AGENTEVOLVE: LLM-GUIDED MULTI-OBJECTIVE EVOLUTIONARY OPTIMIZATION FOR EXPENSIVE BLACK-BOX DESIGN PROBLEMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Many engineering workflows rest on an expensive multi-objective design problem: a designer sets a few named parameters, runs a slow tool to score each candidate, and trades off conflicting objectives such as the area, speed and power of a chip, within a budget of tens to hundreds of evaluations. Tuning chip-implementation flows, sizing analog circuits and searching neural architectures are such problems. Their parameters have names, units and a short description that tell an expert what good designs look like. Evolutionary and Bayesian optimizers ignore this information; language-model optimizers use it but fix the model's role in advance, and budget-matched studies often trace their improvements to the surrounding loop rather than the model. We propose AGENTEVOLVE, a drop-in replacement for evolutionary optimizers in which the model is one more source of candidate designs and earns its budget share by measurement: the initial population it writes, periodic proposals and a sampler program it authors are pooled with evolutionary offspring, and each source receives evaluations in proportion to the hypervolume its past candidates added to the front. We evaluate one configuration on nine benchmarks (OpenROAD, an open-source flow that turns a hardware design into a chip layout, on three chip technologies; the NAS-Bench-301 architecture-search benchmark; XGBoost hyperparameter tuning; a knapsack; two analog circuits) at validation seeds kept apart from development: AGENTEVOLVE needs 3.09× fewer evaluations than the evolutionary optimizer NSGA-II to reach a target hypervolume on NAS-Bench-301 and 2.90× fewer on a low-dropout regulator, ending with 28% and 50% more hypervolume, and outperforms AGENTEVOLVE without LLM on three benchmarks. On the knapsack a program the model writes from the item descriptions brings the median run within 10^{-4} of the exact front's hypervolume, using the cheapest model tier at under $0.33 per run.

1 INTRODUCTION

Multi-objective design problems, such as tuning a chip-implementation flow, sizing an analog circuit or searching a neural architecture, ask for designs that trade off conflicting objectives, and allow only tens to hundreds of evaluations because each evaluation runs a slow tool. Today these designs are found by evolutionary or Bayesian optimizers, which use only measured results, although each problem also comes with a description of its parameters that tells an expert where good designs lie. For example, OpenROAD (Ajayi et al., 2019) is an open-source RTL-to-GDS flow: a chain of synthesis, placement and routing tools that turns a hardware description into a chip layout. It targets open process technologies, each the cell library and design rules of one manufacturing process; we use Nangate45 (45 nm), ASAP7 (a predictive 7 nm kit) and SKY130HD (SkyWater's 130 nm process). One evaluation, a full flow run on a small design, takes 43–102 s, and tuning nine parameters against die area, achieved clock period, power and wirelength allows about a hundred evaluations. The flow's parameters have descriptive names, such as the core utilization and the clock period constraint, and a short description of the flow tells a physical designer which settings trade die area against routability and timing against power. The same holds for analog sizing and architecture search: an expert can use their descriptions, but a black-box optimizer ignores them.

Measurement-driven optimizers, such as the multi-objective evolutionary algorithm NSGA-II (Deb et al., 2002) and Bayesian optimization (Daulton et al., 2021) (OpenROAD's parameter tuner, AutoTuner, uses the Bayesian optimizer HEBO (Cowen-Rivers et al., 2022)), are robust and sample-

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.

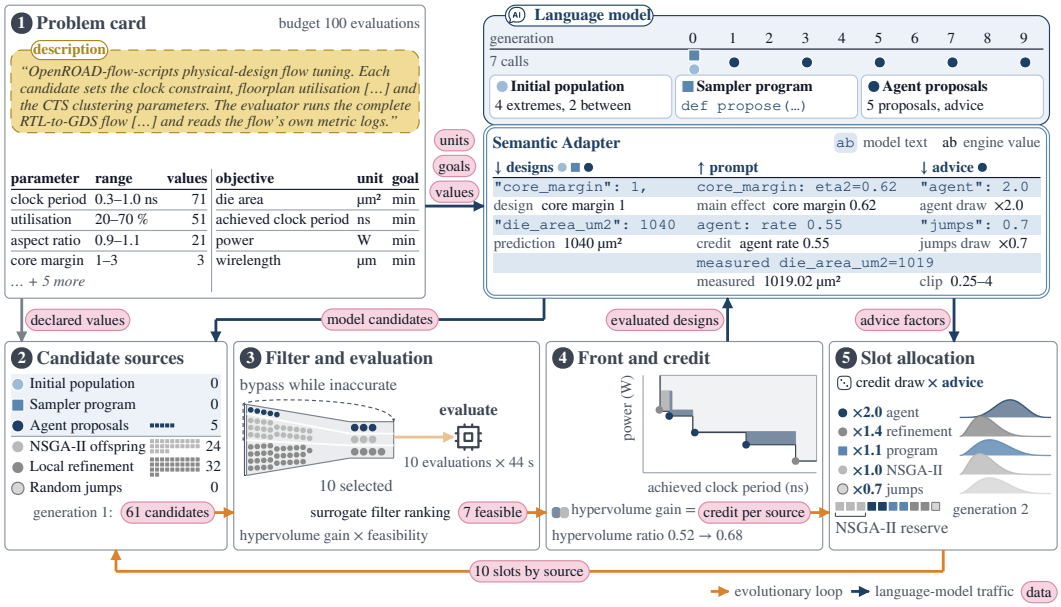


Figure 1: **AGENTEVOLVE in a single validation run on Nangate45** We show one generation of AGENTEVOLVE with the run's own values. The problem card (1) gives a description, the parameters with their values and the objectives with their goals. The language model, called 7 times in the run, writes the initial population (half of generation 0) and a sampler program and returns proposals and per-source advice every second generation; every value it reads or writes passes the Semantic Adapter, which we draw as the model's text above the engine's value. The candidates of all sources form one pool (2); the surrogate filter selects the ones the flow evaluates (3); each source's credit is the hypervolume gain of its candidates (4); and the next generation's slots go to the sources by credit draw times advice, above a reserve for NSGA-II offspring (5).

efficient, but treat the parameters as anonymous numbers and must spend evaluations to learn what the description already implies. Language-model optimizers can read such descriptions: they propose solutions from a history of scored attempts (Yang et al., 2023), write programs that are evolved against a scorer (Romera-Paredes et al., 2023; Novikov et al., 2025; Liu et al., 2024a), or act as priors inside Bayesian optimization (Liu et al., 2024c). But each fixes the model's role in advance, so its share of the evaluations does not depend on how its candidates perform. Most are also tested on a single objective with thousands of cheap evaluations, and budget-matched studies find that their improvements often come from the surrounding loop or a good default rather than the model (Zhang et al., 2024; Rodrigues et al., 2026). The closest prior work, LLM-aided multi-objective evolutionary algorithms (Liu et al., 2023; 2024d) and trust rules for LLM proposals in Bayesian optimization (Chang et al., 2025), decides when to call the model inside one step of the search; none arbitrates several kinds of model contribution against classical search by the hypervolume each one adds.

We present AGENTEVOLVE, which treats the language model as one more source of candidate designs rather than as the optimizer. The model reads a problem card (the description, the parameters with their units and values, and the objectives), writes the initial population, the model-written half of generation 0, proposes whole designs in a periodic agent turn, and authors a sampler program; these candidates are pooled with NSGA-II offspring, local refinement and random jumps, a surrogate filter pre-screens the candidates of every source, and each source receives evaluations in proportion to its credit, the hypervolume gain of its past candidates (the objective-space volume they added to the front, the best trade-offs found so far), allocated by discounted Thompson sampling, a bandit rule that favours recently productive sources. What the model knows enters the search where it pays, typically early, while measurements decide how much of the budget it keeps, so a wrong prior costs only the evaluations it loses. From the recorded model calls, we found that the engine's internal minimisation silently inverted the feedback the model received on maximised objectives. A Semantic Adapter, the single boundary for every value the model reads or writes, removes this class of error by construction. AGENTEVOLVE is a drop-in replacement for a classical optimizer: it takes the same problem object and evaluation budget and returns a front, so switching an NSGA-II script written with the pymoo library to it changes a few lines (Figure 3).

We evaluate one configuration on nine benchmarks in five domains: the chip flows (OpenROAD on the three technologies), the NAS-Bench-301 architecture-search benchmark (NB-301), two XG-

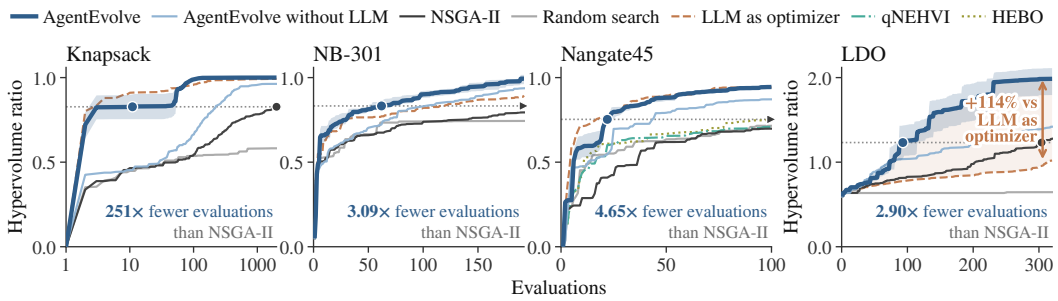


Figure 2: **Anytime hypervolume ratio** on four benchmarks: median over validation seeds of the hypervolume ratio (Section 4; the LDO is a low-dropout regulator) against evaluations, log scale on the knapsack; bands: interquartile range of AGENTEVOLVE. Dotted: the target; dots mark where the medians of AGENTEVOLVE (blue) and NSGA-II (grey) first reach it, an arrowhead that NSGA-II's does not within the budget. Each panel states the ETT ratio against NSGA-II (median of the paired per-seed ratios). On the LDO the shaded gap is to LLM as optimizer, the same model proposing designs without our engine. Only the Nangate45 panel shows the Bayesian specialists qNEHVI and HEBO, which we run on the chip flows (qNEHVI also on the op-amp).

Boost hyperparameter-tuning benchmarks, a two-objective 100-item knapsack, and two analog circuits, an operational amplifier (op-amp) and a low-dropout regulator (LDO). We read every claim at validation seeds kept apart from the development seeds used to design the engine, and call a comparison decisive when AGENTEVOLVE needs significantly fewer evaluations to reach a target hypervolume ratio, fixed before validation, and does not end behind. Against the measurement-driven tradition, AGENTEVOLVE is decisive over NSGA-II on 5 of 9 benchmarks, with 3.09× fewer evaluations to the target on NB-301 and 2.90× fewer on the LDO (Figure 2), and on Nangate45 it is decisive over HEBO. Against AGENTEVOLVE without LLM, which isolates the model's own contribution, it is decisive on NB-301, the knapsack and Nangate45. It does so with the cheapest model tier (GPT-5.6 luna), at under $0.33 per run, and on the knapsack a sampler program the model wrote from the item descriptions has a 92% budget share and brings the median run within 10^{-4} of the exact front's hypervolume. As the mechanism predicts, the model sources receive budget where the description lets the model infer the structure of good designs, and lose it elsewhere. On the op-amp, where only measurements reveal the thin feasible region, the Bayesian optimizer qNEHVI (Daulton et al., 2021) finds it and AGENTEVOLVE does not.

Our contributions are:

- **A generic LLM-guided multi-objective optimizer**, a drop-in replacement for classical ones, in which the model's candidates earn their budget share by hypervolume gain, with one configuration and no rule that refers to a specific benchmark.
- **A validation protocol for LLM-guided optimizers**: development and validation seeds kept apart, targets and verdict rules sealed before validation, and baselines that separate the model's contribution from the engine's, read at matched evaluation budgets and matched wall-clock time.
- **Validation results**: decisive over NSGA-II on 5 of 9 benchmarks and over random search on 5, over AGENTEVOLVE without LLM on three, and over HEBO and qNEHVI on Nangate45.
- **The limits of language-model guidance**, located by its weaker results: a budget floor below which no method separates, feasible regions only measurements reveal, and a feedback defect only recorded calls expose.

2 RELATED WORK

Language models as the optimizer Optimization by prompting (OPRO) feeds a model the history of scored solutions and asks for better ones (Yang et al., 2023), and population-based and agentic variants extend this to multi-objective functions and hyperparameter search (Brahmachary et al., 2024; Liu et al., 2024b). We run this approach as a baseline, LLM as optimizer: the same model as ours, proposing designs without our engine. AGENTEVOLVE instead gives the model's candidates a budget share that measurements decide.

Language models inside evolutionary and program search Evolutionary methods use language models as variation operators, the steps that mutate or recombine candidates (Lehman et al., 2022; Meyerson et al., 2023; Guo et al., 2024), or as authors of programs evolved against an automatic scorer (Romera-Paredes et al., 2023; Novikov et al., 2025; Lange et al., 2025; Liu et al., 2024a; Ye et al., 2024; van Stein & Bäck, 2024), assuming evaluations cheap enough to discard thousands of

pymoo NSGA-II

```
  problem = get_problem("zdt1", n_var=4)
- from pymoo.algorithms.moo.nsga2 import NSGA2
- from pymoo.optimize import minimize
- res = minimize(problem, NSGA2(pop_size=12),
-     ("n_gen", 5), seed=1)

- front = res.F
```

AGENTEVOLVE

```
  problem = get_problem("zdt1", n_var=4)
+ from agent_evolve import optimize
+ from agent_evolve.integrations.pymoo_adapter import from_pymoo
+ res = optimize(from_pymoo(problem), budget=60, seed=1,
+     model="openai/gpt-5.6-luna", effort="high", stack="v100",
+     v100_config=dict(engine="simple", agent=True, authored=True))
+ front = res.pareto_front
```

Figure 3: Integration surface: the same problem object of pymoo, a Python optimization library, under NSGA-II (left) and AGENTEVOLVE (right), which we condensed from the package's pymoo example. We use the right-hand call in every run; `stack` and `v100_config` select the engine of Algorithm 1.

model outputs against one scalar score. In AGENTEVOLVE a model output must earn each of a few hundred multi-objective evaluations, and the credit decides how many it receives.

Language models with measurement-driven optimizers Other systems pair the model with Bayesian optimization (Liu et al., 2024c; Chang et al., 2025) or multi-objective evolutionary algorithms (Liu et al., 2023; 2024d), or apply it to chip flows (Ghose et al., 2025), analog sizing (Yin et al., 2024; KarthikSomayajiN & Li, 2025) and hardware design-space exploration (Xiong et al., 2026); they fix the model's role, and budget-matched studies find such improvements often reduce to the loop or a good default (Zhang et al., 2024; Huang et al., 2024; Rodrigues et al., 2026). AGENTEVOLVE treats every model role as a source whose budget share is set by its credit, as in bandit-based adaptive operator selection, which picks search operators by their recent rewards (Costa et al., 2008; Li et al., 2014), and we report the model's contribution against AGENTEVOLVE without LLM.

3 METHOD

AGENTEVOLVE optimizes a black box $f : \mathcal{X} \to \mathbb{R}^m$ over a gridded space $\mathcal{X}$ within B evaluations. Like a classical optimizer, it takes a problem object and a budget and returns the non-dominated designs it evaluated, so replacing NSGA-II changes only the optimizer call, five lines replaced by six (Figure 3). The model reads a problem card (a short description, the objectives with directions, the parameters with their values); constraints arrive as a graded violation $v(x) \geq 0$. For the chip flows the card states that each candidate sets *"the clock constraint, floorplan utilisation, aspect ratio and margin, global/detail cell padding, the placement density lower bound addon, and the CTS [clock-tree synthesis] clustering parameters"* and that the evaluator runs *"the complete RTL-to-GDS flow"*, with grids such as `core_utilization` (20–70). It gives no per-parameter guidance, so whatever the model adds beyond the card comes from its own background knowledge. Each generation runs the stages of Figure 1 (Algorithm 1); we use one configuration for every benchmark.

Candidate sources Every candidate is tagged with its source. The classical sources are *NSGA-II offspring* (Deb et al., 2002), *local refinement* (one-parameter changes around designs on the front), *random jumps* (uniform, or a design on the front with 30–70% of its parameters redrawn) and, in generation 0, *space-filling draws*. The model sources are the *initial population*, the model-written half of generation 0 with one design at each objective's extreme and the rest between them; *agent proposals*, designs with predicted objectives from an agent turn every $I = \lceil G/8 \rceil$ of a run's G generations; and the *sampler program*, a Python generator written once, run in a sandbox and revised once if its credit rate falls below that of NSGA-II offspring over four generations. No model source is guaranteed a slot, so a source whose candidates perform poorly loses only its own budget share.

Surrogate filter An ensemble of classical surrogates with a violation head (μ_v, σ_v) is refit each generation. A source granted s slots draws $8s$ candidates; the filter keeps the s with the highest expected hypervolume improvement over the feasible front F times probability of feasibility,

$$\text{score}(x) = \text{EHVI}(x \mid F) \cdot \Phi\big((\tau - \mu_v(x))/\sigma_v(x)\big) \tag{1}$$

(τ: the violation tolerance), or the lowest $\mu_v(x)$ before any feasible design. It steps aside, passing pools of s unranked, while its cross-validated error does not beat the mean predictor.

Credit and allocation A source's hypervolume gain $G_{k,g}$ in generation g is the hypervolume its feasible designs $Y_{k,g}$ add to the front, $\text{HV}(F_{g-1} \cup Y_{k,g}) - \text{HV}(F_{g-1})$, or, before any feasible design, the violation they remove below the lowest evaluated, $\sum_x \max(0, v^\star - v(x))$. Over its $n_{k,g}$ evaluated candidates it earns the credit

$$c_{k,g} = \min\big(n_{k,g},\, G_{k,g}/b_g\big), \qquad \rho_{k,g} = c_{k,g}/n_{k,g} \in [0, 1], \tag{2}$$

Algorithm 1 AGENTEVOLVE run loop

Require: κ, B, N, λ ▷ card, budget, population, offspring
$G \leftarrow \lceil (B-N)/\lambda \rceil$; $I \leftarrow \max(1, \lceil G/8 \rceil)$; $\gamma \leftarrow 2^{-1/I}$; $P \leftarrow \textsc{WriteSampler}(\kappa)$
$D \leftarrow \textsc{Evaluate}\big(\textsc{InitialPopulation}(\kappa) \cup \textsc{SpaceFilling}()\big)$ ▷ generation 0
for $g = 1, \ldots, G$ **do**
  if $(g-1) \bmod I = 0$ **then** $(Q, \mathbf{a}) \leftarrow \textsc{AgentTurn}\big(\kappa, \textsc{Brief}(D)\big)$ ▷ proposals Q, advice $\mathbf{a}$
  $\hat{f} \leftarrow \textsc{RefitFilter}(D)$; $\mathbf{s} \leftarrow \textsc{Allocate}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{a}, \lambda)$ ▷ Eq. (4)
  $C \leftarrow \bigcup_k \textsc{Top}_{s_k}\big(\textsc{Draw}_k(8 s_k), \hat{f}\big)$ ▷ Eq. (1)
  $D \leftarrow D \cup \textsc{Evaluate}(C)$; $c_{k,g} \leftarrow \textsc{Credit}_k(D)$; $(\alpha_k, \beta_k) \leftarrow \textsc{Update}(c_{k,\le g}, \gamma)$ ▷ Eqs. (2), (3)
  $X \leftarrow \textsc{Survive}_N(D)$ ▷ feasible, fewest dominators, crowding
  if $\textsc{Behind}(P, \text{NSGA})$ **then** $P \leftarrow \textsc{Revise}(P)$ ▷ at most once
return $\textsc{NonDominated}(D)$

with b_g the best hypervolume gain per candidate of any source in g or, pooled, in the three generations before. A Beta posterior over each source's credit rate ρ, whose evidence decays with a half-life of one turn interval (Raj & Kalyani, 2017) so a source whose value changes can win slots back, drives Thompson sampling (Russo et al., 2017): NSGA-II first takes $\lceil n/4 \rceil$ of the n slots; every other slot goes to $k^\star$,

$$(\alpha_k, \beta_k) = (\alpha_0, \beta_0) + \textstyle\sum_{h<g} \gamma^{\,g-1-h}\, m_{k,h}\, (\rho_{k,h},\, 1-\rho_{k,h}), \tag{3}$$

$$k^\star = \arg\max_k \ \theta_k \operatorname{clip}\big(a_k, \tfrac{1}{4}, 4\big), \qquad \theta_k \sim \mathrm{Beta}(\alpha_k, \beta_k) \text{ with a new draw for each slot,} \tag{4}$$

where $m_{k,h} = \min(n_{k,h}, n_{\text{NSGA},h})$ pairs a source with NSGA-II in each generation, $(\alpha_0, \beta_0) = (2.96, 5.04)$, $\gamma = 2^{-1/I}$, and a_k is the agent's advice (1 when none). No rule refers to a problem.

Agent turn The agent reads the card, a brief of the evaluated designs (the front, main effects, source credits) and a table of its recent proposals, predicted against measured; it returns up to $\min(\lfloor \lambda/2 \rfloor, 6)$ proposals with predictions and an advice multiplier a_k per source. Its only feedback is measurement.

Semantic Adapter The engine minimises every objective internally. The Semantic Adapter is the single boundary through which values reach or leave the model: it renders briefs and prompts in each objective's declared direction and units, maps predictions and sampler outputs back, repairs replies onto the declared parameter shapes (for example, item indices on a space of booleans), and re-asks once, quoting the validator, when a reply cannot be repaired. We show why it exists in Section 6.

4 EXPERIMENTAL SETUP

Benchmarks We evaluate on nine benchmarks in five domains. The chip flows Nangate45, ASAP7 and SKY130HD run OpenROAD (Ajayi et al., 2019) on the small `gcd` design (a greatest-common-divisor circuit) in each technology (die area, achieved clock period, power, wirelength; 9 parameters; $B = 100$; 43–102 s per evaluation). NB-301 (Zela et al., 2020) is a surrogate model of neural architecture training (accuracy, runtime; 34 cell choices; $B = 192$). XGB-1489 and XGB-40981 tune XGBoost on OpenML datasets 1489 and 40981 through the YAHPO Gym surrogates (Pfisterer et al., 2021) (error and model-cost terms; 13 hyperparameters; $B = 192$). The two-objective 100-item knapsack has a known exact front ($B = 2000$). The op-amp and the LDO come from AnalogGym (Li et al., 2024) and are sized in simulation (24 and 21 transistor sizes; $B = 320$; 1.1–6.0 s per evaluation); one objective rewards the specifications met. Their problem cards convey different things: physical parameters whose effects a designer knows (chip flows, circuits), choices with well-known defaults (NB-301, the XGBoost benchmarks), and, on the knapsack, the whole instance, since each item's description carries its values and weight.

Baselines We compare three groups, each controlling for one part of the claim. (i) *No model*: the drop-in baselines random search and NSGA-II (Deb et al., 2002), and AGENTEVOLVE *without LLM*, the engine with the three model sources removed, which isolates the model's contribution. (ii) *Model only*: *LLM as optimizer*, the same model proposing designs from the scored history, as in OPRO (Yang et al., 2023), without our engine, which isolates the engine's contribution; it runs on every benchmark except the op-amp. (iii) *Bayesian specialists and the exact front*: qNEHVI (Daulton et al., 2021), a Gaussian-process Bayesian optimizer for several objectives, on the chip flows and the

Table 1: **Final hypervolume ratio at the evaluation budget**, median over validation seeds. We group rows by what each method uses. Markers give our paired verdict of AGENTEVOLVE against that row: ▲ significant or decisive, = level, ▽ behind. Best value per benchmark in bold; – not run. We compare the Bayesian specialists and the exact front in the text.

Method	Nangate45	ASAP7	SKY 130HD	NB-301	XGB-1489	XGB-40981	Knapsack	Op-amp	LDO
No model									
Random search	0.715▲	0.835▲	0.720▲	0.744▲	0.871=	0.891▽	0.582▲	0.229▲	0.645▲
NSGA-II	0.698▲	0.813▲	0.744=	0.794▲	**0.884**▽	0.876=	0.828▲	0.337▲	1.28▲
AgentEvolve without LLM	0.871▲	0.950▲	**0.867**▽	0.937▲	0.840=	**0.926**▽	0.963▲	0.462=	1.42=
Model only									
LLM as optimizer	0.942=	0.957▲	0.760=	0.894▲	0.860=	0.787▲	0.993▲	–	1.02▲
Model and engine									
AgentEvolve	**0.945**	**0.969**	0.807	**0.994**	0.874	0.871	**1.00**	**0.507**	**1.99**
Validation seeds	4	8	8	4	8	8	4	8	4

op-amp; HEBO (Cowen-Rivers et al., 2022), the Bayesian optimizer of OpenROAD's AutoTuner, on the chip flows; and the knapsack's exact front. We ablate against the *earlier design*, the more complex engine AGENTEVOLVE replaced (Section 6).

Implementation We run everything with one configuration and the cheapest tier of the GPT-5.6 model family (luna, reasoning effort high; Figure 6) and record every model call; calls cost \$0.05–\$0.32 per run (median \$0.11).

Comparison metrics The front of a run is the set of evaluated designs that no other evaluated design beats on every objective. We measure a front by its hypervolume (Zitzler & Thiele, 1999; Guerreiro et al., 2020), the volume of objective space that the front dominates up to a fixed reference point; it grows whenever a new design improves on the front, so it rewards both better and more varied trade-offs. The hypervolume ratio divides this volume by that of a sealed reference front (the exact front on the knapsack), so 1 matches the reference and values above 1 exceed it; on the circuits it counts only the designs the simulator accepts. We compare methods on two metrics: evaluations to the target (ETT), the number of evaluations a run needs to first reach a hypervolume ratio sealed from development runs (a run that never reaches it counts at the budget), and the final hypervolume ratio at the budget. We report each comparison as (ETT ratio, change in final hypervolume ratio), where the ETT ratio is the baseline's ETT over AGENTEVOLVE's, so values above 1 favour AGENTEVOLVE. Where evaluations are expensive, we also compare the wall-clock time to the target (TTT) and the hypervolume ratio at a matched time T.

Seeds and verdicts We made every design decision at *development seeds*, then froze the engine, sealed the verdict rules and read every result at *validation seeds* from a separate block. We run four validation seeds where the run-to-run spread of the final hypervolume ratio on development seeds is 2–4% (knapsack, LDO, NB-301, Nangate45) and eight elsewhere. We pair each comparison by seed. We call a comparison *significant* when a bias-corrected bootstrap (BCa) interval or a sign-flip test ($p < 0.05$) separates the two methods, and *decisive* when ETT is significant, AGENTEVOLVE wins at least three more runs than it loses, and its final hypervolume ratio is not behind. Where evaluations are expensive we apply the same rule to TTT and to the hypervolume ratio at a matched time, from a timestamp on every evaluation.

5 RESULTS

Comparison with drop-in baselines AGENTEVOLVE is decisive over NSGA-II on 5 of the 9 benchmarks: it needs fewer evaluations to the target and ends with more hypervolume on Nangate45 (4.65×, +36%), NB-301 (3.09×, +28%), the knapsack (+21%), the op-amp (2.11×, +57%) and the LDO (2.90×, +50%). On ASAP7 it is significant (+20%; 3/5/0 wins, ties and losses), on SKY130HD and XGB-40981 the two are level (+11% and +0.047%), and on XGB-1489 AGENTEVOLVE is behind NSGA-II (−1.6%; 1/3/4), a benchmark where every method ends within a few points of the others (Section 6). Against random search it is decisive on 5 of 9 benchmarks, significant on ASAP7 and SKY130HD, level on XGB-1489 and behind on XGB-40981 (−2.1%) (Table 1, Figure 4). The lead over NSGA-II opens within the first tens of evaluations on the knapsack, NB-301 and Nangate45 and after a quarter of the budget on the LDO (Figure 2).

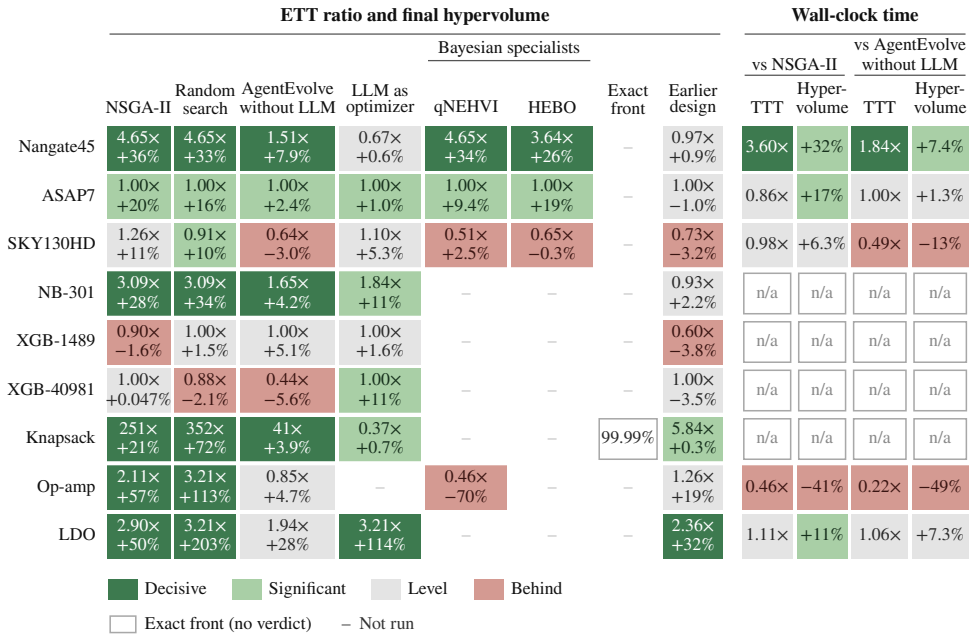


Figure 4: **Paired verdicts at validation seeds** Each cell pairs AGENTEVOLVE with one baseline: top, the ETT ratio; bottom, the change in final hypervolume ratio (Section 4). Colour gives our sealed verdict. Right: the TTT ratio and the hypervolume ratio at a matched time, read only where one evaluation is expensive.

Contributions of the model and of the engine Against AGENTEVOLVE without LLM, AGENTEVOLVE is significant or decisive on 4 of the 9 benchmarks: NB-301 (1.65×, +4.2%), the knapsack (41.0×), Nangate45 (1.51×, +7.9%) and ASAP7 (+2.4%). It is level on XGB-1489, the op-amp and the LDO, where its front has 28% more hypervolume but the difference is not significant at four seeds, and behind on SKY130HD (−3.0%; 2/0/6) and XGB-40981 (−5.6%; 0/1/7), the two benchmarks where the model's initial population starts behind the space-filling draws it replaces: after generation 0 its hypervolume ratio is below that of AGENTEVOLVE without LLM in six of eight runs on both, and the rest of the budget does not recover the difference. Against LLM as optimizer, AGENTEVOLVE is decisive on the LDO (3.21×, +114%), significant on NB-301 (+11%), ASAP7 (+1.0%), XGB-40981 (+11%) and the knapsack (+0.7%), where LLM as optimizer needs fewer evaluations to the target in three of four runs but ends with less hypervolume, and level on Nangate45, XGB-1489 and SKY130HD.

Distance to the exact front On the knapsack the median run of AGENTEVOLVE reaches 99.998% of the exact front's hypervolume, and every run is within 5×10^{-4} of it, through a sampler program the model wrote after reading the item descriptions. We generated the instance for this study, so the program cannot have memorised the optimum and must construct it.

Comparison with the Bayesian specialists On the chip flows AGENTEVOLVE is decisive over HEBO, the optimizer of OpenROAD's AutoTuner, and over qNEHVI on Nangate45 (3.64×, +26% and 4.65×, +34%) and significant over both on ASAP7 (+19% and +9.4%). On SKY130HD, where the model's initial population starts behind, it is behind both: they need fewer evaluations to the target in six of eight runs (2/0/6 against HEBO, 2/0/6 against qNEHVI), although the final hypervolume ratios end close (−0.3% and +2.5%). On the op-amp qNEHVI reaches a design that meets every specification in half of its runs and AGENTEVOLVE in none (−70%, Section 6).

6 ANALYSIS

Budget allocation across sources The credit should give the model sources budget where the card conveys the structure of good designs and take it away elsewhere, and it does under one configuration (Figure 5a). On the knapsack, whose card lists every item's values and weight, the sampler program has a 92% budget share and a 99.8% front share. On Nangate45 and ASAP7 agent proposals have

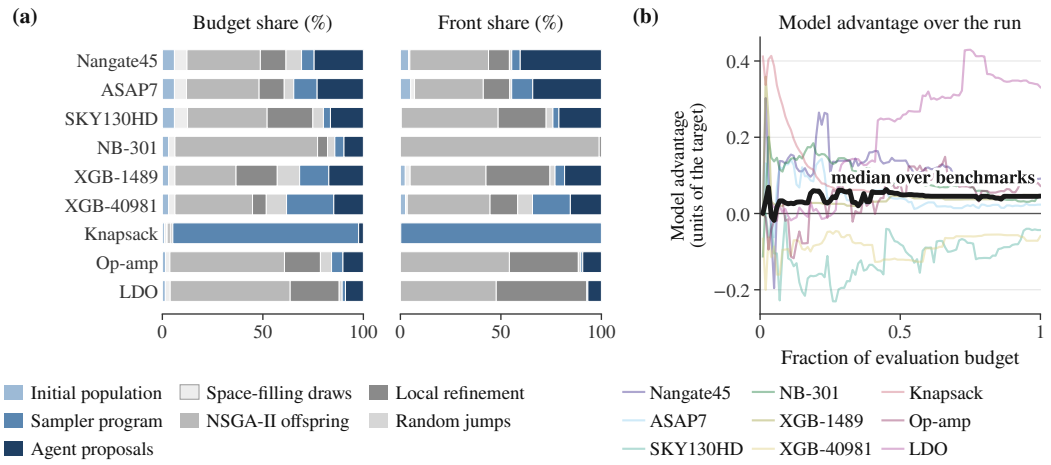


Figure 5: **Budget share, front share and model advantage** (a) Budget share and front share of each source for AGENTEVOLVE at validation seeds; blue shades are the model sources, greys the classical sources. (b) Model advantage, the paired difference in hypervolume ratio between AGENTEVOLVE and AGENTEVOLVE without LLM in units of the target, against the fraction of the budget spent (median per benchmark, faded; bold: the median over benchmarks).

Table 2: **Limits of the guidance** Median over validation seeds of the final hypervolume ratio on the XGBoost benchmarks and the op-amp, and the op-amp runs that reach a design meeting every specification; – not run.

	Final hypervolume ratio			
Method	XGB-1489	XGB-40981	Op-amp	Feasible op-amp runs
AgentEvolve	0.874	0.871	0.507	0 of 8
AgentEvolve without LLM	0.840	0.926	0.462	0 of 8
NSGA-II	0.884	0.876	0.337	0 of 8
Random search	0.871	0.891	0.229	0 of 8
LLM as optimizer	0.860	0.787	–	–
qNEHVI	–	–	2.12	4 of 8
Earlier design	0.900	0.863	0.473	0 of 8

a 23–25% budget share and a 34–40% front share, 1.5–1.7 times their budget share (one run in Appendix B). On NB-301 and the circuits the model sources have under 20% of the budget, and NSGA-II offspring and local refinement build the front.

Timing of the model advantage The model advantage (Figure 5b) appears within the first tenth of the budget on the knapsack and Nangate45 and is still positive at the end; on the LDO it appears after a quarter, once agent proposals and the local refinement around them pay, and keeps growing. Across benchmarks the median advantage is positive from 6% of the budget on and holds between 0.02 and 0.06 targets to the end.

Outcomes of the model's outputs The model's outputs are rarely malformed or wasted: the Semantic Adapter admits almost every initial-population design, agent proposals are almost never rejected and nearly always feasible, and the agent's predictions rank its proposals nearly as the flow does (Appendix A).

Wall-clock time A model call takes one to two minutes. On the chip flows a run's calls take 4–10% of its evaluator time, and on Nangate45 AGENTEVOLVE needs 1.84× less time to the target than AGENTEVOLVE without LLM and 3.60× less than NSGA-II; at a matched time its front has 17% more hypervolume than NSGA-II's on ASAP7 and 6.3% more on SKY130HD, where it is behind AGENTEVOLVE without LLM in seconds as in evaluations (Figure 4, right). On the op-amp (one-second simulations) model latency is 2.3× the evaluator time and AGENTEVOLVE is behind in seconds. The guidance pays in time where an evaluation costs about as much as a model call.

Budget floor On both XGBoost benchmarks every method, random search included, ends within a few points of the others at 192 evaluations (Table 2), and in earlier validation runs random search reached the target on XGB-40981 in 15 of 16 runs. The first draws decide these benchmarks, includ-

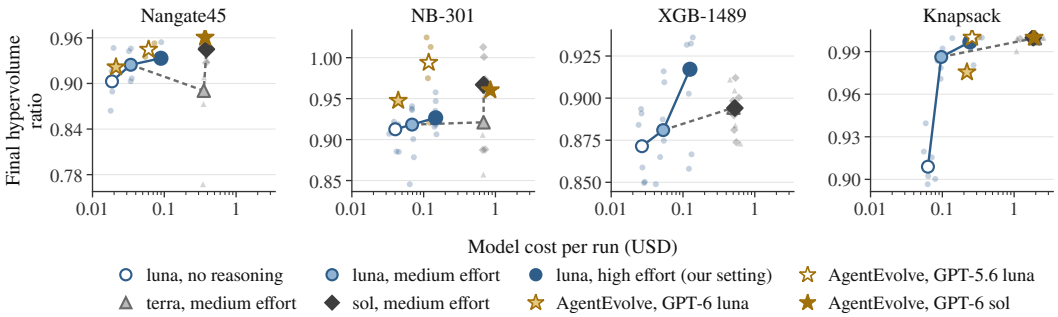


Figure 6: **Reasoning effort, model tier and model generation** We plot the final hypervolume ratio against the model cost of a run. Earlier design at development seeds: solid, the effort of the cheapest tier (GPT-5.6 luna: none, medium, high); dashed, three tiers at medium effort (luna, terra, sol); large markers medians, small ones runs. Stars: AGENTEVOLVE at high effort with GPT-5.6 luna (median of four validation seeds; gold dots) and with GPT-6 luna and sol at one of them (on the LDO instead of XGB-1489).

ing AGENTEVOLVE's deficit to NSGA-II on XGB-1489 (-1.6%) and to AGENTEVOLVE without LLM on XGB-40981 (-5.6%), which opens with the initial population.

Feasibility known only from measurements On the op-amp no run of AGENTEVOLVE reaches the thin region where every specification holds, while qNEHVI, a Gaussian process fitted to measurements and queried one candidate at a time, reaches it in half of its runs (Table 2). The card names 24 transistor sizes, and neither the initial population nor the agent proposals find the feasible set, because the card does not describe where it lies and only simulation reveals it.

Reasoning effort, model tier and model generation We repeat the earlier design across the reasoning effort of the cheapest tier and across three tiers of one family (Figure 6). Effort raises the median final hypervolume ratio on all four benchmarks at about four times the model cost and five to eight times the latency; the step from no reasoning is significant only on the knapsack (six of six runs; 0.909 to 0.997, ETT 43 rather than 681). A larger tier at medium effort costs ten to twenty times as much as the cheapest: the largest tier is decisive over the cheapest at medium effort only on the knapsack, where its ETT is 3 instead of 45, and the middle tier has a higher ETT than the cheapest on Nangate45 (all four runs) and NB-301 (five of six). Against the cheapest tier at high effort, the setting we use, the largest tier is level on all four benchmarks at four to eight times the cost. With GPT-6 luna and sol (high effort, one validation seed; stars) AGENTEVOLVE mostly stays within the spread of GPT-5.6 luna's four runs: GPT-6 sol has ETT 7 on Nangate45 instead of 18–34 at six times the model cost, and the cheaper GPT-6 luna ends lower on Nangate45 and the knapsack.

Recorded model calls and the simplified design The earlier design compared the agent's predictions with values in the engine's internal minimisation form, so its feedback called almost every prediction on a maximised objective supported and scored an exact model-written surrogate of the knapsack objective as perfectly anti-correlated; we found this only in the recorded calls. The Semantic Adapter removes the class by construction, and by deleting the mechanisms that never paid (the model-written surrogate, a hypothesis-verdict module, a dozen allocation rules) we cut the engine to a third of its size. Against the earlier design AGENTEVOLVE is significant or decisive on 2 benchmarks, level on 5, and behind on XGB-1489 (-3.8%) and SKY130HD (-3.2%).

7 CONCLUSION

We presented AGENTEVOLVE, a drop-in replacement for evolutionary optimizers that treats a language model as one more source of candidate designs rather than as the optimizer and lets each source's hypervolume gain decide its budget share. With the cheapest model tier, AGENTEVOLVE is decisive over NSGA-II on 5 of 9 benchmarks and over HEBO, the optimizer of OpenROAD's AutoTuner, on Nangate45, and we found from its per-source credit that the model pays where the card lets it infer the structure of good designs. Its gains are limited by what the card and the model's background knowledge convey. A Bayesian specialist wins where only measurements reveal a thin feasible region, a small budget cannot make up for a weak initial population, and below a budget floor no guidance helps. We have released AGENTEVOLVE as an open-source software project at https://anonymous.4open.science/r/agent_evolve-9C5A/.

AI DISCLOSURE

In this work, we used generative AI tools to implement methods (parts of the optimizer's code, the experiment harness and the figure scripts), to give feedback on the design of experiments, and to help interpret results. We have not used generative AI tools to generate synthetic data sets, develop theoretical models or conceptual frameworks, formulate or prove mathematical claims, or propose or refine hypotheses, and translation, dataset cleaning and qualitative data analysis are not applicable to this work. Additionally, we used generative AI tools to polish the writing, to retrieve and discover related literature, and to draft parts of the paper. We have reviewed all AI-assisted work: AI-written code is covered by the package's test suite, every number in the paper is generated by scripts from the saved run records, and we checked every cited work and every drafted passage against its source and our results. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

T. Ajayi, V. Chhabria, Mateus Fogaça, S. Hashemi, Abdelrahman Hosny, A. Kahng, Minsoo Kim, Jeongsup Lee, U. Mallappa, Marina Neseem, G. Pradipta, S. Reda, Mehdi Saligane, S. Sapatnekar, C. Sechen, M. Shalan, W. Swartz, Lutong Wang, Zhehong Wang, M. Woo, and Bangqi Xu. INVITED: Toward an open-source digital flow: First learnings from the OpenROAD project. In *Design Automation Conference (DAC)*, pp. 1–4, 2019. doi: 10.1145/3316781.3326334. URL https://doi.org/10.1145/3316781.3326334.

Shuvayan Brahmachary, S. Joshi, A. Panda, K. Koneripalli, A. Sagotra, H. Patel, Ankush Sharma, Ameya D. Jagtap, and K. Kalyanaraman. Large language model-based evolutionary optimizer: Reasoning with elitism. *ArXiv*, abs/2403.02054, 2024. doi: 10.48550/arXiv.2403.02054. URL https://doi.org/10.48550/arXiv.2403.02054.

Chih-Yu Chang, Milad Azvar, C. Okwudire, and Raed Al Kontar. LLINBO: Trustworthy LLM-in-the-loop Bayesian optimization. *ArXiv*, abs/2505.14756, 2025. doi: 10.48550/arXiv.2505.14756. URL https://doi.org/10.48550/arXiv.2505.14756.

Luís Da Costa, Á. Fialho, Marc Schoenauer, and M. Sebag. Adaptive operator selection with dynamic multi-armed bandits. pp. 913–920, 2008. doi: 10.1145/1389095.1389272. URL https://doi.org/10.1145/1389095.1389272.

A. Cowen-Rivers, Wenlong Lyu, Rasul Tutunov, Zhi Wang, Antoine Grosnit, Ryan-Rhys Griffiths, A. Maraval, Jianye Hao, Jun Wang, Jan Peters, and Haitham Bou-Ammar. HEBO: An empirical study of assumptions in Bayesian optimisation. *J. Artif. Intell. Res.*, 74:1269–1349, 2022. doi: 10.1613/jair.1.13643. URL https://doi.org/10.1613/jair.1.13643.

Sam Daulton, Maximilian Balandat, and E. Bakshy. Parallel Bayesian optimization of multiple noisy objectives with expected hypervolume improvement. pp. 2187–2200, 2021. URL https://arxiv.org/abs/2105.08195.

K. Deb, S. Agrawal, Amrit Pratap, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.*, 6:182–197, 2002. doi: 10.1109/4235.996017. URL https://doi.org/10.1109/4235.996017.

Amur Ghose, A. Kahng, Sayak Kundu, and Zhiang Wang. Orfs-agent: Tool-using agents for chip design optimization. *2025 ACM/IEEE 7th Symposium on Machine Learning for CAD (MLCAD)*, pp. 1–13, 2025. doi: 10.1109/MLCAD65511.2025.11189204. URL https://doi.org/10.1109/MLCAD65511.2025.11189204.

Andreia P. Guerreiro, C. Fonseca, and L. Paquete. The hypervolume indicator. *ACM Computing Surveys (CSUR)*, 54:1 – 42, 2020. doi: 10.1145/3453474. URL https://doi.org/10.1145/3453474.

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. *ArXiv*, abs/2309.08532, 2024. doi: 10.48550/arXiv.2309.08532. URL https://doi.org/10.48550/arXiv.2309.08532.

Bei-Chen Huang, Xing-Yu Wu, Yu Zhou, Jibin Wu, Liang Feng, Ran Cheng, and Kay Chen Tan. Exploring the true potential: Evaluating the black-box optimization capability of large language models. *ArXiv*, abs/2404.06290, 2024. doi: 10.48550/arXiv.2404.06290. URL https://doi.org/10.48550/arXiv.2404.06290.

S. KarthikSomayajiN and Peng Li. Llm-uso: Large language model-based universal sizing optimizer. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 45: 1907–1920, 2025. doi: 10.1109/TCAD.2025.3604643. URL https://doi.org/10.1109/TCAD.2025.3604643.

R. Lange, Yuki Imajuku, and Edoardo Cetin. Shinkaevolve: Towards open-ended and sample-efficient program evolution. *ArXiv*, abs/2509.19349, 2025. doi: 10.48550/arXiv.2509.19349. URL https://doi.org/10.48550/arXiv.2509.19349.

J. Lehman, J. Gordon, Shawn Jain, Kamal Ndousse, Cathy Yeh, and Kenneth O. Stanley. Evolution through large models. *ArXiv*, abs/2206.08896, 2022. doi: 10.48550/arXiv.2206.08896. URL https://doi.org/10.48550/arXiv.2206.08896.

Jintao Li, Haochang Zhi, Ruiyu Lyu, Wangzhen Li, Zhaori Bi, Keren Zhu, Yanhan Zeng, W. Shan, Changhao Yan, Fan Yang, Yun Li, and Xuan Zeng. AnalogGym: An open and practical testing suite for analog circuit synthesis (invited). *2024 ACM/IEEE International Conference On Computer Aided Design (ICCAD)*, pp. 1–9, 2024. doi: 10.1145/3676536.3697117. URL https://doi.org/10.1145/3676536.3697117.

Ke Li, Á. Fialho, S. Kwong, and Qingfu Zhang. Adaptive operator selection with bandits for a multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 18:114–130, 2014. doi: 10.1109/TEVC.2013.2239648. URL https://doi.org/10.1109/TEVC.2013.2239648.

Fei Liu, Xi Lin, Zhenkun Wang, Shunyu Yao, Xialiang Tong, Mingxuan Yuan, and Qingfu Zhang. Large language model for multi-objective evolutionary optimization. *ArXiv*, abs/2310.12541, 2023. doi: 10.48550/arXiv.2310.12541. URL https://doi.org/10.48550/arXiv.2310.12541.

Fei Liu, Xialiang Tong, Mingxuan Yuan, Xi Lin, Fu Luo, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. Evolution of heuristics: Towards efficient automatic algorithm design using large language model. In *International Conference on Machine Learning*, pp. 32201–32223, 2024a. URL https://arxiv.org/abs/2401.02051.

Siyi Liu, Chen Gao, and Yong Li. Large language model agent for hyper-parameter optimization. *ArXiv*, abs/2402.01881, 2024b. doi: 10.48550/arXiv.2402.01881. URL https://doi.org/10.48550/arXiv.2402.01881.

Tennison Liu, N. Astorga, Nabeel Seedat, and M. van der Schaar. Large language models to enhance Bayesian optimization. *ArXiv*, abs/2402.03921, 2024c. doi: 10.48550/arXiv.2402.03921. URL https://doi.org/10.48550/arXiv.2402.03921.

Wanyi Liu, Long Chen, and Zhenzhou Tang. Large language model aided multi-objective evolutionary algorithm: A low-cost adaptive approach. *ACM Transactions on Evolutionary Learning*, 2024d. doi: 10.1145/3787965. URL https://doi.org/10.1145/3787965.

Elliot Meyerson, M. Nelson, Herbie Bradley, Arash Moradi, Amy K. Hoover, and J. Lehman. Language model crossover: Variation through few-shot prompting. *ACM Transactions on Evolutionary Learning*, 4:1 – 40, 2023. doi: 10.1145/3694791. URL https://doi.org/10.1145/3694791.

Alexander Novikov, Ngân V u, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, S. Shirobokov, B. Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. P. Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, Matej Balog, and Google DeepMind. AlphaEvolve: A coding agent for scientific and algorithmic discovery. *ArXiv*, abs/2506.13131, 2025. doi: 10.48550/arXiv.2506.13131. URL https://doi.org/10.48550/arXiv.2506.13131.

Florian Pfisterer, Lennart Schneider, Julia Moosbauer, Martin Binder, and B. Bischl. Yahpo gym - an efficient multi-objective multi-fidelity benchmark for hyperparameter optimization. pp. 3/1–39, 2021. URL https://arxiv.org/abs/2109.03670.

Vishnu Raj and S. Kalyani. Taming non-stationary bandits: A bayesian approach. *ArXiv*, abs/1707.09727, 2017. URL https://arxiv.org/abs/1707.09727.

Carson Rodrigues, Oysturn Vas, Isaiah Abner DCosta, and Nithish Kumar Prabhakaran. When is an llm worth it for hyperparameter optimization? a budget-matched study on tabular data finds the warm-start is a default configuration, not the model. *ArXiv*, abs/2606.21641, 2026. doi: 10.48550/arXiv.2606.21641. URL https://doi.org/10.48550/arXiv.2606.21641.

B. Romera-Paredes, M. Barekatain, Alexander Novikov, Matej Balog, M. P. Kumar, Emilien Dupont, Francisco J. R. Ruiz, J. Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, Alhussein Fawzi, Joshua A. Grochow, Andrea Lodi, Jean-Baptiste Mouret, Talia Ringer, and Tao Yu. Mathematical discoveries from program search with large language models. *Nature*, 625: 468 – 475, 2023. doi: 10.1038/s41586-023-06924-6. URL https://doi.org/10.1038/s41586-023-06924-6.

Daniel Russo, Benjamin Van Roy, Abbas Kazerouni, and Ian Osband. A tutorial on thompson sampling. *Found. Trends Mach. Learn.*, 11:1–96, 2017. doi: 10.1561/2200000070. URL https://doi.org/10.1561/2200000070.

Niki van Stein and Thomas Bäck. LLaMEA: A large language model evolutionary algorithm for automatically generating metaheuristics. *IEEE Transactions on Evolutionary Computation*, 29: 331–345, 2024. doi: 10.1109/TEVC.2024.3497793. URL https://doi.org/10.1109/TEVC.2024.3497793.

Jia Xiong, Runkai Li, Chenxu Niu, Guangyuan Gao, Changwen Xing, Yifan Zhang, Xinlai Wan, Jieran Cui, Chen Bai, Yusheng Hua, Ying Wang, Ming Ling, Xi Wang, and Tao Xie. Microevo: Knowledge-guided llm sampling for efficient microarchitecture design space exploration. 2026. URL https://arxiv.org/abs/2608.06183.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. *ArXiv*, abs/2309.03409, 2023. doi: 10.48550/arXiv.2309.03409. URL https://doi.org/10.48550/arXiv.2309.03409.

Haoran Ye, Jiarui Wang, Zhiguang Cao, and Guojie Song. ReEvo: Large language models as hyper-heuristics with reflective evolution. *ArXiv*, abs/2402.01145, 2024. doi: 10.48550/arXiv.2402.01145. URL https://doi.org/10.48550/arXiv.2402.01145.

Yuxuan Yin, Yu Wang, Boxun Xu, and Peng Li. ADO-LLM: Analog design Bayesian optimization with in-context learning of large language models. *2024 ACM/IEEE International Conference On Computer Aided Design (ICCAD)*, pp. 1–9, 2024. doi: 10.1145/3676536.3676816. URL https://doi.org/10.1145/3676536.3676816.

Arber Zela, Julien Siems, Lucas Zimmer, Jovita Lukasik, Margret Keuper, and F. Hutter. Surrogate nas benchmarks: Going beyond the limited search spaces of tabular nas benchmarks. 2020. URL https://arxiv.org/abs/2008.09777.

Rui Zhang, Fei Liu, Xi Lin, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. Understanding the importance of evolutionary search in automated heuristic design with large language models. *ArXiv*, abs/2407.10873, 2024. doi: 10.48550/arXiv.2407.10873. URL https://doi.org/10.48550/arXiv.2407.10873.

E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Trans. Evol. Comput.*, 1999. doi: 10.1109/4235.797969. URL https://doi.org/10.1109/4235.797969.