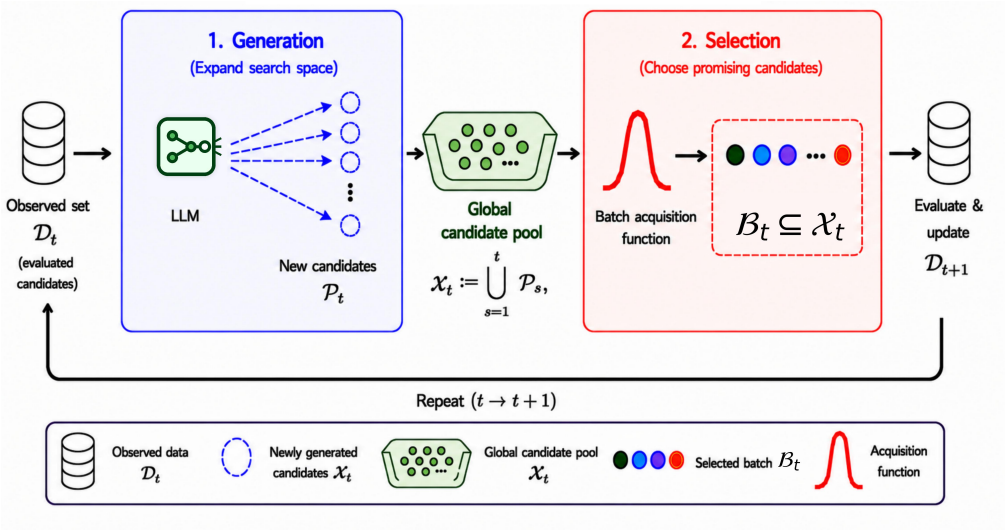


Figure 1: Overview of GAS. An LLM adds structured candidates to a cumulative pool, a conventional BO acquisition rule selects candidates for evaluation, and the resulting observations update the surrogate and inform subsequent generation.

Generation and selection have distinct roles but form a single adaptive process. Each evaluation updates the surrogate and the history that conditions later generation. Selection therefore affects not only which available candidate is evaluated, but also which candidates the LLM may propose next. Leaving a useful candidate unevaluated can delay both improvement of the incumbent and the feedback needed to discover better solutions.

Our analysis formalizes this coupling. Simple regret relative to the best generator-reachable value decomposes exactly into generator regret—the gap left by the available pool—and selection regret—the improvement missed by evaluation selection. Under sufficient conditions, we derive a finite-budget bound and prove that expected simple regret vanishes if and only if average expected missed improvement vanishes. A GP Thompson sampling specialization further ties the selection term to information gain over the generator-reachable domain.

We evaluate GAS on semantic search, molecular optimization, symbolic regression, and 3D Ackley. Across all four tasks, GAS matches or outperforms direct LLM optimizers and BO–LLM methods, with statistically significant gains over the strongest baseline on three. It achieves comparable or better performance than the closest BO–LLM baseline while using over two orders of magnitude fewer LLM tokens. Comparisons with standard BO clarify the intended regime: direct search is effective when a high-quality candidate domain is available, whereas adaptive generation can reach useful objects excluded from a fixed candidate set.

Contributions.

- **Framework.** We formulate a modular generate-and-select framework in which an adaptive black-box LLM constructs an evolving candidate domain and a conventional BO acquisition rule selects evaluations from a cumulative pool. This separates the proposal budget from the evaluation budget and allows earlier proposals to be reconsidered as the surrogate changes.
- **Analysis.** We derive a finite-budget simple-regret bound that couples generator improvement opportunities with cumulative missed improvements in selection. The result characterizes convergence to the best generator-reachable value and yields a GP–TS specialization with information gain measured on the reachable domain.
- **Evidence.** Across four optimization tasks, GAS matches or outperforms direct LLM optimizers and BO–LLM methods, with statistically significant gains over the strongest baseline on three. It achieves comparable or better performance than the closest BO–LLM baseline while using over two orders of magnitude fewer LLM tokens.

2 Related Work

Related methods differ in how candidates enter the search and what role the LLM plays. GAS separates two ideas from this literature: an LLM adaptively expands a candidate pool from evaluated history, while a conventional BO acquisition rule allocates evaluations within it. We compare this design with direct LLM optimizers, BO–LLM hybrids. Finally, we briefly discuss non-LLM BO methods for structured domains.

LLMs as Black-Box Optimizers LLMs have been used as derivative-free optimizers through iterative prompting, evolutionary operations, or combinations of the two (Yang et al., 2024; Brahmachary et al., 2025; Lange et al., 2024). These methods condition on a task description and previous evaluations, then generate the candidates queried in the next iteration. Direct generation supports heterogeneous structured representations without enumerating the domain or decoding from a separate search space. Candidate construction and evaluation allocation occur in the same step, since generated proposals are typically passed directly to the objective. GAS retains adaptive generation of structured objects but separates proposal from evaluation. It can generate more candidates than it evaluates and use surrogate uncertainty to allocate the evaluation budget.

LLMs within Bayesian Optimization BO–LLM methods can first be divided by whether the design space is directly accessible. LLM Processes use an LLM as a probabilistic surrogate, while LLINBO combines LLM recommendations with statistical experts (Requeima et al., 2024; Chang et al., 2025). Both retain acquisition optimization over an explicit domain: query locations can be specified independently of the LLM. They therefore do not address the candidate-construction problem considered here, where BO can operate only on candidates supplied by the generator.

Among methods that use an LLM to construct candidates, LLAMBO is our closest baseline. It uses LLMs for warm-starting, candidate sampling, and predictive modeling in hyperparameter optimization (Liu et al., 2024). GAS instead restricts the LLM to candidate generation and uses a GP for prediction and acquisition. This division retains conventional BO uncertainty estimates and limits its LLM inference to proposal generation. Across our experiments, LLAMBO consumes over two orders of magnitude more LLM tokens than GAS (Table 3).

BOPRO and BO–LLM provide two other candidate-generation interfaces. BOPRO optimizes acquisition in a continuous embedding space and conditions LLM generation on points near the latent optimizer (Agarwal et al., 2025). GAS evaluates acquisition only after generation, ranking realized proposals and retaining them in a cumulative pool. BO–LLM also applies GP-based selection to candidates obtained by expanding high-performing prompts with an LLM (Ballew et al., 2025). Its generator, representation, and evaluation procedure are designed for prompt classification. GAS provides a modular generate-and-select interface across heterogeneous structured domains and analyzes the feedback between its two components.

Bayesian Optimization over Structured Domains Structured BO methods make non-Euclidean domains searchable through specialized representations and acquisition optimizers. Kernel-based approaches pair structured kernels with domain-specific acquisition optimization (Moss et al., 2020). Latent-space approaches optimize in learned continuous representations and decode the resulting points into structured objects (Gómez-Bombarelli et al., 2018; Maus et al., 2022). Both rely on search machinery suited to the domain, such as a structured kernel, generative model, training corpus, or decoder. In GAS, the LLM constructs the candidates, and the representation is used only to model and rank realized proposals. The resulting adaptive BO search domain does not require acquisition optimization over the full structured space or decoding arbitrary latent points.

Theory for generated search spaces. The candidate pool in GAS changes with the observed history. Our GP–TS specialization draws on posterior-sampling analyses for time-varying action sets (Russo & Van Roy, 2014). Guided-diffusion BO also studies optimization through generated candidates, but fine-tunes a pretrained sampler toward high-acquisition regions. Its regret analysis tracks probability-mass lift on near-optimal sets, score-learning error, and sampler approximation (Adachi et al., 2026). We instead treat the generator as a black box. Our bound links incumbent-relative improvement opportunities in its cumulative pool to missed improvements in selection, and the GP–TS specialization measures complexity on the generator-reachable domain.

3 PROBLEM SETTING

We consider maximizing an expensive black-box function $f : \mathcal{X} \to \mathbb{R}$ over a structured domain $\mathcal{X}$, such as words, molecules, or symbolic expressions. An evaluation of $x \in \mathcal{X}$ produces $y = f(x)+\varepsilon$, where ε denotes observation noise.

Optimization starts from a dataset $\mathcal{D}_0$ of size n_0 and proceeds for T iterations, evaluating a batch of q candidates per iteration. After t iterations, the dataset is $\mathcal{D}_t = \{(x_i, y_i)\}_{i=1}^{n_t}$, with $n_t = n_0 + tq$. Performance is measured by simple regret,

$$r_t = f(x^\star) - \max_{i \le n_t} f(x_i), \qquad x^\star \in \arg\max_{x \in \mathcal{X}} f(x),$$

with the objective of minimizing r_T.

The structured domain may be too large or irregular to enumerate or search directly, and arbitrary elements may be invalid or implausible. Candidates are instead accessed through a generative model $\mathcal{G}$. At iteration t, the generator receives a context $\mathcal{C}_t$ and returns r candidates,

$$\mathcal{P}_t = \mathcal{G}(\mathcal{C}_t; r), \qquad \mathcal{P}_t \subseteq \mathcal{X}.$$

The context may depend on the task description and the observed history $\mathcal{D}_t$. We instantiate $\mathcal{G}$ with an LLM and treat it as a stochastic black box.

An optimization method under this interface specifies the context rule, the number of proposals per iteration, the candidate-retention policy, and the rule used to select each evaluation batch. The next section defines these choices for GAS.

4 GENERATE AND SELECT

GAS separates candidate construction from evaluation allocation. A generator $\mathcal{G}$ proposes structured candidates, and a pool-based selector $\mathcal{A}_t$ chooses which candidates to evaluate at iteration t. Both components use the observed history: generation through its context and selection through $\mathcal{A}_t$.

Let $\mathcal{D}_t$ and $\mathcal{X}_t$ denote the evaluated dataset and cumulative candidate pool at the start of iteration t, respectively, with $\mathcal{X}_0 = \{x : (x, y) \in \mathcal{D}_0\}$. Let η collect the task description, constraints, and other task-side information, and let τ be the rule that combines this information with $\mathcal{D}_t$ to construct a generation context. The selector $\mathcal{A}_t$ is based on $\mathcal{D}_t$ and maps an available pool to a batch of q candidates. We assume $|\mathcal{D}_t| \ge k$ at every iteration. Each iteration proceeds as follows:

1. **Generate.** Form $\mathcal{C}_t = \tau(\eta, \mathcal{D}_t)$. In our implementation, τ includes the k inputs from $\mathcal{D}_t$ having the highest values, ordered from worst to best; their numerical values are omitted. Query the generator for r proposals, $\mathcal{P}_t = \mathcal{G}(\mathcal{C}_t; r)$.
2. **Select.** Add the proposals to the cumulative pool and select a batch:

   $$\mathcal{X}_{t+1} = \mathcal{X}_t \cup \mathcal{P}_t, \qquad \mathcal{B}_t = \mathcal{A}_t(\mathcal{X}_{t+1}; q).$$

   The pool retains evaluated and unevaluated candidates; previously evaluated points remain eligible for selection.
3. **Evaluate and update.** Evaluate each $x \in \mathcal{B}_t$ as $y_{t,x} = f(x) + \varepsilon_{t,x}$ and update

   $$\mathcal{D}_{t+1} = \mathcal{D}_t \cup \{(x, y_{t,x}) : x \in \mathcal{B}_t\},$$

   where $\varepsilon_{t,x}$ denotes observation noise. The updated pair $(\mathcal{D}_{t+1}, \mathcal{X}_{t+1})$ defines the state at the start of the next iteration.

The analysis in Section 5 treats $\mathcal{A}_t$ as a generic pool-based selector. In the experiments, $\mathcal{G}$ is an LLM and $\mathcal{A}_t$ is either batch Thompson sampling (qTS) or batch noisy expected improvement (qNEI) under a GP fitted to representations $\phi : \mathcal{X} \to \mathbb{R}^d$. The GP uses ϕ to model the generated candidates, and acquisition is evaluated directly over the finite pool. Optimization in the ambient embedding space and decoding of arbitrary vectors are not required.

5 THEORETICAL ANALYSIS

Simple regret decomposes into a gap between the best reachable value and the best available candidate, and a gap between that candidate and the best evaluated solution. We analyze the feedback between these generator and selection terms, first for an arbitrary selector and then for GP Thompson sampling. Formal details and proofs appear in Appendix 10.

5.1 SETTING AND BENCHMARK

We consider the sequential, noise-free setting: $q = 1$ and evaluating x reveals $f(x)$ exactly. The objective is drawn from a prior $f \sim \Pi$, conditional on any task-side information η. All expectations below average over the objective and the randomness of generation and selection. We assume $\mathbb{E}[\sup_{x\in\mathcal{X}} |f(x)|] < \infty$, so that the expected optimization gaps are finite.

Let $\mathcal{F}_t$ denote the observed history immediately before generation at round t, including η, the evaluated dataset $\mathcal{D}_t$, and all previous proposals and selections. The generator samples a batch $\mathcal{P}_t$ of at most r candidates using a context constructed from the observed data. We impose no independence assumption within a batch or across rounds. Starting from the inputs in the nonempty initial dataset, $\mathcal{X}_0$, the pool evolves as $\mathcal{X}_t = \mathcal{X}_{t-1} \cup \mathcal{P}_t$. The selector chooses $x_t \in \mathcal{X}_t$ and adds its observation to $\mathcal{D}_t$. Define the incumbent as $m_t := \max_{(x,f(x))\in\mathcal{D}_t} f(x)$. Thus $m_{t+1} = \max\{m_t, f(x_t)\}$ and $\max_{x\in\mathcal{X}_t} f(x) \geq m_{t+1}$, since the pool contains all previously evaluated inputs.

Let $\mathcal{X}_{\text{reach}}$ contain the initial inputs and candidates attainable through finite histories of generation and evaluation with positive-probability proposals, allowing any admissible selector. Write $f^\dagger = \sup_{x\in\mathcal{X}_{\text{reach}}} f(x)$ and define

$$r_t := f^\dagger - m_t.$$

Here r_t denotes simple regret before round t, while r denotes the proposal batch size. Our performance measure is the Bayesian simple regret $\mathbb{E}[r_{T+1}]$ after T evaluations. The benchmark $f^\dagger$ may depend on the objective and initial state; it need not be attained or equal the optimum over the unrestricted domain.

5.2 REGRET DECOMPOSITION AND GENERATOR IMPROVEMENT

After round t, simple regret decomposes exactly as

$$r_{t+1} = \underbrace{f^\dagger - \max_{x\in\mathcal{X}_t} f(x)}_{\text{generator regret}} + \underbrace{\max_{x\in\mathcal{X}_t} f(x) - m_{t+1}}_{\text{selection regret } \ell_t}. \tag{1}$$

Generator regret measures the quality of the available candidates relative to the benchmark. Selection regret measures how much better the incumbent could have been using those candidates. Both components depend on the full history of generation and selection. In particular, missing a useful candidate can also delay the feedback needed to generate better candidates, delaying reductions in generator regret. The decomposition identifies where the current gap remains; it does not separate the two mechanisms into independent sources of error.

Let $L_T := \sum_{t=1}^T \ell_t$ denote cumulative selection regret, or equivalently cumulative missed improvement. It is controlled by the selector's usual cumulative pool regret:

$$L_T \leq R_T^{\text{pool}} := \sum_{t=1}^{T} \left(\max_{x\in\mathcal{X}_t} f(x) - f(x_t) \right). \tag{2}$$

The distinction is that ℓ_t compares against the updated incumbent, whereas pool regret compares against the point evaluated at round t.

Generator improvement assumption. With probability at least p, the pool offers an improvement that closes an α fraction of the remaining gap. Formally, there exist $\alpha \in (0, 1)$ and $p \in (0, 1]$ such that, at every round with $r_t > 0$,

$$\mathbb{P}\left(\max_{x\in\mathcal{X}_t} f(x) \geq m_t + \alpha r_t \mid \mathcal{F}_t, f \right) \geq p \tag{3}$$

almost surely. The constants are uniform over the histories under consideration. Conditioning on f specifies a requirement for almost every objective realization; it does not give the generator access to unobserved objective values.

As the incumbent improves, the required objective value of this opportunity increases. The condition is imposed on the accumulated pool: it holds automatically when a sufficiently good candidate is already stored, so the generator need not keep improving while useful candidates await selection. A corresponding condition on the newly generated batch alone would be sufficient but stronger. The assumption allows arbitrary data-dependent contexts and dependent proposals; p describes the success probability of a batch as a whole. It is a substantive condition on the generator's evolution, not a property guaranteed by using an LLM or by increasing r.

5.3 Main result

Theorem 1 (Generation–selection bound) *Under the setting above and the generator improvement assumption equation 3, every selector satisfies, for all $T \geq 1$,*

$$\mathbb{E}[r_{T+1}] \leq C_1 \mathbb{E}[r_1] \exp(-c\alpha p T) + \frac{C_2}{\alpha} \frac{\mathbb{E}[L_T]}{T}. \tag{4}$$

Here $C_1, C_2, c > 0$ are absolute constants.

The first term shows how successive improvement opportunities reduce the initial expected gap. The second captures the selector's failure to realize available improvements. In particular, p affects the discovery time scale but does not multiply the selection term by $1/p$. This separation follows from pool persistence: once a useful candidate is generated, delaying its use incurs missed improvement on subsequent rounds, without requiring another successful proposal.

The uniform improvement probability yields the exponential term. Allowing this probability to decrease at finer accuracy levels still guarantees convergence under sublinear selection loss, but potentially at a slower rate (Appendix 10.5).

Convergence and stagnation. An ideal selector with $L_T = 0$ achieves an exponentially decreasing bound. More generally, expected simple regret converges to zero if and only if average expected missed improvement vanishes, $\mathbb{E}[L_T]/T \to 0$. Thus, under the improvement assumption, persistent stagnation below the benchmark entails a nonvanishing average missed improvement. The generator may stop proposing better points while a useful candidate remains unselected, but the joint process cannot remain suboptimal when average missed improvement vanishes.

By equation 2, any Bayesian cumulative pool-regret bound can replace $\mathbb{E}[L_T]$ in the theorem. Sublinear pool regret is therefore sufficient for convergence, although it is not necessary because L_T can be smaller.

5.4 GP Thompson sampling specialization

Fix the task information η and suppose the objective follows a correctly specified GP prior $f \mid \eta \sim \mathcal{GP}(\mu, k)$, with a fixed, known kernel satisfying $k(x, x) \leq 1$. The kernel may act on the embeddings $\phi(x)$. Assume the generator uses only the observed history and randomness independent of f, so that proposals reveal no additional information about f given $\mathcal{F}_t$. At each round, Thompson sampling draws an independent function from the posterior and maximizes it over the realized pool $\mathcal{X}_t$.

Let $n_0 = |\mathcal{X}_0|$. For a candidate domain $A \subseteq \mathcal{X}$, the quantity $\gamma_T(A)$ measures the kernel complexity over T evaluations within A:

$$\gamma_T(A) := \frac{1}{2} \sup_{z_1, \ldots, z_T \in A} \log\det\big(I + [k(z_i, z_j)]_{i,j=1}^{T}\big)\,. \tag{5}$$

This is the usual GP information-gain quantity with unit reference noise; the reference noise only defines the complexity measure, while observations and posterior updates remain noise-free.

Selection takes place within $\mathcal{X}_t \subseteq \mathcal{X}_{\text{reach}} \subseteq \mathcal{X}$. Combining the posterior-sampling pool-regret argument of Russo & Van Roy (2014) with Theorem 1 yields

$$\mathbb{E}[r_{T+1}] \leq C_1 \mathbb{E}[r_1] e^{-c\alpha p T} + \frac{C_3}{\alpha} \sqrt{\frac{\log\big(2(n_0 + rT)T\big)}{T}}\, \mathbb{E}\Big[\sqrt{\gamma_T(\mathcal{X}_{\text{reach}})}\Big]\,, \tag{6}$$

Table 1: Experimental protocol. The final column gives the evaluation and generation batch sizes (q, r) used by GAS.

Task	Reported metric	Runs	Budget	(q, r)
Semantle	Best similarity ↑	50	400	$(10, 30)$
MolOpt	Best score ↑	30	200	$(5, 20)$
SymReg	Normalized MSE ↓	30	200	$(10, 40)$
Ackley-3D	Simple regret ↓	30	50	$(1, 10)$

where C_3 is absolute. The expectation accounts for the possible dependence of the reachable set on the objective and initial state. Since $\gamma_T(\mathcal{X}_{\text{reach}}) \leq \gamma_T(\mathcal{X})$, selection can benefit from a smaller domain with lower kernel complexity. The bound therefore reflects both the improvement opportunities supplied by generation and the complexity of selecting among reachable candidates.

6 Experimental Evaluation

6.1 Experimental setup

We consider three structured tasks—Semantle word search, molecular optimization with DockString, and symbolic regression—and the controlled Ackley-3D benchmark. Table 1 summarizes the protocol. All methods use the same warm starts and evaluation budget within each matched run.

All LLM-based methods use Qwen 3.7 Flash. GP-based methods use a Matérn-$5/2$ kernel with marginal-likelihood hyperparameter fitting. The representations are gte-Qwen2-1.5B-instruct embeddings for words, MolFormer embeddings for molecules, expression values on a fixed grid for SymReg, and normalized coordinates for Ackley-3D. We report means and standard errors over matched runs; Appendix 11 provides further details. We compare with the direct LLM optimizers OPRO (Yang et al., 2024) and LLM-ES (Menet et al., 2026); the BO–LLM methods BOPRO (Agarwal et al., 2025) and LLAMBO (Liu et al., 2024); and standard GP-based BO over a fixed candidate set. The fixed set is the discretized grid for Ackley-3D and the $10{,}000$ most frequent eligible `wordfreq` entries with Google News Word2Vec embeddings for Semantle.[1] Because the methods use the LLM differently, we equalize objective evaluations rather than LLM calls and report token usage separately. Table 2 compares GAS (qNEI) with the strongest baseline on matched instances; the three structured-task p-values are jointly Holm-adjusted.

6.2 Overall performance and efficiency

Table 2 shows that GAS (qNEI) achieves the best reported mean on all four tasks. Its advantage over the strongest baseline is significant on Semantle, MolOpt, and SymReg (Holm-adjusted $p = 0.0034$ for each). On Ackley-3D, GAS and standard BO reach zero final regret on every seed, but GAS (qNEI) incurs roughly half the cumulative regret: 10.47 ± 1.78 versus 20.82 ± 2.82 ($p = 0.0008$). Standard BO succeeds when its fixed domain contains the optimum, whereas adaptive generation can reach candidates excluded from a restricted fixed set, as on Semantle. qNEI and qTS have the same reported final mean on Semantle and Ackley-3D; qNEI performs better on MolOpt and SymReg. Figure 2 shows the full trajectories.

Table 3 reports prompt and completion tokens rather than LLM calls. LLAMBO consumes 104–538 times as many tokens as GAS (qNEI), while attaining a worse reported final mean on every task.

6.3 Controlled regret decomposition

We compute both terms in equation 1 exactly on Ackley-3D using GAS (qTS), 30 seeds, $q = 1$, and $r = 10$. We compare the full 21^3 grid with two 11^3 boxes: one contains the optimum, while the other excludes it and has a reachability gap of 3.625 (Figure 3). On the full support, the first generation round reduces generator regret from 2.967 ± 0.136 to 1.011 ± 0.239. With $\alpha = 0.5$, the improvement event in equation 3 occurs in 77% of observed eligible rounds on the full support

[1] `https://pypi.org/project/wordfreq/`

Table 2: Final objective after the full evaluation budget (mean ± SE). Bold denotes the best reported mean in each column. Semantle and MolOpt report the best score found. SymReg reports capped normalized MSE, $\min(\mathrm{MSE}/\mathrm{Var}(y), 1)$, and Ackley-3D reports simple regret. Holm-adjusted p-values compare GAS (qNEI) with the strongest baseline on each structured task.

Method	Semantle ↑	MolOpt ↑	SymReg ↓	Ackley-3D ↓
GAS (qTS)	$\mathbf{0.955 \pm 0.015}$	0.828 ± 0.014	0.197 ± 0.052	$\mathbf{0.000 \pm 0.000}$
GAS (qNEI)	$\mathbf{0.955 \pm 0.015}$	$\mathbf{0.872 \pm 0.011}$	$\mathbf{0.123 \pm 0.040}$	$\mathbf{0.000 \pm 0.000}$
OPRO	0.865 ± 0.023	0.824 ± 0.013	0.302 ± 0.062	1.520 ± 0.274
BOPRO	0.895 ± 0.020	0.761 ± 0.014	0.351 ± 0.058	2.576 ± 0.142
LLAMBO	0.873 ± 0.025	0.803 ± 0.014	0.251 ± 0.055	0.637 ± 0.198
LLM-ES	0.870 ± 0.021	0.614 ± 0.027	0.428 ± 0.071	1.040 ± 0.161
Standard BO	0.817 ± 0.018	0.774 ± 0.013	0.318 ± 0.061	$\mathbf{0.000 \pm 0.000}$
Strongest baseline	BOPRO	OPRO	LLAMBO	Standard BO
Holm-adjusted p	**0.0034**	**0.0034**	**0.0034**	–

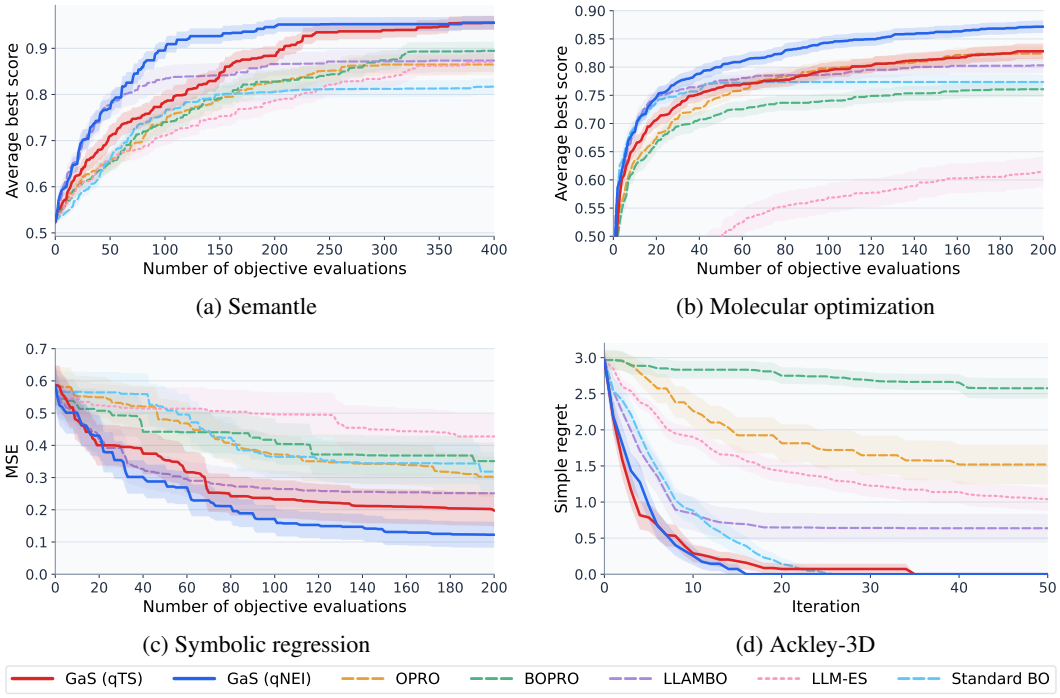


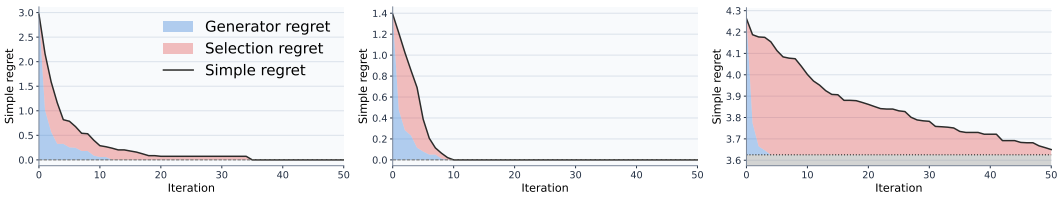
Figure 2: Best observed performance versus black-box evaluations. Higher is better for Semantle and molecular optimization; lower is better for symbolic regression and Ackley-3D. Shading denotes one standard error across runs.

and 76% on the small support. Selection regret then accounts for most of the remaining gap, while L_T/T decreases from 0.57 at $T = 10$ to 0.17 at $T = 50$ and simple regret reaches zero on all seeds.

Restricting the domain to the small box reduces cumulative selection regret from 8.30 ± 2.45 to 3.27 ± 0.63 and reaches zero simple regret on every seed by round 10 rather than round 20, consistent with the dependence on $\gamma_T(\mathcal{X}_{\mathrm{reach}})$. On the box that excludes the optimum, generator regret reaches the support gap by round 5, while selection regret decreases to 0.025 ± 0.021. This illustrates convergence to $f^\dagger$; no selector restricted to this support can close the remaining gap.

6.4 Ablations

We ablate candidate count, context size and composition, selection rule, batch size, adaptive generation, and LLM scale on Semantle and symbolic regression. Each comparison changes one factor



(a) Full support (b) Small support (c) Support excluding the optimum

Figure 3: Simple-regret decomposition for GAS (qTS) on Ackley-3D over full, small, and optimum-excluding supports (30 seeds). Blue denotes generator regret, red denotes selection regret, and gray denotes the reachability gap on the optimum-excluding support.

within a matched configuration. Generating the pool only once lowers the Semantle score from 0.955 to 0.843 and raises normalized MSE from 0.123 to 0.318. Replacing acquisition-based selection with uniform selection lowers the Semantle score from 0.944 to 0.887 and yields an MSE of 0.266. Increasing r from 20 to 40 improves the two metrics from 0.948 to 0.968 and from 0.260 to 0.123, respectively, without additional evaluations.

Context construction also matters: performance varies nonmonotonically with context size, and replacing the top-10 examples with random examples or the full history lowers the Semantle score to 0.822 and 0.689. Performance is comparatively stable over $q \in \{5, 10, 15\}$ and across the tested LLM scales. Appendix 8.1 provides full curves and standard errors.

7 Conclusion

We introduced generate-and-select (GAS), a modular framework in which a black-box LLM constructs an evolving candidate pool and conventional BO selects costly evaluations. Our analysis decomposes simple regret relative to the best generator-reachable value into generator and selection terms. Under a recurring-improvement condition on the pool, it yields a finite-budget bound and characterizes convergence through average missed improvement; a GP–TS specialization ties selection complexity to information gain over the generator-reachable domain.

Across four structured and synthetic tasks, GAS attains the best reported final mean, with significant gains over the strongest baseline on three. It also achieves better reported means than the closest BO–LLM baseline while using over two orders of magnitude fewer LLM tokens. The ablations support the roles of adaptive generation, acquisition-based selection, and their feedback.

The method remains limited by its generator: it cannot recover solutions outside the generator-reachable domain, and poor coverage can constrain performance. The guarantees further assume recurring improvement opportunities and analyze sequential, noise-free selection; the GP–TS result additionally assumes a correctly specified GP, whereas our experiments use batches, noisy observations, and fitted hyperparameters. Finally, acquisition values rank candidates only after generation and do not guide proposals within a round. Extending the theory to the empirical setting and using selection uncertainty to improve generator coverage without sacrificing modularity are important directions for future work.

AI use statement

In this work, we used generative AI tools for writing and debugging code. Specifically, AI coding assistants (Claude Code, Anthropic; Codex, OpenAI) helped implement experiment scripts, adapters that run existing baselines (e.g., LLAMBO and LLM-ES) on our tasks, plotting and table-generation scripts, and unit tests, and helped launch and monitor experiment runs. We also used them to discuss implementation and experimental-design choices, such as which ablations and diagnostic analyses to run. The research questions, the proposed method, and the claims of the paper were developed by the authors. We additionally used an AI assistant (Claude, Anthropic) to review drafts for internal consistency, verify bibliography entries, polish grammar and phrasing, and draft or revise descriptive text such as figure captions and appendix paragraphs summarizing our tables; all such text was checked by the authors against the underlying results. The use of generative AI to create or annotate

data is not applicable to this work: all benchmark tasks are taken from existing sources or defined programmatically.

Separately from AI assistance, LLMs are part of the methods we study: Qwen3-14B, Qwen 3.7 Flash, and Qwen 3.8 Max serve as candidate generators in our method and in the LLM-based baselines. These models are experimental components, not tools used to prepare the paper; their configuration and usage are reported in Section 6 and Appendix 8.1.

We have reviewed all AI-assisted work. All AI-assisted code was read and tested by the authors: baseline implementations were checked against the original papers and reference code, unit tests were run, and intermediate outputs (per-seed logs and optimization traces) were inspected before any reported figure or table was produced. All reported numbers are generated directly by our experiment scripts, not transcribed by hand or by an AI tool. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

Ethics statement

This work does not involve human subjects, personal data, or new data collection. All tasks use existing public benchmarks or programmatically defined objectives (Semantle, symbolic regression, DockString molecular docking, and synthetic test functions), and LLMs are accessed through a commercial API without any user data. Our molecular optimization experiments optimize computed drug-likeness and docking scores only; no compounds are synthesized or tested, and docking scores are a coarse proxy for biological activity. As with any black-box molecular optimization method, GaS could in principle be applied to objectives that reward harmful compounds; we do not expect it to provide meaningful uplift beyond existing generative chemistry tools, and any practical use would still require experimental validation. Finally, because GaS searches only within the support of an LLM generator, it may inherit and amplify the generator's biases by concentrating search in limited regions of the solution space.

References

David H. Ackley. *A Connectionist Machine for Genetic Hillclimbing*. Kluwer Academic Publishers, Boston, MA, 1987.

Masaki Adachi, Anita Yang, Yakun Wang, and Song Liu. Regret analysis of guided diffusion for black-box optimization over structured inputs, 2026. URL https://arxiv.org/abs/2605.10385.

Dhruv Agarwal, Manoj Ghuhan Arivazhagan, Rajarshi Das, Sandesh Swamy, Sopan Khosla, and Rashmi Gangadharaiah. Searching for optimal solutions with LLMs via Bayesian optimization. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=aVfDrl7xDV.

Adam Ballew, Jingbo Wang, and Shaogang Ren. LLM based Bayesian optimization for prompt search. *arXiv preprint arXiv:2510.04384*, 2025. URL https://arxiv.org/abs/2510.04384.

Shuvayan Brahmachary, Subodh M. Joshi, Aniruddha Panda, Kaushik Koneripalli, Arun Kumar Sagotra, Harshil Patel, Ankush Sharma, Ameya D. Jagtap, and Kaushic Kalyanaraman. Large language model-based evolutionary optimizer: Reasoning with elitism. *Neurocomputing*, 622:129272, 2025. doi: https://doi.org/10.1016/j.neucom.2024.129272. URL https://www.sciencedirect.com/science/article/pii/S0925231224020435.

Chih-Yu Chang, Milad Azvar, Chinedum Okwudire, and Raed Al Kontar. Llinbo: Trustworthy LLM-in-the-loop Bayesian optimization. *arXiv preprint arXiv:2505.14756*, 2025.

Peter I. Frazier. A tutorial on Bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2021.

Miguel García-Ortegón, Gregor N. C. Simm, Austin J. Tripp, José Miguel Hernández-Lobato, Andreas Bender, and Sergio Bacallado. Dockstring: Easy molecular docking yields better benchmarks for ligand design, 2021.

Roman Garnett. *Bayesian Optimization*. Cambridge University Press, 2023.

Rafael Gómez-Bombarelli, Jennifer N. Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D. Hirzel, Ryan P. Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, 4(2):268–276, 2018.

Robert Lange, Yingtao Tian, and Yujin Tang. Large language models as evolution strategies. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 579–582, 2024. doi: 10.1145/3638530.3654238.

Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. Towards general text embeddings with multi-stage contrastive learning. *arXiv preprint arXiv:2308.03281*, 2023.

Tennison Liu, Nicolás Astorga, Nabeel Seedat, and Mihaela van der Schaar. Large language models to enhance Bayesian optimization. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=OOxotBmGol.

Natalie Maus, Haydn Jones, Juston Moore, Matt J Kusner, John Bradshaw, and Jacob Gardner. Local latent space Bayesian optimization over structured inputs. *Advances in Neural Information Processing Systems*, 35:34505–34518, 2022.

Nicolas Menet, Aleksandar Terzic, Michael Hersche, Andreas Krause, and Abbas Rahimi. Thompson sampling via fine-tuning of LLMs. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=u6fc354gfS.

Henry Moss, David Leslie, Daniel Beck, Javier Gonzalez, and Paul Rayson. BOSS: Bayesian optimization over string spaces. *Advances in Neural Information Processing Systems*, 33:15476–15486, 2020.

James Requeima, John Bronskill, Dami Choi, Richard E. Turner, and David Duvenaud. LLM processes: Numerical predictive distributions conditioned on natural language. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/c5ec22711f3a4a2f4a0a8ffd92167190-Abstract-Conference.html.

Jerret Ross, Brian Belgodere, Vijil Chenthamarakshan, Inkit Padhi, Youssef Mroueh, and Payel Das. Large-scale chemical language representations capture molecular structure and properties, 2022. URL https://arxiv.org/abs/2106.09553.

Daniel Russo and Benjamin Van Roy. Learning to optimize via posterior sampling. *Mathematics of Operations Research*, 39(4):1221–1243, 2014.

James Wilson, Frank Hutter, and Marc Deisenroth. Maximizing acquisition functions for Bayesian optimization. *Advances in Neural Information Processing Systems*, 31, 2018.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Bb4VGOWELI.

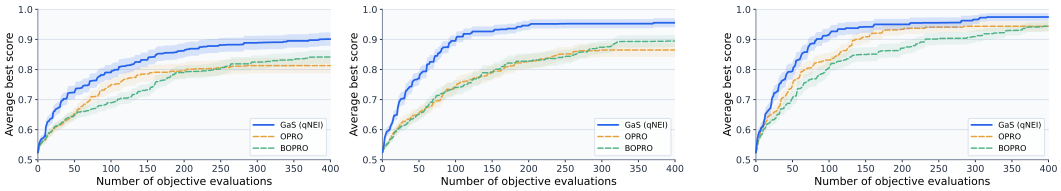
8 Appendix

8.1 Ablation studies

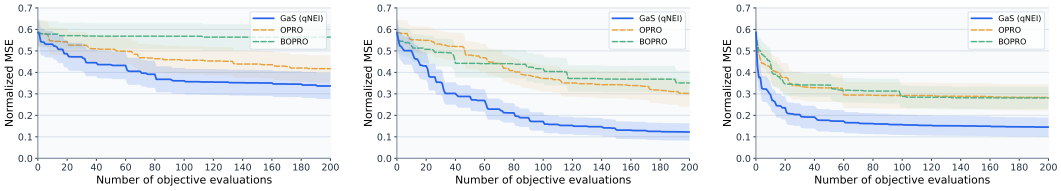
We investigate how the performance of GaS emerges from the interaction between generation and selection. Generation constructs a candidate pool that evolves over time, while selection governs how effectively this pool is exploited. The ablations below examine this interaction: changing selection can also change the feedback used for subsequent generation.

8.1.1 Effect of generator choice

Across three Qwen models of different scale, GaS (qNEI) reaches higher scores on Semantle and lower normalized MSE on symbolic regression than OPRO and BOPRO. With the strongest generator (Qwen 3.8 Max) the final Semantle gap narrows, but GaS still converges faster.



(a) *Semantle*

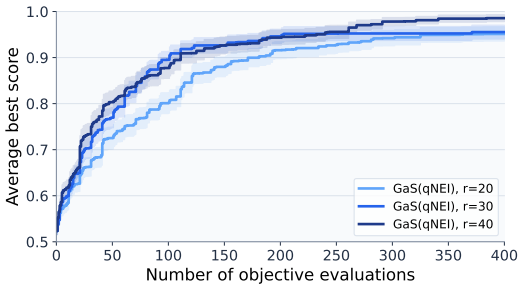


(b) *Symbolic regression*

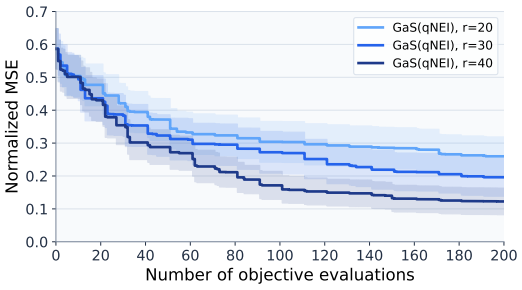
Figure 4: Effect of the generator LLM. Best-so-far average score on Semantle (top; higher is better) and normalized MSE on symbolic regression (bottom; lower is better) versus the number of objective evaluations, for GaS (qNEI), OPRO, and BOPRO with Qwen3-14B (left), Qwen 3.7 Flash (middle), and Qwen 3.8 Max (right).

8.1.2 Effect of generation budget

Increasing the number of generated candidates per iteration improves performance, confirming that GaS benefits from a larger adaptive candidate pool. Gains are substantial when increasing from 20 to 40 candidates, consistent with improved discovery probability.



(a) *Semantle*

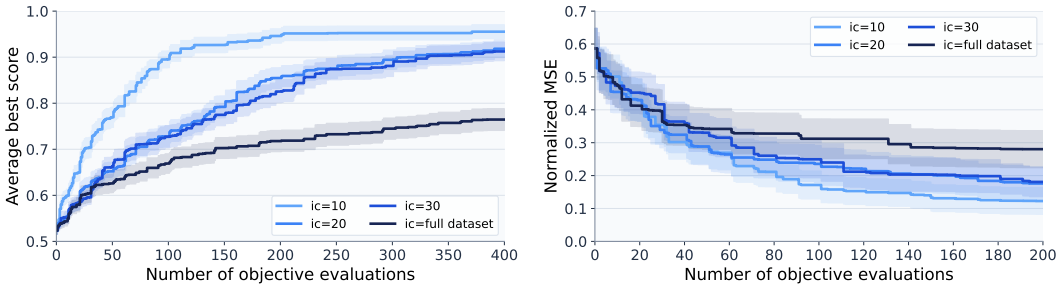


(b) *Symbolic Regression*

Figure 5: Ablation on the candidate pool size per iteration in GaS. All runs use the same black-box evaluation budget; we vary only the number of candidates generated per iteration, comparing GaS with qNEI acquisition function.

8.1.3 Effect of number of in-context exemplars

Performance degrades monotonically as the number of exemplars increases over the tested range, with k=10 performing best on both tasks; using the full history is substantially worse (Semantle: 0.69 vs 0.94). We hypothesize that long prompts dilute the signal from high-scoring exemplars and reduce diversity in generation.

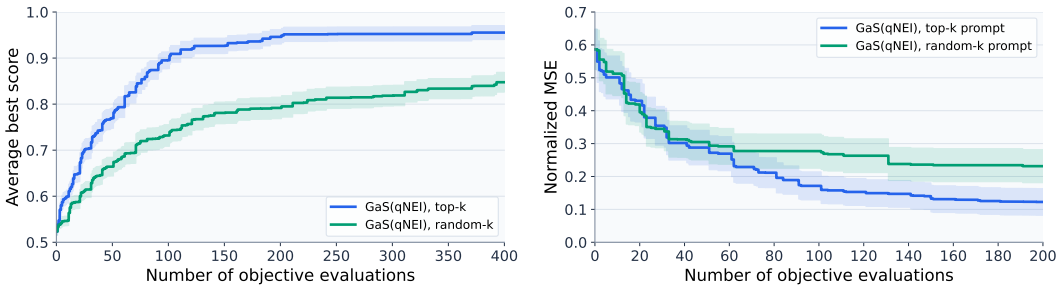


(a) *Semantle* (b) *Symbolic Regression*

Figure 6: Effect of in-context example count in GaS. Curves show best-so-far score (Semantle, higher is better) and normalized MSE (symbolic regression, lower is better) versus black-box evaluations for GaS with qNEI.

8.1.4 TOP-k VERSUS RANDOM EXEMPLARS

We compare the default top-k exemplar selection strategy against randomly selected in-context exemplars. Top-k improves over random exemplar selection, showing that the quality of the prompt context matters. This suggests that GaS benefits not only from generating an adaptive candidate pool, but also from conditioning the generator on informative prior evaluations.

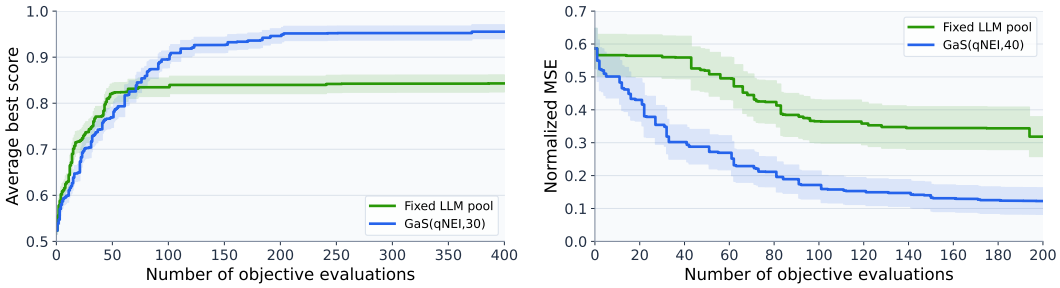


(a) *Semantle* (b) *Symbolic Regression*

Figure 7: Effect of exemplar selection strategy in GaS. Curves show the objective values for GaS with qNEI, comparing top-k exemplar selection against random exemplar selection.

8.1.5 ADAPTIVE VS. FIXED CANDIDATE POOLS

Adaptive candidate generation substantially outperforms selection from a fixed pool, even when the pool is large. This shows that GaS benefits from iteratively refreshing the candidate set rather than simply filtering a large static pool, and that conditioning generation on the evolving dataset is critical for discovering high-quality solutions.

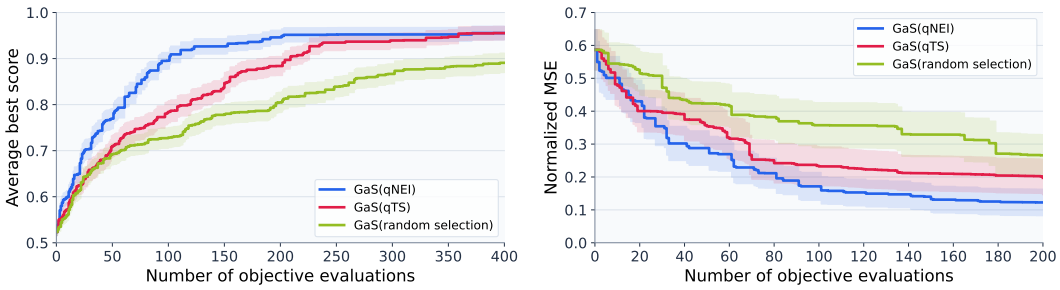


(a) *Semantle (fixed pool = 1200)* (b) *Symbolic Regression (fixed pool = 800)*

Figure 8: Comparison of fixed-pool and adaptive-pool optimization. The fixed-pool baseline selects from an LLM generated pool, while GaS with qNEI adaptively generates and selects.

8.1.6 Effect of acquisition function

GaS with random selection is already comparable to the strongest baselines, indicating that adaptive generation alone yields a strong candidate pool. Acquisition-based selection adds substantial further gains, especially on symbolic regression (normalized MSE 0.27 $\rightarrow$ 0.12). qNEI converges fastest on Semantle, where qTS catches up by the end of the budget, and is best on symbolic regression.

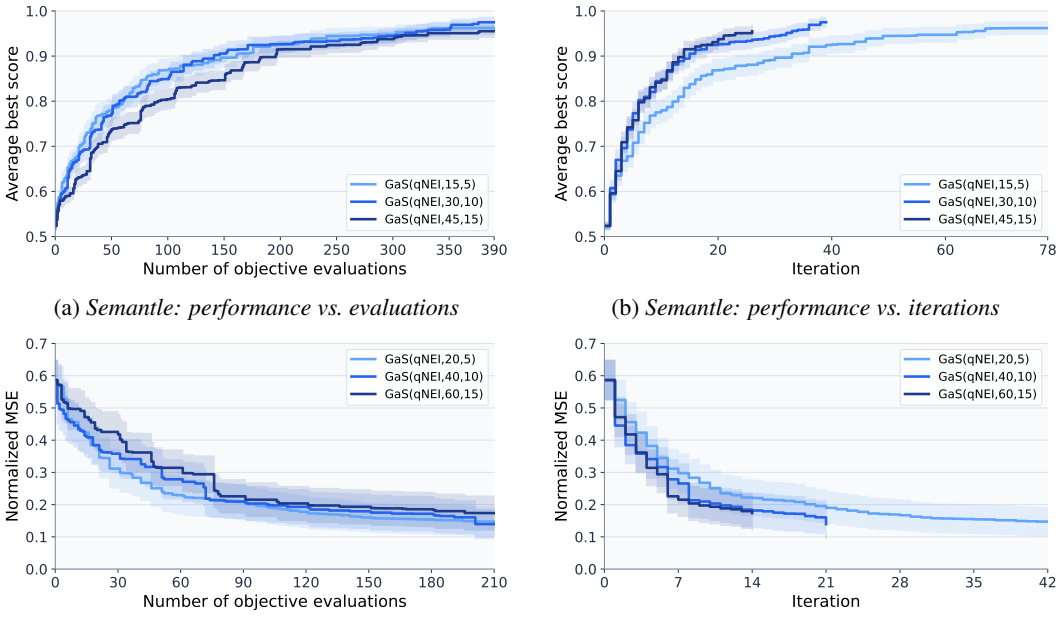


(a) *Semantle* (b) *Symbolic Regression*

Figure 9: Ablation on acquisition functions in GaS.

8.1.7 Effect of evaluation batch size

Smaller per-iteration settings achieve better performance when measured against total objective evaluations, indicating that more frequent feedback improves sample efficiency. Larger settings make more progress per optimization iteration, highlighting a trade-off between sample efficiency and iteration-level efficiency when generation, context size, and evaluation batch size are scaled together.



(a) *Semantle: performance vs. evaluations*

(b) *Semantle: performance vs. iterations*

(c) *Symbolic regression: performance vs. evaluations*

(d) *Symbolic regression: performance vs. iterations*

Figure 10: Scaling per-iteration parameters in GaS on Semantle (top) and symbolic regression (bottom).

9 Acquisition Rules

In this section, we provide detailed definitions and implementations of the acquisition rules used in our experiments. Recall that at iteration t, the surrogate model induces an acquisition rule

$$\pi_t(\cdot \mid \mathcal{D}_t),$$

which selects a batch $\mathcal{B}_t \subseteq \mathcal{X}_t$ of size q based on the current dataset. We consider two instantiations: q-noisy expected improvement (qNEI) and batch Thompson sampling (qTS).

9.1 Batch noisy expected improvement (qNEI)

In the presence of observation noise, expected improvement is defined relative to the posterior distribution over latent function values at previously evaluated points. Let $\mathbf{f}_t = (f(x_i))_{i=1}^{n_t}$ denote the latent function values corresponding to the dataset $\mathcal{D}_t$, and let $\mathbf{f}_\mathcal{B} = (f(x))_{x\in\mathcal{B}}$ denote the latent values at a candidate batch $\mathcal{B}$.

The q-noisy expected improvement is defined as

$$\alpha_{\mathrm{qNEI}}(\mathcal{B}; \mathcal{D}_t) := \mathbb{E}\left[\max\Big(\max_{x\in\mathcal{B}} f(x) - \max_{i\le n_t} f(x_i),\, 0\Big) \;\middle|\; \mathcal{D}_t\right],$$

where the expectation is taken with respect to the joint posterior over $\mathbf{f}_t$ and $\mathbf{f}_\mathcal{B}$.

Batch construction. Selecting the optimal batch requires solving

$$\mathcal{B}_t \in \operatorname*{arg\,max}_{\substack{\mathcal{B}\subseteq\mathcal{X}_t \\ |\mathcal{B}|=q}} \alpha_{\mathrm{qNEI}}(\mathcal{B}; \mathcal{D}_t),$$

which is combinatorial. We approximate this using a greedy procedure: starting from the empty set, we iteratively add the point that maximizes the marginal gain in acquisition value conditioned on previously selected points. This approach is justified by the submodularity of qNEI, which yields a constant-factor approximation guarantee relative to the optimal batch (Wilson et al., 2018).

9.2 BATCH THOMPSON SAMPLING (QTS)

Thompson sampling defines a stochastic acquisition rule based on sampling from the surrogate posterior. At iteration t, we draw q independent sample functions

$$\tilde{f}_1, \ldots, \tilde{f}_q \overset{\text{i.i.d.}}{\sim} p(f \mid \mathcal{D}_t),$$

and use each sampled function to select one point from the candidate pool $\mathcal{X}_t$.

Batch construction. To construct a batch of size q, we use the standard independent-sample batch Thompson sampling procedure:

1. Initialize $\mathcal{B}_t \leftarrow \emptyset$.
2. Draw independent posterior samples
$$\tilde{f}_1, \ldots, \tilde{f}_q \overset{\text{i.i.d.}}{\sim} p(f \mid \mathcal{D}_t).$$
3. For $i = 1, \ldots, q$:
   - Select
   $$x_i \in \arg\max_{x \in \mathcal{X}_t \setminus \mathcal{B}_t} \tilde{f}_i(x),$$
   - Update $\mathcal{B}_t \leftarrow \mathcal{B}_t \cup \{x_i\}$.

10 THEORETICAL DETAILS AND PROOFS

This appendix makes the setup in Section 5 precise, proves the generation–selection bound and its GP–Thompson sampling specialization, and discusses their interpretation. The decomposition is an accounting identity for the current gap. The bounds analyze generation and selection jointly, including their interaction through feedback.

10.1 PROBABILITY MODEL, HISTORIES, AND REACHABILITY

We take $\mathcal{X}$ to be countable, as for finite strings, programs, or discrete molecular representations. All maps and probability kernels are measurable. Fix the task information η and draw f from its conditional prior. The initial dataset $\mathcal{D}_1$ is finite and nonempty, and contains exact evaluations. Its input set is $\mathcal{X}_0$. The generator returns a set $\mathcal{P}_t$ with $|\mathcal{P}_t| \le r$ after parsing and duplicate removal; this set may be empty. The pool is $\mathcal{X}_t = \mathcal{X}_{t-1} \cup \mathcal{P}_t$, the selector chooses $x_t \in \mathcal{X}_t$, and the observation is $f(x_t)$. Repeated evaluations are allowed in the mathematical model. They do not change the incumbent in this noise-free setting.

The pre-generation history $\mathcal{F}_t$ is generated by η, $\mathcal{D}_1$, and $(\mathcal{P}_s, x_s, f(x_s))_{s<t}$. It includes any additional observed algorithm state. Contexts can use any part of this observed history; restricting them to functions of the dataset is a special case. Dependence within batches and between successive batches is allowed. For the general theorem, the improvement condition is imposed conditional on both $\mathcal{F}_t$ and f; the further nonanticipation assumptions needed for the GP specialization are stated in Section 10.6.

Reachable benchmark. Fix f and the initial state. Start with their initial history. At any reachable finite history, extend it by a batch having positive probability under the generator's prescribed conditional law at that history, any selection from the updated pool, and its exact objective value. Repeat this construction over all finite histories, allowing every admissible selection rule. Let $\mathcal{X}_{\text{reach}}$ be the union of the initial inputs and the candidates appearing in these histories, and set

$$f^\dagger = \sup_{x \in \mathcal{X}_{\text{reach}}} f(x).$$

The generator's laws are specified on these histories, including ones not visited by the particular selector under analysis. This construction uses histories consistent with the same objective and keeps track of the accumulated pool. It excludes arbitrary contexts that could not arise from generation and evaluation. Since the domain and batch space are countable, an actual run stays within this reachable set almost surely. The benchmark may depend on f and the initial state, but does not depend on which admissible selector is analyzed. No maximizing point need exist.

Integrability and notation. Write $B = \sup_{x \in \mathcal{X}} |f(x)|$. The main paper assumes $\mathbb{E}[B] < \infty$. Consequently $f^\dagger$ is finite almost surely and $0 \le r_t \le r_1 \le 2B$ is integrable. The general proof only needs a finite reachable benchmark and $\mathbb{E}[r_1] < \infty$; the assumption in the main text is a convenient sufficient condition. In this appendix only, write

$$v_t := \max_{x \in \mathcal{X}_t} f(x), \qquad g_t := f^\dagger - v_t, \qquad d_t := v_t - f(x_t).$$

Here $v_0 = m_1$, g_t is generator regret, and d_t is instantaneous pool regret. We retain $m_t = \max_{(x,f(x)) \in \mathcal{D}_t} f(x)$, $r_t = f^\dagger - m_t$, $\ell_t = v_t - m_{t+1}$, $L_T = \sum_{t=1}^T \ell_t$, and $R_T^{\mathrm{pool}} = \sum_{t=1}^T d_t$. The symbol r without a subscript remains the proposal budget.

10.2 Regret identities and the role of feedback

Pool persistence and inclusion of the initial data give, pathwise,

$$m_t \le m_{t+1} \le v_t \le f^\dagger. \tag{7}$$

The simple-regret decomposition and selection comparison follow:

$$r_{t+1} = g_t + \ell_t, \tag{8}$$

$$0 \le \ell_t \le r_{t+1}, \qquad d_t = \ell_t + \big(m_{t+1} - f(x_t)\big) \ge \ell_t. \tag{9}$$

Summation gives $L_T \le R_T^{\mathrm{pool}}$. These statements hold before taking expectations and require neither the improvement assumption nor a surrogate model.

The decomposition evaluates the best observation retained after a round. The regret of the individual evaluation instead obeys

$$f^\dagger - f(x_t) = g_t + d_t. \tag{10}$$

Thus simple regret and evaluation regret have the same generator component but different selection components. This distinction explains why sublinear cumulative pool regret is sufficient, but need not be necessary, for simple-regret convergence.

Neither identity makes the components independent. The pool depends on contexts formed from selected observations, and the selector acts on that pool. A selection decision can therefore change both future g_t and future ℓ_t. The two terms in Theorem 1 are also not separate bounds on g_T and ℓ_T: the theorem controls their sum through the full history of missed improvements.

10.3 Proof of the generation–selection bound

We give explicit, nonoptimized constants.

Theorem 2 (Explicit generation–selection bound) *Under the setup above, fix any admissible selector and suppose that for some $\alpha \in (0,1)$ and $p \in (0,1]$,*

$$\mathbb{P}(v_t \ge m_t + \alpha r_t \mid \mathcal{F}_t, f) \ge p$$

almost surely whenever $r_t > 0$. Then, for every integer $T \ge 1$,

$$\mathbb{E}[r_{T+1}] \le 3\mathbb{E}[r_1] e^{-\alpha p T/24} + \frac{8}{\alpha T}\mathbb{E}[L_T]. \tag{11}$$

The constants α, p are uniform over rounds and histories of this process. The result applies to every selector for which the assumption holds; a guarantee uniform over all admissible histories supplies the same constants for all selectors.

Proof strategy. We divide progress into geometric gap levels solely for the proof. At a given level, a discovery round waits for a useful candidate. Once it is stored, each further round either crosses the next level or incurs a definite amount of selection regret. Discovery rounds have geometric waiting-time bounds even with dependent proposals; the other delays are charged to L_T.

10.3.1 AN ADAPTIVE WAITING-TIME BOUND

Consider at most J successive episodes. During an episode, each trial has conditional success probability at least p given the full preceding history. The episode ends at its first success, or earlier. Episodes may be skipped and the process may stop before reaching all of them. If D is the total number of trials, then

$$\mathbb{P}(D > n) \le \mathbb{P}\left(\sum_{j=1}^{J} Z_j > n\right), \tag{12}$$

where Z_j are independent geometric random variables with success probability p, supported on $\{1, 2, \ldots\}$.

To see this, conditional on an episode's entry history, the probability that its first k trials all fail is at most $(1-p)^k$, by iterated conditioning. Its trial count is therefore stochastically dominated by a geometric variable; early exits and skipped episodes only decrease the count. The conditional domination is uniform over entry histories, so successive counts can be coupled, using independent uniform variables and conditional inverse distribution functions, below independent geometric variables. This proves equation 12; independence of the actual trials or episode lengths is unnecessary.

In particular, if $J \le pn/2$, the binomial representation of the sum of geometric waiting times and a Chernoff bound give

$$\mathbb{P}(D > n) \le \mathbb{P}(\mathrm{Bin}(n, p) < J) \le e^{-pn/8}. \tag{13}$$

The case $J = 0$ has $D = 0$. The same statements hold conditional on fixed objective and initial-state information when the conditional success guarantee holds under that conditioning.

10.3.2 LEVELS, DISCOVERY, AND SELECTION DELAYS

For the remainder of the proof, condition on f and the initial state, so that r_1 is fixed. If $r_1 = 0$, all later simple regrets are zero. Otherwise set

$$\beta = 1 - \alpha/2, \qquad u_j = \beta^j r_1, \qquad j = 0, 1, \ldots.$$

Round t is in phase j when $u_{j+1} < r_t \le u_j$. Monotonicity of r_t ensures that each phase is visited at most once; one evaluation can skip several phases. A candidate is useful for phase j if its value is at least $f^\dagger - (1-\alpha)u_j$.

Call a round a discovery round when no such candidate is present in $\mathcal{X}_{t-1}$. At any discovery round in phase j,

$$m_t + \alpha r_t = f^\dagger - (1-\alpha) r_t \ge f^\dagger - (1-\alpha) u_j.$$

The improvement assumption therefore gives conditional probability at least p of obtaining a useful candidate. Discovery within a phase stops at that success or when the phase ends. Once a useful candidate is present, persistence prevents any further discovery rounds in that phase. Accordingly, the total number of discovery rounds in the first J phases is dominated as in equation 12.

Consider instead a round with a useful candidate already stored before generation. If the round does not finish its phase, then $r_{t+1} > \beta u_j$, and

$$\begin{aligned} \ell_t = v_t - m_{t+1} &\ge r_{t+1} - (1-\alpha) u_j \\ &> (\beta - 1 + \alpha) u_j = \frac{\alpha}{2} u_j \ge \frac{\alpha}{2} r_{T+1}, \qquad t \le T. \end{aligned} \tag{14}$$

This is the charging step: delaying an available improvement costs selection regret without another generation success.

10.3.3 COMBINING THE TWO TYPES OF DELAY

For $T \ge 2$, choose

$$n = \lfloor T/2 \rfloor, \qquad J = \lfloor pn/2 \rfloor.$$

Let D count discovery rounds up to time T while the process is in one of the phases $0, \ldots, J-1$. By equation 13, conditional on f and the initial state, $\mathbb{P}(D > n) \le e^{-pn/8}$. If $r_{T+1} \le u_J$, the target level has already been reached. Otherwise every round through T belongs to one of those first

J phases. On $\{D \le n\}$, at most n rounds are discovery rounds, and at most J other rounds finish a phase. At least

$$T - n - J \ge T/4$$

rounds consequently satisfy equation 14. Summing their losses yields $L_T \ge \alpha T r_{T+1}/8$. These alternatives imply the pathwise inequality

$$r_{T+1} \le \beta^J r_1 + \frac{8L_T}{\alpha T} + r_1 \mathbf{1}\{D > n\}. \tag{15}$$

When $J = 0$, the first term alone bounds r_{T+1}, so the same inequality remains valid.

Because $n \ge T/3$, $\beta^{-1} \le 2$, and $\log \beta \le -\alpha/2$,

$$\beta^J \le 2e^{-\alpha p n/4} \le 2e^{-\alpha p T/12}, \qquad e^{-pn/8} \le e^{-pT/24} \le e^{-\alpha p T/24}.$$

Taking conditional expectations in equation 15 thus gives

$$\mathbb{E}[r_{T+1} \mid f, \mathcal{F}_1] \le 3r_1 e^{-\alpha p T/24} + \frac{8}{\alpha T}\mathbb{E}[L_T \mid f, \mathcal{F}_1].$$

Averaging over f and the initial state proves equation 11 for $T \ge 2$. Conditioning before averaging is important: it does not assume that r_1 and the discovery delays are independent. For $T = 1$, use $r_2 \le r_1$ and $3e^{-\alpha p/24} > 1$. This completes the proof of Theorem 2 and establishes Theorem 1 with $C_1 = 3$, $C_2 = 8$, and $c = 1/24$.

10.4 Convergence, stagnation, and cumulative regret

Necessary and sufficient selection condition. Under the improvement assumption,

$$\mathbb{E}[r_{T+1}] \longrightarrow 0 \quad \Longleftrightarrow \quad \mathbb{E}[L_T]/T \longrightarrow 0. \tag{16}$$

Sufficiency follows from Theorem 2. For necessity, equation 9 gives

$$0 \le \frac{\mathbb{E}[L_T]}{T} \le \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}[r_{t+1}],$$

and the right-hand side tends to zero by Cesàro averaging. Necessity does not require the improvement assumption. Since r_t is nonincreasing and bounded by the integrable r_1, it has an almost-sure limit r_∞, and dominated convergence gives $\mathbb{E}[r_t] \to \mathbb{E}[r_\infty]$. Thus the equivalent conditions in equation 16 also imply, and are implied by, $r_t \to 0$ almost surely.

Persistent stagnation. If $\mathbb{P}(r_\infty > 0) > 0$, then Theorem 2 implies

$$\liminf_{T\to\infty} \frac{\mathbb{E}[L_T]}{T} \ge \frac{\alpha}{8}\mathbb{E}[r_\infty] > 0.$$

This is a statement about expected cumulative missed improvement; it does not assert a pathwise lower bound on every individual round. The generator can temporarily stop producing better candidates while an adequate opportunity remains stored. The improvement assumption then holds with probability one, but leaving that opportunity unused incurs selection regret. If no adequate candidate is stored, the conditional probability lower bound prevents an indefinitely long sequence of failed discovery trials with positive probability at any fixed gap level.

Ideal selection. If $\ell_t = 0$ almost surely at every round, a sharper bound is

$$\mathbb{E}[r_{T+1}] \le (1 - \alpha p)^T \mathbb{E}[r_1]. \tag{17}$$

Indeed, a successful improvement event gives $r_{t+1} \le (1 - \alpha) r_t$, and otherwise $r_{t+1} \le r_t$. Conditional expectation and iteration prove the claim. The more conservative constants in Theorem 2 accommodate arbitrary selectors.

Why a one-step argument is weaker. Let $I_t = \mathbf{1}\{v_t \geq m_t + \alpha r_t\}$. The identity $r_{t+1} = g_t + \ell_t$ implies

$$r_{t+1} \leq r_t - \alpha r_t I_t + \ell_t.$$

On taking expectations, summing, and using monotonicity,

$$\alpha p \sum_{t=1}^{T} \mathbb{E}[r_t] \leq \mathbb{E}[r_1] - \mathbb{E}[r_{T+1}] + \mathbb{E}[L_T], \qquad \mathbb{E}[r_{T+1}] \leq \frac{\mathbb{E}[r_1] + \mathbb{E}[L_T]}{\alpha p T}. \tag{18}$$

This valid bound pays $1/p$ on the selection term and has only an inverse-time initial-gap term. The persistence argument gives the stronger bound used in the main paper.

What persistence adds. Persistence strengthens the rate rather than being necessary for the convergence implication itself. The one-step bound in equation 18 also applies to temporary proposal pools that retain the incumbent. For example, let the incumbent have value 0, and let a value-1 point be proposed with probability p each round. A selector that always ignores it has simple regret 1. If each proposal is discarded, its expected cumulative missed improvement is only pT; if proposals are retained, its average expected missed improvement tends to 1. This illustrates why a bound without a $1/p$ selection penalty needs an additional retention property. The proof uses retention of useful opportunities until they are realized or superseded; keeping every candidate is a sufficient way to ensure this property.

Cumulative evaluation regret. Although the main result concerns the best evaluated point, define $R_T^{\mathrm{eval}} := \sum_{t=1}^{T}(f^\dagger - f(x_t))$. Equations equation 8 and equation 10 give

$$R_T^{\mathrm{eval}} = \sum_{t=1}^{T} r_{t+1} + R_T^{\mathrm{pool}} - L_T.$$

Using equation 18, $r_{t+1} \leq r_t$, and $L_T \leq R_T^{\mathrm{pool}}$, we obtain

$$\mathbb{E}[R_T^{\mathrm{eval}}] \leq \frac{\mathbb{E}[r_1] + \mathbb{E}[R_T^{\mathrm{pool}}]}{\alpha p}. \tag{19}$$

Thus a sublinear Bayesian pool-regret bound also gives sublinear Bayesian cumulative evaluation regret relative to $f^\dagger$. This is a distinct guarantee from the simple-regret bound and can have different dependence on p.

10.5 Assumptions, examples, and reachable performance

Feedback can couple both components. Consider three points a, b, c with values $0, 1, 2$, initially observing only a. The generator proposes b until b is evaluated, and proposes c thereafter. The reachable benchmark is $f^\dagger = 2$. The improvement assumption holds with $\alpha = 1/2$ and $p = 1$ at every reachable history with positive gap. Selecting b and then c reaches the benchmark in two rounds. A selector that repeatedly evaluates a instead has $g_t = 1$, $\ell_t = 1$, and $r_{t+1} = 2$ at every round. The failure to select b both leaves an improvement unused and prevents the feedback that would reveal c. The generator can therefore stop improving while the improvement assumption continues to hold.

Linear pool regret is inconclusive. The preceding selector has linear pool regret and fails to converge. Conversely, suppose the initial dataset already contains points of values 1 and 0, the reachable benchmark is 1, and the selector repeatedly evaluates the latter. Then $r_t = \ell_t = 0$ but $R_T^{\mathrm{pool}} = T$. The same behavior can follow a finite prefix that first discovers and evaluates the optimum. The same phenomenon can occur with distinct new evaluations if the generator supplies a fresh zero-valued point at each round. Consequently even an actual linear growth of pool regret, rather than merely an $O(T)$ upper bound, does not determine whether simple regret converges.

The improvement assumption excludes suboptimal traps. Reachability alone does not ensure convergence. Modify the three-point example so that the first proposal is b or c with equal probability, and thereafter the generator repeats that same point forever. An ideal selector has $L_T = 0$, and

c is reachable, but on the branch that proposes only b the simple regret remains 1. The improvement assumption fails on that branch. More generally, the assumption requires opportunities that close a fixed fraction of the gap; it does not follow merely from better prompting, a large proposal budget, or eventual accessibility under some other history.

Batches and context dependence. The assumption concerns the conditional distribution of the entire retained batch and the resulting pool. Neither its points nor successive batches must be independent. The actual probability of an improvement can change with the context; p is a uniform lower bound. If proposals happened to be conditionally independent and each hit a suitable improvement set with probability at least u, the batch success probability would be at least $1 - (1 - u)^r$. This is a special case, not an assumption of the analysis. Without such structure, increasing r need not improve p.

Accuracy-dependent improvement opportunities. The uniform probability in the main assumption can be relaxed while retaining convergence. Keep a fixed $\alpha \in (0, 1)$ and suppose that, for every target accuracy $\epsilon > 0$, there is a deterministic $p_\epsilon \in (0, 1]$ such that, at every round with $r_t > \epsilon$,

$$\mathbb{P}\left(\max_{x \in \mathcal{X}_t} f(x) \geq m_t + \alpha r_t \mid \mathcal{F}_t, f\right) \geq p_\epsilon \tag{20}$$

almost surely. Each p_ϵ is uniform over rounds and histories above that accuracy, but no positive lower bound over all accuracies is required. In particular, p_ϵ may tend to zero as $\epsilon \downarrow 0$. Under the remaining assumptions of the main theorem, every selector satisfies, for all $T \geq 1$ and $\epsilon > 0$,

$$\mathbb{E}[r_{T+1}] \leq \epsilon + 3\mathbb{E}[r_1]e^{-\alpha p_\epsilon T/24} + \frac{8}{\alpha T}\mathbb{E}[L_T]. \tag{21}$$

For a bound at a single accuracy, the probability condition is needed only at that accuracy.

Proof of the accuracy-dependent bound. Fix $\epsilon > 0$ and, for analysis only, cap objective values at $a = f^\dagger - \epsilon$: $\widetilde{f}(x) = \min\{f(x), a\}$. Keep the original histories, generated pools, selections, and observed feedback. The capped benchmark is a, the capped incumbent is $\widetilde{m}_t = \min\{m_t, a\}$, and the remaining gap is

$$\widetilde{r}_t = a - \widetilde{m}_t = (r_t - \epsilon)_+.$$

When this gap is positive, an opportunity in equation 20 also satisfies

$$\max_{x \in \mathcal{X}_t} \widetilde{f}(x) \geq \widetilde{m}_t + \alpha \widetilde{r}_t.$$

Moreover, writing $v_t = \max_{x \in \mathcal{X}_t} f(x)$, the capped selection loss obeys

$$0 \leq \widetilde{\ell}_t = \min\{v_t, a\} - \min\{m_{t+1}, a\} \leq v_t - m_{t+1} = \ell_t.$$

The main theorem's proof therefore applies to these capped quantities with probability p_ϵ, conditioning on the original $\mathcal{F}_t$ and f. It uses the conditional improvement guarantee, persistence, and the incumbent update; it does not require the actual algorithm to observe the capped values. Since $\widetilde{r}_1 \leq r_1$ and $r_{T+1} \leq \epsilon + \widetilde{r}_{T+1}$, the explicit constants 3, 8, and $1/24$ from the main proof give equation 21.

Consequence for convergence. If $\mathbb{E}[L_T] = o(T)$, fix $\epsilon > 0$ and let $T \to \infty$ in equation 21 to obtain $\limsup_T \mathbb{E}[r_{T+1}] \leq \epsilon$. Then let $\epsilon \downarrow 0$. The converse still follows from $\ell_t \leq r_{t+1}$ and Cesàro averaging. Thus expected simple regret converges to zero if and only if $\mathbb{E}[L_T]/T \to 0$ under this weaker assumption. The time needed for a given accuracy depends on p_ϵ, so there need not be a uniform exponential convergence rate. The phases in the main proof remain proof devices; the relaxed assumption and its bound are stated directly in terms of target accuracy.

When the global optimum is inaccessible. Let $f^\star = \sup_{x \in \mathcal{X}} f(x)$. The comparison with the unrestricted domain is exactly

$$\mathbb{E}[f^\star - m_{T+1}] = \mathbb{E}[f^\star - f^\dagger] + \mathbb{E}[r_{T+1}]. \tag{22}$$

The first term is an accessibility gap for the specified generator, task information, and initial state, with all admissible selectors allowed in the reachability definition. The theorem controls the second

term. In particular, it remains meaningful when the global maximizer cannot be generated, and when the reachable supremum is not attained. For cumulative evaluation regret, the corresponding global comparison adds $T\mathbb{E}[f^\star - f^\dagger]$ to $\mathbb{E}[R_T^{\mathrm{eval}}]$.

Exclusion from the support at one context is not enough to establish an accessibility gap: later evaluated feedback may make the same point reachable. Conversely, a point that could be generated under an inconsistent or unreachable context does not belong to the benchmark used here. This distinguishes structural inaccessibility from slow discovery within the reachable set.

10.6 GP Thompson sampling on adaptive pools

This subsection proves the specialization in equation 6. The pool-regret bound itself does not require the generator improvement assumption; that assumption is used only when combining it with Theorem 2.

10.6.1 Posterior correctness and the information structure

Conditional on fixed task information η, assume $f \sim \mathcal{GP}(\mu, k)$, where the mean and kernel are known, fixed, and $k(x, x) \leq 1$. The embedding, if used to define k, is also fixed. The prior is correctly specified. Initial evaluations and all later algorithmic decisions are nonanticipating: they use observed history and algorithmic randomness, without additional information about unobserved values of f. In particular,

$$\mathcal{P}_t \perp\!\!\!\perp f \mid \mathcal{F}_t. \tag{23}$$

Initial data collection must obey the analogous condition; an initial design selected using unrecorded objective values would need to be included in the observation model.

Let $\mathcal{F}_t^+ = \sigma(\mathcal{F}_t, \mathcal{P}_t)$ be the history after generation and before selection. The pool is fixed under this conditioning, and the posterior law of f is the exact GP posterior obtained from the observed evaluations. In particular, revealing the proposals does not further change that law. Adaptive observation locations cause no extra likelihood factor involving unobserved values: their conditional selection probabilities depend on the recorded history. Denote the posterior mean and standard deviation by $\mu_t(x)$ and $\sigma_t(x)$.

Thompson sampling draws $\widetilde{f}_t$ independently of f, conditional on $\mathcal{F}_t^+$, from this posterior and takes $x_t \in \arg\max_{x \in \mathcal{X}_t} \widetilde{f}_t(x)$. One joint Gaussian draw on the finite pool suffices. It is not necessary to sample the function on the whole domain. Fix the same deterministic tie-breaking order for this maximization and for $x_t^* \in \arg\max_{x \in \mathcal{X}_t} f(x)$.

10.6.2 Posterior matching on the realized pool

Given $\mathcal{F}_t^+$, the maximizers x_t and x_t^* have the same distribution, while x_t is independent of f. For any $\mathcal{F}_t^+$-measurable function U_t on the pool, adding and subtracting its values gives

$$\begin{aligned}\mathbb{E}[d_t \mid \mathcal{F}_t^+] = \mathbb{E}[f(x_t^*) - U_t(x_t^*) \mid \mathcal{F}_t^+] \\ + \mathbb{E}[U_t(x_t) - \mu_t(x_t) \mid \mathcal{F}_t^+].\end{aligned} \tag{24}$$

The cancellation uses equality of the conditional laws of the two maximizers, not a common maximizer across rounds. This is the posterior-sampling regret decomposition of Russo & Van Roy (2014), applied conditionally to the current pool. We give the remaining steps to handle the finite adaptive pool and noiseless observations explicitly.

Let $M_t = n_0 + rt$, where $n_0 = |\mathcal{X}_0|$ is fixed or a deterministic upper bound on the initial pool size, and set

$$a_t = \sqrt{2\log(2M_t t^2)}, \qquad U_t(x) = \mu_t(x) + a_t\sigma_t(x), \qquad s_t = \max_{x \in \mathcal{X}_t} \sqrt{k(x, x)} \leq 1.$$

For a standard normal Z and $a \geq 0$, $\mathbb{E}[(Z-a)_+] \leq e^{-a^2/2}/\sqrt{2\pi}$. Here $(z)_+ = \max\{z, 0\}$ is used only for this Gaussian tail bound. Summing marginal tail expectations on the pool,

$$\begin{aligned}\mathbb{E}[f(x_t^*) - U_t(x_t^*) \mid \mathcal{F}_t^+] &\leq \sum_{x \in \mathcal{X}_t} \mathbb{E}[(f(x) - U_t(x))_+ \mid \mathcal{F}_t^+] \\ &\leq \frac{M_t s_t}{\sqrt{2\pi}} e^{-a_t^2/2} = \frac{s_t}{2\sqrt{2\pi}\, t^2}. \end{aligned} \tag{25}$$

No independence among candidate values is used. Combining with equation 24, taking expectations, and summing yields

$$\mathbb{E}[R_T^{\text{pool}}] \leq a_T\, \mathbb{E}\left[\sum_{t=1}^{T} \sigma_t(x_t)\right] + \frac{1}{2\sqrt{2\pi}} \sum_{t=1}^{T} \frac{\mathbb{E}[s_t]}{t^2}. \tag{26}$$

10.6.3 A VARIANCE BOUND FOR NOISELESS OBSERVATIONS

For a nonempty domain $A \subseteq \mathcal{X}$ and a sequence $z_1, \ldots, z_T \in A$, let $K_T = [k(z_i, z_j)]_{i,j=1}^T$ and define

$$\gamma_T(A) = \frac{1}{2} \sup_{z_1, \ldots, z_T \in A} \log\det(I + K_T).$$

Repeated locations are allowed in the supremum. This quantity is finite under $k(x, x) \leq 1$, since $\log\det(I + K_T) \leq \operatorname{tr}(K_T) \leq T$. Also $A \subseteq A'$ implies $\gamma_T(A) \leq \gamma_T(A')$. Sublinear growth is an additional property of the kernel and domain, not a consequence of variance normalization.

For each realized sequence $x_1, \ldots, x_T$, let q_t be the posterior variance at x_t in an auxiliary GP regression problem with the same prior kernel and unit-variance independent observation noise at $x_1, \ldots, x_{t-1}$, without using the initial observations. The actual noiseless posterior uses at least as much information, so

$$0 \leq \sigma_t(x_t)^2 \leq q_t \leq 1. \tag{27}$$

This is a matrix comparison for fixed locations. Exact conditioning, with a pseudoinverse if needed, cannot have larger conditional variance than noisy conditioning, and additional initial observations can only reduce it. The comparison therefore holds pathwise for adaptive locations and also for repeated evaluations.

The determinant identity for sequential GP conditioning gives

$$\sum_{t=1}^{T} \log(1 + q_t) = \log\det(I + K_{x_{1:T}}) \leq 2\gamma_T(\mathcal{X}_T).$$

The last inequality is pathwise: every selected location belongs to the final accumulated pool. It does not condition the posterior on that future pool or require it to be independent of the objective. Furthermore, $\mathcal{X}_T \subseteq \mathcal{X}_{\text{reach}}$ almost surely, so $\gamma_T(\mathcal{X}_T) \leq \gamma_T(\mathcal{X}_{\text{reach}})$. For $0 \leq q \leq 1$, concavity of $\log(1+q)$ gives $\log(1+q) \geq q \log 2$. Hence

$$\sum_{t=1}^{T} \sigma_t(x_t)^2 \leq \frac{2\gamma_T(\mathcal{X}_T)}{\log 2}, \qquad \sum_{t=1}^{T} \sigma_t(x_t) \leq \sqrt{\frac{2T\gamma_T(\mathcal{X}_T)}{\log 2}}. \tag{28}$$

The second inequality is Cauchy–Schwarz. The auxiliary noise is only a proof device; the algorithm still observes and conditions on exact values. In particular, we never identify $\gamma_T(A)$ with mutual information from noiseless observations, which need not be finite.

10.6.4 EXPLICIT GP–TS BOUNDS

Since the pools are nested, $s_t \leq s_T$ for $t \leq T$. Substituting equation 28 into equation 26 and summing t^{-2} proves

$$\begin{aligned}\mathbb{E}[R_T^{\text{pool}}] \leq \sqrt{\frac{4T\log(2M_T T^2)}{\log 2}}\, \mathbb{E}\Big[\sqrt{\gamma_T(\mathcal{X}_T)}\Big] \\ + \frac{\pi^2}{12\sqrt{2\pi}} \mathbb{E}[s_T].\end{aligned} \tag{29}$$

The additive term can be absorbed into the leading term. Indeed, pathwise, $\gamma_T(\mathcal{X}_T) \geq \frac{1}{2}\log(1+s_T^2) \geq \frac{1}{2}(\log 2)s_T^2$, using a repeated location of largest prior variance in the finite pool. Consequently, $\mathbb{E}[s_T] \leq \sqrt{2/\log 2}\,\mathbb{E}[\sqrt{\gamma_T(\mathcal{X}_T)}]$. Also $\log(2M_T T^2) \leq 2\log(2M_T T)$ and $T\log(2M_T T) \geq \log 2$. These inequalities give, conservatively,

$$\mathbb{E}[R_T^{\mathrm{pool}}] \leq 5\sqrt{T\log\big(2(n_0+rT)T\big)}\,\mathbb{E}\Big[\sqrt{\gamma_T(\mathcal{X}_T)}\Big]. \tag{30}$$

All these inequalities also hold when the relevant kernel variances are zero. By monotonicity, the same bound holds with $\mathcal{X}_T$ replaced by $\mathcal{X}_{\mathrm{reach}}$, or by any fixed domain containing all candidates.

Combining equation 30, $L_T \leq R_T^{\mathrm{pool}}$, and Theorem 2 establishes

$$\begin{aligned}\mathbb{E}[r_{T+1}] \leq\ & 3\mathbb{E}[r_1]e^{-\alpha p T/24} \\ & + \frac{40}{\alpha}\sqrt{\frac{\log\big(2(n_0+rT)T\big)}{T}}\,\mathbb{E}\Big[\sqrt{\gamma_T(\mathcal{X}_{\mathrm{reach}})}\Big].\end{aligned} \tag{31}$$

This proves equation 6 with $C_3 = 40$. The numerical constants are not optimized.

10.6.5 Scope and interpretation of the GP specialization

Smaller domains and kernel complexity. The Gaussian tail calculation ranges over the realized pool, which explains the logarithmic dependence on $n_0 + rT$ rather than the cardinality of $\mathcal{X}$. The variance argument gives the additional hierarchy

$$\mathbb{E}\Big[\sqrt{\gamma_T(\mathcal{X}_T)}\Big] \leq \mathbb{E}\Big[\sqrt{\gamma_T(\mathcal{X}_{\mathrm{reach}})}\Big] \leq \sqrt{\gamma_T(\mathcal{X})}.$$

These inequalities quantify a potential statistical benefit of selecting within a smaller domain. They remain valid when the reachable set depends on the objective. The proof first conditions on the observed history and current pool to establish posterior matching, then applies pathwise variance inequalities and takes expectations. Conditioning the GP prior on the entire future pool or reachable set would generally change the posterior law and is not used.

For any fixed nonempty A containing all possible candidates, the expected complexity can instead be bounded by $\sqrt{\gamma_T(A)}$. Thus the familiar deterministic information-gain bound is recovered with A in place of the full domain. A smaller domain need not have strictly smaller information gain: the benefit depends on the kernel correlations and directions accessible within it. In particular, full generator support can make $\mathcal{X}_{\mathrm{reach}} = \mathcal{X}$, while the sharper realized-pool bound can still be smaller. Excluding useful regions can also lower $f^\dagger$, as discussed in equation 22.

A concrete convergence regime. If the kernel restricted to a fixed domain A containing all candidates has rank at most d, K_T has at most d nonzero eigenvalues and trace at most T on that domain. Concavity of the logarithm gives

$$\gamma_T(A) \leq \frac{d}{2}\log(1+T/d), \qquad d \geq 1.$$

For fixed α, p, r, n_0, the selection term in equation 31 is therefore $\widetilde{O}(\sqrt{d/T}/\alpha)$. More generally, $\mathbb{E}[\sqrt{\gamma_T(\mathcal{X}_{\mathrm{reach}})}]\sqrt{\log(2(n_0+rT)T)/T} \to 0$ suffices for convergence; the realized-pool quantity gives a weaker sufficient condition. For a fixed enclosing domain A, it is enough that $\gamma_T(A)\log(2(n_0+rT)T) = o(T)$. An arbitrary normalized kernel need not satisfy this condition.

Exact posterior sampling. Posterior matching requires the correct conditional law and an independent posterior draw. The proof does not directly cover refitted plug-in kernel parameters, approximate posterior samples, or approximate maximization. Such methods remain covered by the general generation–selection theorem whenever their missed improvements can be controlled, but require their own selection analysis.

Scope of the general result. The arguments use sequential evaluations, a nondecreasing incumbent formed from exact observations, and retention of all generated candidates. Noisy evaluations, evaluation batches larger than one, or pool pruning require changes to the analysis. The context rule itself can be arbitrary: the general theorem concerns its induced improvement probabilities, not a particular prompting or exemplar rule.

11 ADDITIONAL EXPERIMENTAL DETAILS AND RESULTS

This section provides additional details and supporting results for the experimental evaluation in Section 6. We include task descriptions and ablation studies that further analyze the behavior of GaS. Together, these results provide a more detailed and causal understanding of the performance trends observed in the main paper.

Semantle. Semantle Agarwal et al. (2025) is a word-search task in which the goal is to identify a hidden English word using a limited number of guesses. After each guess, the optimizer receives only a scalar similarity score: the cosine similarity between SimCSE Gao et al. (2021) embeddings of the guessed word and the target word. Each problem instance consists of a target word together with an initial warm-start set of 20 randomly sampled words and their scores. We use 10 target words, each with 5 warm-start sets of 20 words each, and a budget of 400 black-box evaluations per run. We report the *average best score*, i.e., the highest similarity found so far, averaged over the 50 runs; a score of 1 means that the target word has been found. For surrogate modeling, words are embedded with gte-Qwen2-1.5B-instruct Li et al. (2023). All methods evaluate $q = 10$ words per round; GAS, OPRO, and BOPRO use $k = 10$ in-context examples, and GAS generates $r = 30$ candidates per round.

Molecular optimization. For molecular optimization, each candidate is a SMILES string. The objective is a scalarized score combining drug-likeness and binding affinity. Drug-likeness is measured by QED, while binding affinity is measured using Vina docking scores computed with DockString García-Ortegón et al. (2021). Following Agarwal et al. (2025), a Vina score $v(x)$ is negated and normalized using the typical range $[-12, -3]$, so that larger values are better, and then combined with QED as

$$f(x) = 0.2 \cdot \mathrm{QED}(x) + 0.8 \cdot \mathrm{NNV}(x), \qquad \mathrm{NNV}(x) = \frac{-v(x) - 3}{9},$$

where $\mathrm{NNV}(x) \in [0, 1]$ denotes the negative normalized Vina score. Thus, higher objective values correspond to molecules with better combined drug-likeness and binding affinity. We use 30 protein targets and a budget of 200 black-box evaluations. The initial warm-start set contains 3 molecules (acetamide, pentane, and benzene), shared by all targets and methods. We report the average best score over the 30 targets. For surrogate modeling, molecules are embedded with MolFormer Ross et al. (2022). All methods evaluate $q = 5$ molecules per round; GAS, OPRO, and BOPRO use $k = 5$ in-context examples, and GAS generates $r = 20$ candidates per round.

Symbolic regression. Each candidate is a symbolic expression in the variable t. Expressions are validated via their abstract syntax tree (AST), allowing addition, subtraction, multiplication, division, unary signs, numeric constants, t, and the functions `sin`, `cos`, and `exp`, up to a depth of 10 and a length of 128 characters. Expressions that fail validation, or whose outputs are non-finite or exceed 10^6 in magnitude, are invalid and still consume an evaluation. Valid candidates are scored by negative mean squared error (MSE) against the target outputs, with optimal value 0. We use 30 target expressions, each observed at 51 equally spaced points on $t \in [-2, 2]$ and a budget of 200 black-box evaluations. Because MSE varies by orders of magnitude across targets, we report the normalized MSE, $\min\{\mathrm{MSE}/\mathrm{Var}(y), 1\}$, averaged over targets, where $\mathrm{Var}(y)$ is the variance of the target outputs. For surrogate modeling, expressions are embedded by evaluating them on a fixed 64-point grid over $[-5, 5]$. Each run is initialized with a curated warm-start library of 20 expressions containing polynomial, trigonometric, and exponential terms. All methods evaluate $q = 10$ expressions per round; GAS, OPRO, and BOPRO use $k = 10$ in-context examples, and GAS generates $r = 40$ candidates per round.

Ackley-3D. We evaluate on the three-dimensional Ackley function Ackley (1987) with the standard parameters ($a = 20$, $b = 0.2$, $c = 2\pi$), discretized to the integer grid $\mathcal{S} = \{0, \ldots, 20\}^3$ (9,261 candidates) via $x = -0.5 + 0.1\, s$ for $s \in \mathcal{S}$. The global optimum at $x = 0$ thus lies at $s = (5, 5, 5)$, away from the grid centre. Each evaluation returns the Ackley loss with Gaussian observation noise ($\sigma = 0.01$). Each run starts from a warm-start set of 10 points that the LLM proposes as a space-filling design, without access to the objective, and has a budget of 50 evaluations. We use 30 seeds, with warm starts shared across methods. We report the *simple regret*, i.e., the lowest noise-free loss

Table 3: Mean LLM token usage per run. Values report mean ± sample standard error in millions of prompt-plus-completion tokens, as reported by the API.

Method	Semantle	MolOpt	SymReg	Ackley-3D
GAS (qTS)	0.0819 ± 0.0085	0.4147 ± 0.0250	0.2789 ± 0.0266	0.0058 ± 0.0016
GAS (qNEI)	0.0574 ± 0.0080	0.5729 ± 0.0266	0.2860 ± 0.0256	0.0044 ± 0.0008
OPRO	0.0632 ± 0.0051	0.0793 ± 0.0026	0.0874 ± 0.0074	0.0211 ± 0.0027
BOPRO	0.0535 ± 0.0047	0.0646 ± 0.0023	0.1247 ± 0.0107	0.0329 ± 0.0012
LLAMBO	30.8721 ± 4.9908	61.3600 ± 4.2365	29.8571 ± 2.0611	1.0207 ± 0.2478
LLM-ES	0.0132 ± 0.0008	0.0175 ± 0.0006	0.0433 ± 0.0022	0.0107 ± 0.0005

Table 4: Mean wall-clock time per run. Values report mean ± sample standard error in minutes.

Method	Semantle	MolOpt	SymReg	Ackley-3D
GAS (qTS)	2.75 ± 0.31	52.51 ± 2.96	3.18 ± 0.43	0.47 ± 0.12
GAS (qNEI)	2.06 ± 0.31	54.83 ± 2.26	3.30 ± 0.42	0.37 ± 0.06
OPRO	2.52 ± 0.21	31.18 ± 1.36	1.22 ± 0.09	1.29 ± 0.16
BOPRO	3.77 ± 0.45	30.71 ± 1.17	2.55 ± 0.21	2.33 ± 0.09
LLAMBO	11.92 ± 1.56	88.45 ± 20.59	11.72 ± 0.83	1.81 ± 0.38
LLM-ES	0.67 ± 0.04	18.24 ± 2.31	0.65 ± 0.03	0.59 ± 0.03
Standard BO	0.11 ± 0.01	25.15 ± 0.39	8.02 ± 0.33	0.04 ± 0.00

Times include LLM API latency, surrogate fitting and objective evaluation (docking dominates on MolOpt); runs that find the optimum stop early. On MolOpt and SymReg, Standard BO includes generating its fixed pool (500 molecules and 800 expressions).

among the evaluated points, averaged over seeds. For surrogate modeling, points are represented by their grid coordinates, normalized to the unit cube. All methods evaluate $q = 1$ point per round; GAS, OPRO, and BOPRO use $k = 10$ in-context examples, and GAS generates $r = 10$ candidates per round.

11.1 Implementation and compute environment

We implement all methods in `PyTorch` (2.9), using `BoTorch` (0.16) for Gaussian process modeling and `Sentence-Transformers` (5.2) for embedding models; MolOpt docking scores are computed with `DockString` (AutoDock Vina) and `RDKit`. LLM inference uses the Qwen models through the `Alibaba Cloud Model Studio (DashScope)` API, with thinking mode disabled. Experiments are run on `Ubuntu 22.04.5 LTS` with 28 vCPUs, 177 GiB RAM, and a single `NVIDIA H100 PCIe` (80 GB) GPU.

11.2 Token usage and costs

Table 3 reports the LLM token budget each method consumes per run. On Semantle, GaS uses about as many tokens as OPRO and BOPRO, and on Ackley-3D it is the cheapest method overall, since it typically reaches the optimum within a few iterations, after which no further LLM calls are made. On MolOpt and SymReg, GaS uses roughly 2–9× more tokens than OPRO and BOPRO, because each iteration generates a larger candidate pool for the acquisition function to choose from; it still stays below 0.6M tokens per run. LLAMBO is the clear outlier: because it uses the LLM as a surrogate model as well as a candidate generator, it consumes over a hundred times as many tokens as GAS on every task. LLM-ES is the cheapest method on Semantle, MolOpt and SymReg, since each call only recombines a few parent candidates. Relative to OPRO and BOPRO, the additional cost is a small constant factor, far from LLAMBO's. The improvements of GAS are therefore not bought with a disproportionate generation budget (see Table 2).

Table 4 reports wall-clock time per run. On Semantle, GaS takes 2.1–2.8 minutes, on par with OPRO (2.5) and BOPRO (3.8) and 4–6× faster than LLAMBO (11.9). On Ackley-3D, GaS is the fastest LLM-based method (0.4–0.5 minutes), 3–6× faster than OPRO and BOPRO, because it typically

finds the optimum within a few iterations and stops. On SymReg, GaS takes 3.2–3.3 minutes, about 1.3× BOPRO and 2.7× OPRO, since each iteration generates and embeds a larger candidate pool; it is still 2.4× faster than Standard BO (8.0, including pool generation) and 3.6× faster than LLAMBO (11.7). On MolOpt, where docking dominates, GaS takes 52.5–54.8 minutes, about 1.7–1.8× OPRO and BOPRO (31.2 and 30.7) but roughly 40% less than LLAMBO (88.5). LLM-ES is the fastest LLM-based method on the other three tasks.

11.2.1 PROMPTS

11.2.2 WARM-START GENERATION

Semantle

```
Your task is to guess a hidden test word from the English dictionary.
Start by guessing {n_cands} diverse words to maximize your chance of
guessing the hidden word. Be creative.

Do not guess any of the following words: {list of words repeated by
the LLM in the previous iteration}.
```

Dockstring

```
Your task is to find the optimal drug molecule that has both a high
druglikeness (QED) as well as a strong binding affinity (vina) to the
target protein. While both properties are important, binding affinity
is twice as important as the druglikeness. Here is the target
protein: {target_protein}.
Do not propose any of the following molecules: {list of molecules
repeated by the LLM in the previous iteration}.
```

Symbolic Regression

```
Your task is to find a symbolic expression in terms of t that best
fits the observed input-output pairs. Use only valid expressions over
t with numbers, the operators +, -, *, /, and the functions sin, cos,
and exp.

Here are the observed input-output pairs:
t = {t_vals}
y = {y_vals}

Start by guessing {n_cands} diverse expressions to maximize your
chance of finding the best expression. Be creative.

Do not guess any of the following expressions: {list of expressions
repeated by the LLM in the previous iteration}.

(Note: give only a list of expressions in the provided JSON format,
e.g. {"response": ["expr1", "expr2", ...]})
```

Ackley

```
Choose exactly {n_cands} distinct, space-filling integer coordinate
tuples in S={0,...,20}^3. Spread them across boundaries and the
interior and avoid clustering. Seed {seed} distinguishes equally good
layouts. Do not give explanations, formulas, benchmark names, scores,
or comments.
```

11.2.3 SOLUTION REPRESENTATION

To compute embeddings for the generated solutions, we first map the raw solution text to a template to provide additional context for sentence-embedding models when needed:

- Semantle: `"{x}"`
- Ackley: `"Euclidean coordinate values"`
- Dockstring: `"Protein target:{target} Molecule Candidate:{x}"`
- Symbolic Regression: `"Outputs on a fixed grid"`

11.2.4 SOLUTION GENERATION

Semantle

```
Your task is to guess a hidden test word from the English dictionary.
Use the below series of your previous guesses (in increasing) order
of their similarity to the hidden word in terms of their *meaning*)
to make a new guess. Your new guess should not have been made before
and should score higher than your previous guesses. Analyze your
previous guesses to decide what word could reach a target score of
{target_score:.4f}. If you guess an invalid word (i.e., not in the
dictionary or a repeat guess), you will get no score, so stick to
proper, single-word English words, and do not repeat your previous
guesses! Now, guess exactly n=%s new word(s) that could give you a
target score of {target_score:.4f}. (Note: give only the word in the
provided JSON format)
```

Dockstring

```
Your task is to find the optimal drug molecule that has both a high
druglikeness (QED) as well as a strong binding affinity (vina) to the
target protein. While both properties are important, binding affinity
is twice as important as the druglikeness. Here is the target
protein: { target_protein}. Use the below series of your previous
guesses in increasing order of their scalarized scores (between 0 and
1) to make a new guess to maximize the score. Your new guess should
not have been made before. Analyze your previous guesses to decide
what molecule to guess next. If you propose an invalid molecule or
make a repeat guess, you will get no score, so stick to valid SMILES
strings, and do not repeat your previous guesses! Now, guess exactly
n=%s new molecule(s) that could be the optimal molecule.. Be
creative! Now, guess exactly n=%s new molecule(s) that could give you
a target score of {target_score:.4f}. (Note: give only a list of
SMILES strings in the provided JSON format, e.g. \
{{"response": ["SMILES1", "SMILES2", ...]}})
```

Symbolic Regression

```
Given the following observed input-output pairs, your task is to find
a symbolic expression in terms of t that best fits the data. Use only
valid expressions over t with numbers, the operators +, -, *, /, and
the functions sin, cos, and exp.

t = {t_vals}
y = {y_vals}

Use the below series of your previous guesses, in
{increasing/decreasing} order of their scores, to make a new guess.
Your new guess should not have been made before and should score
```

```
higher than your previous guesses. Analyze your previous guesses to
decide what expression could reach the target score of {target_score}.

If you propose an invalid expression or repeat a previous guess, you
will get no score, so stick to valid expressions over t only and do
not repeat your previous guesses.

In particular, DO NOT REPEAT the following expressions: {list of
expressions repeated by the LLM in the previous iteration}.

Here are your top previous guesses, from {worst/best} to {best/worst}:

Expression: {expression_1}
Score: {score_1}

Expression: {expression_2}
Score: {score_2}

...

Now, guess exactly n={n_cands} new expression(s) that could give you
a target score of {target_score}.

Note: give only a list of expressions in the provided JSON format,
e.g. {"response": ["expr1", "expr2", ...]}.
```

Ackley

```
You are solving a black-box optimisation problem over
integer-coordinate
candidates. The objective is hidden. Return JSON only with exactly
this shape:
{"response": ["s1,s2,s3"]}. Every response entry must be a canonical
comma-separated integer coordinate string with no brackets, spaces,
decimals,
explanation, or extra keys.
Search an unknown black-box objective over the discrete integer domain
S={0,...,20}^3.
Observed coordinates, ordered from least to most promising:
{}

Return exactly {n_cands} distinct unseen coordinate strings likely to
improve
the objective.
Unavailable coordinates repeated in the immediately previous response:
{list of coordinates repeated by the LLM in the previous response}.
```