BEYOND FINAL REGRET: STRESS-TESTING HYPERPARAMETER OPTIMIZATION CONCLUSIONS ACROSS EVALUATION REGIMES

Anonymous authors
Paper under double-blind review

ABSTRACT

Hyperparameter optimization (HPO) methods are commonly compared on fixed collections of tasks using aggregate final performance, implicitly treating the benchmark and evaluation protocol as fixed. We study a different question: *how sensitive are empirical conclusions about HPO methods to controlled changes in the evaluation regime?* We introduce BEYOND FINAL REGRET, a configurable benchmark generator that exposes search-space structure, fidelity reliability, cost regime, distribution shift, evaluation protocol, and backend realization as explicit experimental variables. The framework generates reproducible benchmark instances and retains objective, fidelity, incumbent, and cost trajectories, enabling evaluation through final quality, anytime behavior, quality–cost tradeoffs, conditional comparisons, and repeated-run distributions. We evaluate seven representative HPO methods across in-distribution, interpolation, extrapolation, cross-family, and targeted stress-test settings. The experiments show that several conclusions obtained from aggregate final performance are conditional on the evaluation regime. Relative performance changes with search-space structure, fidelity reliability, available resources, distribution shift, protocol, and backend, while other relationships remain stable across perturbations. In particular, methods with strong final quality can incur substantially larger computational costs, making recommendations dependent on the available resource budget rather than on a single cost-normalized scalar. These results position benchmark design itself as an experimental variable and provide an executable framework for measuring the sensitivity of HPO conclusions to that variable.

1 INTRODUCTION

Hyperparameter optimization (HPO) is an essential component of modern machine-learning pipelines, and new HPO methods are commonly evaluated on benchmark suites using final objective value, regret, or aggregate rank. Resources such as HPOBench (Eggensperger et al., 2021), YAHPO Gym (Pfisterer et al., 2022), HPO-B (Arango et al., 2021), and OpenML benchmarking suites (Bischl et al., 2017) have substantially improved the reproducibility and scale of these comparisons. However, the conclusions obtained from a benchmark are conditioned not only on the optimizer, but also on how the evaluation itself is constructed.

In particular, optimizer comparisons can depend on search-space structure, the reliability of low-fidelity evaluations, computational-cost variation, the distribution of benchmark instances, the evaluation protocol, and the backend used to realize an objective. These dependencies are especially relevant for multi-fidelity methods such as Hyperband (Li et al., 2018) and BOHB (Falkner et al., 2018), whose behavior depends on whether inexpensive partial-resource evaluations are informative about full-budget performance. Consequently, a method that appears strong under one benchmark regime need not retain the same relative behavior when the available budget, fidelity relationship, task distribution, or backend changes.

The broader principle that empirical optimization should be evaluated through carefully controlled experiments rather than simple algorithmic "horse races" is well established (Hooker, 1995; Barr et al., 1995; Rardin & Uzsoy, 2001). Our contribution is therefore not a new argument for robust

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.

experimentation, nor do we claim novelty for regret, rank analysis, repeated-run variability, multi-fidelity evaluation, or cost-aware comparison individually. Instead, we study how these principles can be operationalized in HPO by treating the *evaluation regime itself as an experimental variable*. Our central question is: *which conclusions about HPO methods remain stable when benchmark conditions are changed systematically while the optimizer set and evaluation pipeline are held fixed?*

To answer this question, we introduce BEYOND FINAL REGRET, a configurable benchmark generator and evaluation framework exposing six families of experimental conditions: search-space structure, fidelity reliability, cost regime, distribution shift, evaluation protocol, and backend realization. Rather than generating a single static task collection, the framework supports in-distribution, interpolation, extrapolation, and cross-family evaluations, together with targeted stress tests for structural difficulty, fidelity mismatch, budget sensitivity, protocol variation, and backend diversity. Each study is generated from a deterministic configuration, optimizers are executed while retaining objective, fidelity, incumbent, and cost trajectories, and the resulting artifacts are audited and analyzed across benchmark slices.

The framework is designed to study the *sensitivity of empirical conclusions*, rather than to produce a universal optimizer ranking. Accordingly, final regret is complemented by anytime trajectories, fixed-resource comparisons, quality-cost trade-offs, conditional ranking analyzes, and repeated-run distributions. Across seven representative HPO methods, the released studies show that some apparent performance relationships persist across evaluation conditions, whereas others change under different structural, fidelity, budget, split, protocol, or backend regimes. In particular, methods achieving strong final quality can operate in substantially different computational-cost regimes, making practical comparisons dependent on the resource budget rather than on a single aggregate score.

Our contributions are:

-
- We introduce a configurable HPO benchmark generator that treats search-space structure, fidelity reliability, cost regime, distribution shift, evaluation protocol, and backend realization as explicit experimental variables.
- We provide an end-to-end workflow for deterministic benchmark generation, optimizer execution, trajectory retention, auditing, and conditional reporting across evaluation regimes.
- We organize complementary evaluation views, including final quality, anytime behavior, fixed-budget comparison, quality-cost trade-offs, conditional ranks, and repeated-run variability, around the question of conclusion sensitivity.
- We provide empirical evidence across seven representative optimizers and multiple stress-test families showing which HPO conclusions persist and which are conditional on the evaluation regime.

BEYOND FINAL REGRET is therefore intended to complement, rather than replace, large fixed HPO benchmark suites. Its purpose is to provide a controlled experimental instrument for determining whether conclusions obtained under one evaluation setting continue to hold when the assumptions defining that setting are systematically perturbed.

2 RELATED WORK

HPO algorithms and benchmark infrastructures. HPO spans simple model-free strategies such as random search (Bergstra & Bengio, 2012), Bayesian and model-based optimization (Snoek et al., 2012; Hutter et al., 2011), and increasingly large collections of reproducible benchmark problems. HPOBench (Eggensperger et al., 2021), YAHPO Gym (Pfisterer et al., 2022), HPO-B (Arango et al., 2021), and OpenML benchmarking suites (Bischl et al., 2017) provide standardized interfaces, task collections, and inexpensive tabular or surrogate evaluations. Other resources target particular evaluation regimes, including high-dimensional HPO in LassoBench (Šehić et al., 2022), federated HPO in FedHPO-Bench (Wang et al., 2023), and joint architecture hyperparameter optimization in JAHS-Bench-201 (Bansal et al., 2022). Closely related NAS benchmarks such as NAS-Bench-101 (Ying et al., 2019), NAS-Bench-201 (Dong & Yang, 2020), and the surrogate-based NAS-Bench-301 (Siems et al., 2020) further demonstrate how search-space and benchmark construction

affect reproducible algorithm comparison, while repositories such as TabRepo (Salinas & Erickson, 2023) enable large-scale analysis from precomputed evaluations. These resources substantially improve comparability, but typically evaluate algorithms within a specified benchmark regime. Our focus is complementary: we systematically perturb properties of that regime and examine whether the conclusions themselves remain stable.

Multi-fidelity, budget, and cost-aware HPO. Resource-aware optimization has developed along several complementary directions. Freeze–Thaw Bayesian optimization uses partial learning curves to allocate training effort adaptively (Swersky et al., 2014), while FABOLAS models both performance and computational cost as dataset size varies (Klein et al., 2017). Hyperband (Li et al., 2018) formalizes aggressive resource allocation and early stopping, ASHA (Li et al., 2020) adapts this principle to massively parallel settings, and BOHB (Falkner et al., 2018) combines bandit-based allocation with model-based search. Benchmark suites such as HPOBench and YAHPO Gym already support multi-fidelity evaluation (Eggensperger et al., 2021; Pfisterer et al., 2022). Our contribution is therefore not multi-fidelity or cost-aware evaluation itself. Instead, fidelity reliability and computational cost are treated as controlled *stress variables*: the same optimizer set can be evaluated when low-fidelity signals are reliable, delayed, distorted, or misleading, and when conclusions are viewed at different resource levels. We consequently emphasize anytime trajectories, fixed-budget comparisons, and quality–cost trade-offs rather than assuming a single scalarization provides a universally appropriate notion of efficiency.

Experimental algorithmics and benchmarking methodology. The methodological motivation for this work has a substantially longer history than modern HPO. Hooker called for an empirical science of algorithms (Hooker, 1994) and later criticized comparative "horse races" that report performance without characterizing the conditions responsible for it (Hooker, 1995). Barr et al. (Barr et al., 1995), McGeoch (McGeoch, 1996), and Rardin and Uzsoy (Rardin & Uzsoy, 2001) emphasized experimental design, instance selection, computational effort, variability, and reproducible reporting. Similar concerns have been raised in evolutionary computation (Eiben & Jelasity, 2002), statistical comparisons across multiple problems (Demšar, 2006), and black-box optimization benchmarking through platforms such as COCO (Hansen et al., 2021); recent synthesis further identifies problem specification, performance measures, analysis, experimental design, and reproducibility as central benchmarking requirements (Bartz-Beielstein et al., 2020). We build directly on this literature and explicitly avoid claiming novelty for ranks, regret, repeated-run statistics, fidelity-aware evaluation, or cost analysis individually. The contribution of BEYOND FINAL REGRET is their HPO-specific operationalization in a single executable generator that jointly controls search-space structure, fidelity reliability, cost regime, distribution shift, protocol, and backend. This makes the *sensitivity of an empirical HPO conclusion* rather than another aggregate leaderboard the primary object of study.

3 CONTROLLED BENCHMARK AND STRESS-TEST FRAMEWORK

3.1 BENCHMARK MODEL

Rather than treating an HPO benchmark as a fixed collection of tasks, we model it as a generator over structured benchmark instances. An instance is represented as

$$b = (\mathcal{X}, f, c, \phi, \psi, \beta)\,, \tag{1}$$

where $\mathcal{X}$ is the configuration space, $f(\mathbf{x}, r)$ is the objective response of configuration $\mathbf{x}$ at resource level r, $c(\mathbf{x}, r)$ is the associated evaluation cost, ψ contains split metadata, and β identifies the backend realization.

The controlled benchmark conditions are collected in

$$\phi = (s, q, g, h, p), \qquad \theta = (\phi, \beta), \tag{2}$$

where s denotes search-space structure, q fidelity reliability, g the cost regime, h the distribution-shift regime, and p the evaluation protocol. Given condition descriptor θ and random seed ω, the generator instantiates

$$b = G(\theta, \omega). \tag{3}$$

This captures the central design principle of BEYOND FINAL REGRET: benchmark conditions are explicit experimental inputs rather than fixed, implicit properties of a task collection.

Table 1: Controlled axes in BEYOND FINAL REGRET.

Axis	Implemented regimes
Search space	continuous, mixed, conditional, hierarchical
Fidelity reliability	reliable, delayed, heterogeneous, rank-flipped, locally misleading, anti-correlated
Cost regime	uniform, skewed, heavy-tailed, bursty, configuration-dependent
Distribution shift	in-distribution, interpolation, extrapolation, cross-family
Evaluation protocol	fixed seed, seed bag, reshuffled split, metric and anytime variants
Backend realization	surrogate, real-task, external connector, hybrid

For evaluation, the common reference objective is the response at maximum resource,

$$f_b^\star(\mathbf{x}) = f(\mathbf{x}, r_{\max}), \tag{4}$$

which provides a common basis for assessing the configuration ultimately returned by each optimizer, including methods that use lower fidelities during search.

The six axes correspond to distinct sources of conditionality in an HPO conclusion. Search-space structure determines what configurations can be explored; fidelity reliability controls how informative inexpensive partial-resource evaluations are; the cost regime determines their computational consequences; distribution shift determines where conclusions are expected to transfer; the evaluation protocol controls how evidence is collected; and backend realization determines how the objective is instantiated.

Controlled instance generation and analysis of conditions of success are established principles in empirical algorithmics (Hooker, 1995; Barr et al., 1995; Rardin & Uzsoy, 2001; Bartz-Beielstein et al., 2020). We therefore do not claim novelty for controlled experimentation itself. The contribution is its HPO-specific operationalization through a generator that jointly exposes these six conditions while retaining a common execution, auditing, and reporting pipeline.

3.2 SPLIT FAMILIES AND CONTROLLED SHIFT

Let $\mathcal{B}$ denote the generated benchmark collection. Each study organizes its instances into mutually disjoint partitions,

$$\mathcal{B} = \mathcal{B}_{\text{train}} \cup \mathcal{B}_{\text{val}} \cup \mathcal{B}_{\text{test}}, \qquad \mathcal{B}_i \cap \mathcal{B}_j = \varnothing \quad (i \neq j). \tag{5}$$

These partitions structure benchmark construction and evaluation; they are not training sets for the HPO algorithms. In the *in-distribution* regime, held-out instances come from the same broad family. *Interpolation* withholds combinations or regions inside represented families, *extrapolation* evaluates structurally shifted or more difficult regions, and *cross-family* evaluation withholds qualitatively different family groups. The split design therefore tests whether benchmark conclusions transfer beyond the conditions from which they were obtained, complementing fixed collections such as OpenML suites and HPO-B (Bischl et al., 2017; Arango et al., 2021).

3.3 STRESS-TEST FAMILIES

The released benchmark instantiates the controlled axes at three levels. First, a core study provides the broad aggregate comparison. Second, four dedicated studies isolate in-distribution, interpolation, extrapolation, and cross-family behavior. Third, targeted studies independently stress search-space structure, fidelity trustworthiness, computational-cost behavior, distribution shift, protocol variation, and backend realization.

A separate misleading-fidelity ablation tests failure of a central assumption underlying resource-aware HPO: that inexpensive evaluations provide useful information about full-fidelity performance. Hyperband, ASHA, and BOHB exploit partial-resource evaluations to allocate computational effort (Li et al., 2018; 2020; Falkner et al., 2018); the benchmark therefore includes conditions in which fidelity information is reliable as well as delayed, heterogeneous, rank-flipped, or misleading. Real-task and external-connector extensions exercise the same interface beyond the controlled surrogate studies and provide complementary evidence on backend dependence.

3.4 Executable Workflow and Extensibility

Every experiment begins from a study manifest specifying the active benchmark conditions, scenario catalog, split family, optimizer set, evaluation budget, repetitions, and random seed. The implementation follows the common workflow

$$\textsc{Configure} \rightarrow \textsc{Generate} \rightarrow \textsc{Execute} \rightarrow \textsc{Audit} \rightarrow \textsc{Report}.$$

Configuration produces a deterministic benchmark blueprint; registered optimizers are then executed while retaining objective, fidelity, incumbent, evaluation-cost, and cumulative-cost traces. The audit stage checks split separation, expected configurations, and run completeness, and the reporting stage produces instance-, scenario-, backend-, budget-, trajectory-, and aggregate views.

The workflow also defines how the benchmark is extended. A new optimizer implements and registers the common optimizer interface, while new conditions within an existing axis are configuration-driven. Introducing a fundamentally new axis requires extending the condition descriptor and instance generator so that the property is recorded and available as a reporting slice. For example, to determine whether a new optimizer depends on reliable low-fidelity information, a user registers the optimizer, selects the fidelity-trustworthiness or misleading-fidelity study, executes it alongside the reference methods, and compares full-fidelity quality, cumulative-cost trajectories, and per-instance outcomes across fidelity regimes.

Overall, Section 3 specifies *what is varied* in the benchmark, whereas in Section 4 we specify *how the resulting HPO evidence is collected and compared.*

4 Experimental Design and Evaluation Protocol

4.1 Optimizers and Study Configuration

We evaluate seven representative HPO strategies spanning complementary search and resource-allocation paradigms:

$$\mathcal{M} = \{\textsc{Random}, \textsc{Sobol}, \text{TPE}, \text{BO}, \textsc{Hyperband}, \text{ASHA}, \text{BOHB}\}. \tag{6}$$

Random search provides a model-free baseline (Bergstra & Bengio, 2012), Sobol sampling represents low-discrepancy search (Sobol, 1967), TPE and surrogate Bayesian optimization represent model-based HPO (Bergstra et al., 2011; Snoek et al., 2012), and Hyperband, ASHA, and BOHB represent multi-fidelity resource allocation (Li et al., 2018; 2020; Falkner et al., 2018). The set is intentionally compact: the methods serve as qualitatively different probes of conclusion sensitivity rather than as an exhaustive optimizer leaderboard.

The core study generates 96 benchmark instances and evaluates its 72-instance held-out extrapolation partition using five repetitions per method and a nominal budget of 44 evaluation opportunities. Four dedicated studies examine in-distribution, interpolation, extrapolation, and cross-family regimes, while targeted studies perturb search-space structure, fidelity reliability, cost behavior, distribution shift, protocol, and backend realization. Principal stress studies use five repetitions and budgets between 28 and 40 evaluations; smaller real-task and external-connector extensions use three repetitions and a budget of 12. Study-specific instance counts, budgets, and repetitions are summarized in Appendix A.1, while detailed train/validation/test split counts are reported in Appendix B.1. Complete study manifests are provided with the released artifact.

4.2 Common Execution Protocol

All optimizers interact with the same benchmark-instance interface. For method m, instance b, and one execution, the framework retains the complete trace

$$\mathcal{T}_{m,b} = \{(\mathbf{x}_t, r_t, y_t, \kappa_t)\}_{t=1}^{T_{m,b}}, \tag{7}$$

where $\mathbf{x}_t$ is the queried configuration, r_t its resource level, $y_t = f(\mathbf{x}_t, r_t)$ the observed objective, and $\kappa_t = c(\mathbf{x}_t, r_t)$ the incurred evaluation cost. The incumbent and cumulative cost are additionally retained after every evaluation, allowing final-quality, anytime, and fixed-cost analyses to be reconstructed from the same execution records.

At the end of the search budget, each optimizer returns an incumbent $\hat{\mathbf{x}}_{m,b}$. Its full-fidelity objective value and total search cost are

$$F_{m,b} = f(\hat{\mathbf{x}}_{m,b}, r_{\max}), \qquad C_{m,b} = \sum_{t=1}^{T_{m,b}} \kappa_t. \tag{8}$$

Thus, methods may differ internally in search logic and fidelity allocation, but their returned configurations are assessed using the same maximum-fidelity objective and the same cost-accounting mechanism. This distinction is essential when comparing single-fidelity methods with Hyperband-, ASHA-, and BOHB-style resource allocation.

4.3 Primary Quality Measure

Because raw objective scales differ across generated instances, performance is calibrated separately for each instance. Let $\bar{F}_{m,b}$ denote the mean full-fidelity outcome of method m on instance b across repetitions. The calibration anchors are the best and worst method-level means on that instance, $F_b^{\min} = \min_m \bar{F}_{m,b}$ and $F_b^{\max} = \max_m \bar{F}_{m,b}$. We then define

$$R_{m,b} = \frac{\bar{F}_{m,b} - F_b^{\min}}{F_b^{\max} - F_b^{\min} + \epsilon}, \tag{9}$$

where lower values are better and $\epsilon > 0$ prevents numerical degeneracy. Mean normalized regret over the declared evaluation instances provides the conventional aggregate final-quality view.

We use this aggregate as a compact description of a specified benchmark regime, not as a context-independent optimizer ranking. Instance-level distributions, split- and scenario-specific results, and repeated-run records are retained to determine whether the aggregate conclusion is representative of the underlying evidence.

4.4 Conclusion-Oriented Evaluation Views

The central question is whether a conclusion obtained from the aggregate quality view remains supported when the evaluation conditions change. We therefore complement mean normalized regret with three classes of evidence.

Conditional breakdowns. We compare the aggregate quality ordering with orderings obtained on meaningful scenario, split, protocol, fidelity, and backend slices. Rank displacement is used descriptively to locate where an aggregate conclusion persists or changes; it is not treated as an independent performance criterion. Method-specific displacement is additionally visualized through the conclusion-sensitivity heatmap in Section 5.3. Formal definitions are given in Appendix A.

Resource-aware behavior. Equal evaluation counts need not imply equal computational expenditure when resource levels have different costs. We therefore avoid using a single regret–cost scalarization as the primary efficiency measure. Instead, the retained traces support: (i) incumbent normalized regret versus cumulative cost, (ii) comparisons under common instance-specific cost limits at 25%, 50%, and 100% of the reference budget, and (iii) terminal quality–cost Pareto analysis. These views make the assumed resource constraint explicit rather than embedding it in a universal cost-adjusted ranking.

Repeated-run variability. Repeated executions are analyzed at the method–instance level. Rather than relying on an average standard deviation as a standalone "stability score," we retain the distribution of within-instance variation across benchmark instances and summarize it using the median, interquartile range, and distributional plots. This distinguishes optimization quality from repeatability and exposes cases in which an aggregate summary hides heterogeneous behavior.

Our reporting principle is therefore deliberately hierarchical: aggregate quantities provide an entry point to a declared study, while distributions, trajectories, and conditional breakdowns determine how broadly that aggregate conclusion should be interpreted. When an aggregate ordering conflicts with an important controlled breakdown, the conditional evidence governs the interpretation. Exact definitions of rank displacement, fixed-cost reconstruction, Pareto dominance, anytime evaluation, and repeated-run dispersion are provided in Appendix A.

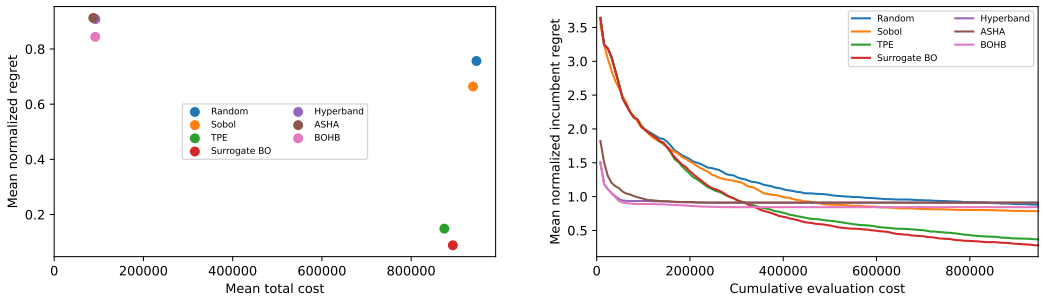


Figure 1: Resource-aware evaluation on the core benchmark. **Left:** mean terminal normalized regret versus mean total cost. ASHA, BOHB, TPE, and surrogate BO form the empirical non-dominated set, spanning distinct quality–cost regimes. **Right:** mean incumbent normalized regret versus cumulative evaluation cost reconstructed from the retained traces.

5 Results: When Do HPO Conclusions Persist or Change?

We evaluate the central question of this work: whether an HPO conclusion obtained under one evaluation regime remains supported when the regime is changed in a controlled way. We first establish the conventional final-quality conclusion, then condition it on computational cost, benchmark shift, controlled stress factors, and repeated execution. All secondary analyses use the traces retained from the same optimizer runs.

5.1 Aggregate Quality Is Informative but Not Universal

On the 72 held-out instances of the core benchmark, surrogate BO obtains the lowest mean normalized regret (0.0886), followed by TPE (0.1489); the complete seven-method comparison is reported in Appendix B.2. This establishes a clear aggregate final-quality conclusion, but does not imply instance-wise dominance. Indeed, surrogate BO has lower normalized regret than TPE on $43/72$ instances (59.7%), while TPE is lower on the remaining $29/72$. The paired mean difference, $R_{\mathrm{TPE}} - R_{\mathrm{BO}}$, is 0.0603; an instance-bootstrap analysis gives a descriptive 95% interval of approximately $[0.002, 0.118]$. Thus, the aggregate advantage is persistent enough to be visible at the collection level but heterogeneous across individual benchmark instances. The corresponding ECDF is provided in Appendix B.3.

5.2 Resource Constraints Change the Supported Conclusion

The aggregate quality result changes meaning once computational expenditure is made explicit. Surrogate BO and TPE incur mean total costs of approximately 8.93×10^5 and 8.75×10^5, respectively, whereas BOHB, Hyperband, and ASHA operate near 0.9×10^5 cost units. Figure 1 therefore separates terminal quality from computational cost rather than combining them into a single scalar score.

The terminal Pareto view contains four non-dominated operating points. ASHA occupies the lowest-cost extreme; BOHB improves final quality at a similar cost scale; TPE moves to the low-regret/high-cost region; and surrogate BO achieves the lowest aggregate terminal regret. Consequently, the terminal data do not support a unique cost-aware ordering without an explicit resource preference. We next impose common instance-specific cost limits. For instance b, let $B_{\mathrm{ref},b}$ be the largest method-level mean terminal cost. Table 2 evaluates the best completed incumbent at fractions of this reference cost.

The supported conclusion is therefore budget-indexed. BOHB is strongest under the tightest common cost constraint, whereas surrogate BO becomes strongest once approximately half of the reference cost is available. This crossover, also visible in Figure 1, is more informative than replacing the final-quality leaderboard with a second scalarized "efficiency" leaderboard.

Table 2: Core-study conclusions at common cost checkpoints. Values are mean normalized regret at the corresponding cost limit.

Cost limit	Lowest regret	Next lowest
$0.25B_{\mathrm{ref},b}$	BOHB (0.8440)	Hyperband (0.9082)
$0.50B_{\mathrm{ref},b}$	Surrogate BO (0.6306)	TPE (0.6590)
$B_{\mathrm{ref},b}$	Surrogate BO (0.1000)	TPE (0.1901)

5.3 Controlled Perturbations Separate Persistence from Sensitivity

The split studies first reveal an important persistence result. Surrogate BO retains the lowest aggregate normalized regret under in-distribution (0.0749), interpolation (0.1096), extrapolation (0.0934), and cross-family (0.1308) evaluation. Changing the split family therefore does not automatically overturn the aggregate conclusion. However, this persistence is not uniform within the splits: TPE becomes the lowest-regret method on the core-MLP slice in the in-distribution study and on the core-Transformer slice under interpolation, extrapolation, and cross-family evaluations. The targeted studies make this distinction between global persistence and local sensitivity more explicit. Figure 2 reports, for each optimizer and study, its mean absolute rank displacement across scenario slices relative to its aggregate quality rank. The formal definition is given in Appendix A.4.

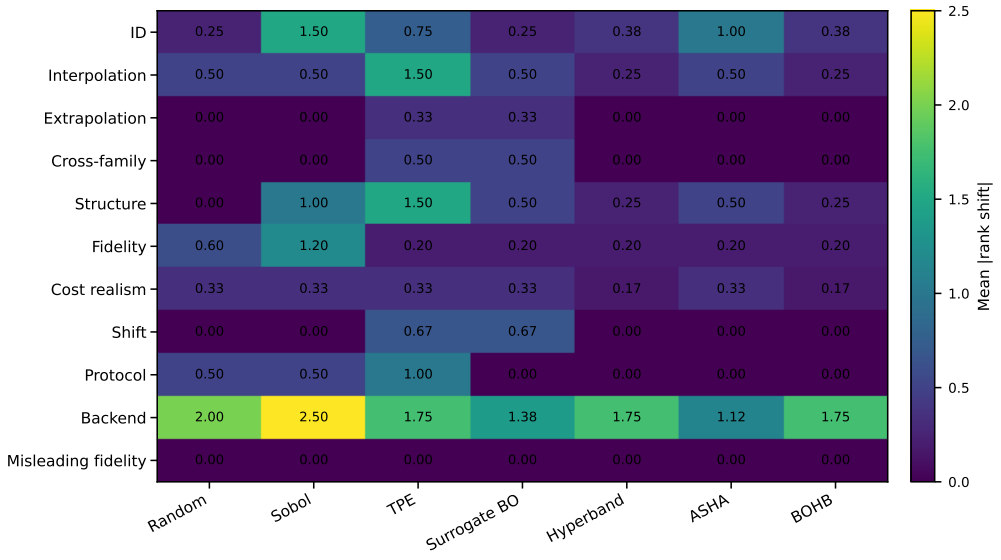


Figure 2: Conclusion sensitivity across controlled evaluation regimes. Each cell reports the mean absolute displacement of an optimizer's quality rank across scenario slices relative to its aggregate rank within that study. Lower values indicate greater preservation of the aggregate ordering.

Different perturbations expose distinct forms of conditionality. Structural stress changes the aggregate low-regret method from surrogate BO to TPE, while the continuous-SVM slice favors surrogate BO. Fidelity and cost stress preserve surrogate BO in aggregate but favor TPE on specific CNN and Transformer slices; protocol variation preserves the leading method while changing lower ranks.

Backend variation produces the largest displacement. The aggregate study favors surrogate BO, whereas Sobol leads on the real-task SVM slice, and BOHB, Hyperband, and TPE tie for the lowest regret on HPOBench-XGBoost. The maximum mean rank displacement is 2.29, with one method moving six positions. Because these slices are small diagnostic probes, they indicate backend sensitivity rather than general optimizer superiority. Low displacement denotes consistency *within* a study, not agreement with the core benchmark; the misleading-fidelity ablation illustrates this distinction.

5.4 Mechanism Stress and Run-to-Run Variation

The misleading-fidelity ablation isolates one mechanism behind conclusion changes. TPE and surrogate BO achieve mean normalized regrets of 0.1040 and 0.1719, whereas Hyperband, ASHA, and BOHB remain near 0.98. Although the multi-fidelity methods are substantially cheaper, misleading low-fidelity observations prevent those savings from translating into competitive full-fidelity quality, showing that their advantage depends on fidelity informativeness. Repeated runs reveal additional heterogeneity. We therefore examine within-instance dispersion across seeds rather than relying on a single average stability statistic. Sobol is nearly deterministic on many instances, while stochastic model-based and resource-allocation methods show broader variation. Full distributions are reported in Appendix B.9.

6 Discussion and Limitations

Our results support a more specific conclusion than simply stating that HPO rankings are unstable. Some conclusions persist across substantial changes in the evaluation regime, whereas others depend on computational budget, search-space structure, fidelity, reliability, task slice, or backend. In particular, aggregate final-quality conclusions remain stable across the principal split families, while resource-aware and scenario-level analyzes reveal important conditional changes. HPO conclusions should therefore be reported together with the evaluation conditions under which they have been tested.

This perspective follows the longstanding view in empirical algorithmics that benchmarking should study distributions, trajectories, and conditions of success rather than reduce evaluation to an algorithm "horse race" (Hooker, 1995; Barr et al., 1995; Rardin & Uzsoy, 2001). We do not claim novelty for this general principle or for regret, ranks, Pareto analysis, or repeated-run statistics individually. The contribution of Beyond Final Regret is their HPO-specific operationalization through a generator that jointly controls search-space structure, fidelity reliability, cost regime, split family, protocol, and backend. This complements established HPO benchmarks (Eggensperger et al., 2021; Pfisterer et al., 2022; Arango et al., 2021): fixed benchmark collections support reproducible comparison, whereas controlled generation allows the robustness of those comparisons to be stress-tested.

The study has several limitations. The generated and surrogate instances isolate selected HPO mechanisms but cannot reproduce all dependencies present in real workloads; the real-task and connector studies are intentionally small and should be viewed as diagnostic rather than population-level evidence. The seven optimizers provide broad methodological coverage but do not represent the full HPO design space. Normalized regret is defined relative to the evaluated methods on each instance and should therefore be interpreted within the declared comparison set. Finally, robustness is established only with respect to the perturbations actually tested: persistence under these regimes is not evidence of universal robustness, while a reversal under a stress condition does not imply that the condition is common in practice. The benchmark is therefore intended to complement, rather than replace, evaluation on large real-world HPO suites.

7 Conclusion

We introduced Beyond Final Regret, a generator-based framework for stress-testing whether empirical conclusions about HPO methods persist under controlled changes in search-space structure, fidelity reliability, cost regime, split family, protocol, and backend. Rather than replacing one leaderboard with another, the framework combines common execution traces with distributional, conditional, and resource-aware views that expose when an aggregate conclusion is stable and when it becomes regime dependent. Across the experimental studies, some conclusions remain consistent, notably the aggregate final-quality advantage of surrogate BO across the principal split families, while others change under tighter computational budgets, structural and fidelity perturbations, or backend variation. These results reinforce the need to report HPO performance together with the conditions under which the evidence was obtained. Beyond Final Regret is intended as a complementary evaluation instrument for testing such robustness systematically, reproducibly, and beyond final regret summaries alone.

AI USE STATEMENT

We used generative AI tools to assist with drafting and editing manuscript text, improving clarity, and organizing the presentation of the work. We also used these tools for code debugging assistance where applicable. The authors designed the experiments, executed the evaluation pipeline, inspected the resulting data, verified the reported numbers against the experimental artifacts, and determined the interpretations and conclusions. All AI-assisted text and code incorporated into the work were reviewed by the authors, and code changes were tested before use. We take responsibility for the final content of this work, including any text or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies how benchmark design and evaluation protocols affect conclusions about hyperparameter optimization methods. It does not involve human participants or the collection of personal information. The generated benchmark scenarios are controlled experimental settings and should not be interpreted as representative of every real-world HPO task. We report conditional findings and limitations to reduce the risk of treating the observed rankings as universal recommendations.

REPRODUCIBILITY STATEMENT

The paper specifies the benchmark-generation axes, study configurations, optimizer set, evaluation budgets, random seeds, metrics, and analysis protocol. The accompanying artifact (in the Supplementary material) contains the implementation, study manifests, execution and analysis scripts, and recorded results needed to reproduce the reported comparisons. We retain per-run objective, fidelity, incumbent, and cost trajectories so that aggregate results and resource-aware analyzes can be checked. Additional study details and results are provided in the appendix.

ACKNOWLEDGMENTS

All acknowledgments are anonymous.

REFERENCES

Sebastian Pineda Arango, Hadi S Jomaa, Martin Wistuba, and Josif Grabocka. Hpo-b: A large-scale reproducible benchmark for black-box hpo based on openml. 2021.

Archit Bansal, Danny Stoll, Maciej Janowski, Arber Zela, and Frank Hutter. Jahs-bench-201: A foundation for research on joint architecture and hyperparameter search. volume 35, pp. 38788–38802, 2022.

Richard S Barr, Bruce L Golden, James P Kelly, Mauricio GC Resende, and William R Stewart Jr. Designing and reporting on computational experiments with heuristic methods. *Journal of heuristics*, 1(1):9–32, 1995.

Thomas Bartz-Beielstein, Carola Doerr, Daan van den Berg, Jakob Bossek, Sowmya Chandrasekaran, Tome Eftimov, Andreas Fischbach, Pascal Kerschke, William La Cava, Manuel Lopez-Ibanez, et al. Benchmarking in optimization: Best practice and open issues. *arXiv preprint arXiv:2007.03488*, 2020.

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. volume 24, 2011.

Bernd Bischl, Giuseppe Casalicchio, Matthias Feurer, Pieter Gijsbers, Frank Hutter, Michel Lang, Rafael G Mantovani, Jan N van Rijn, and Joaquin Vanschoren. Openml benchmarking suites. *arXiv preprint arXiv:1708.03731*, 2017.

Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan):1–30, 2006.

Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. 2020.

Katharina Eggensperger, Philipp Müller, Neeratyoy Mallik, Matthias Feurer, René Sass, Aaron Klein, Noor Awad, Marius Lindauer, and Frank Hutter. Hpobench: A collection of reproducible multi-fidelity benchmark problems for hpo. 2021.

Agoston E Eiben and Márk Jelasity. A critical note on experimental research methodology in ec. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*, volume 1, pp. 582–587. IEEE, 2002.

Stefan Falkner, Aaron Klein, and Frank Hutter. Bohb: Robust and efficient hyperparameter optimization at scale. In *International conference on machine learning*, pp. 1437–1446. PMLR, 2018.

Nikolaus Hansen, Anne Auger, Raymond Ros, Olaf Mersmann, Tea Tušar, and Dimo Brockhoff. Coco: A platform for comparing continuous optimizers in a black-box setting. *Optimization Methods and Software*, 36(1):114–144, 2021.

John N Hooker. Needed: An empirical science of algorithms. *Operations research*, 42(2):201–212, 1994.

John N Hooker. Testing heuristics: We have it all wrong. *Journal of heuristics*, 1(1):33–42, 1995.

Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *International conference on learning and intelligent optimization*, pp. 507–523. Springer, 2011.

Aaron Klein, Stefan Falkner, Simon Bartels, Philipp Hennig, and Frank Hutter. Fast bayesian optimization of machine learning hyperparameters on large datasets. In *Artificial intelligence and statistics*, pp. 528–536. PMLR, 2017.

Liam Li, Kevin Jamieson, Afshin Rostamizadeh, Ekaterina Gonina, Jonathan Ben-Tzur, Moritz Hardt, Benjamin Recht, and Ameet Talwalkar. A system for massively parallel hyperparameter tuning. volume 2, pp. 230–246, 2020.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of machine learning research*, 18(185):1–52, 2018.

Catherine C McGeoch. Toward an experimental method for algorithm simulation. *INFORMS Journal on Computing*, 8(1):1–15, 1996.

Florian Pfisterer, Lennart Schneider, Julia Moosbauer, Martin Binder, and Bernd Bischl. Yahpo gym-an efficient multi-objective multi-fidelity benchmark for hyperparameter optimization. In *International Conference on Automated Machine Learning*, pp. 3–1. PMLR, 2022.

Ronald L Rardin and Reha Uzsoy. Experimental evaluation of heuristic optimization algorithms: A tutorial. *Journal of Heuristics*, 7(3):261–304, 2001.

David Salinas and Nick Erickson. Tabrepo: A large scale repository of tabular model evaluations and its automl applications. 2023.

Kenan Šehić, Alexandre Gramfort, Joseph Salmon, and Luigi Nardi. Lassobench: A high-dimensional hyperparameter optimization benchmark suite for lasso. In *International Conference on Automated Machine Learning*, pp. 2–1. PMLR, 2022.

Julien Niklas Siems, Lucas Zimmer, Arber Zela, Jovita Lukasik, Margret Keuper, and Frank Hutter. Nas-bench-301 and the case for surrogate benchmarks for neural architecture search. 2020.

Jasper Snoek, Hugo Larochelle, and Ryan Adams. Practical bayesian optimization of machine learning algorithms. volume 25, 2012.

Ilya M Sobol. Distribution of points in a cube and approximate evaluation of integrals. *USSR Computational mathematics and mathematical physics*, 7:86–112, 1967.

Kevin Swersky, Jasper Snoek, and Ryan Prescott Adams. Freeze-thaw bayesian optimization. *arXiv preprint arXiv:1406.3896*, 2014.

Zhen Wang, Weirui Kuang, Ce Zhang, Bolin Ding, and Yaliang Li. Fedhpo-bench: A benchmark suite for federated hyperparameter optimization. In *International Conference on Machine Learning*, pp. 35908–35948. PMLR, 2023.

Chris Ying, Aaron Klein, Eric Christiansen, Esteban Real, Kevin Murphy, and Frank Hutter. Nas-bench-101: Towards reproducible neural architecture search. In *International conference on machine learning*, pp. 7105–7114. PMLR, 2019.

A Experimental Protocol Details

This appendix provides the study-level settings and formal definitions underlying the compact experimental protocol in Section 4. The definitions operate on the records retained during the original optimizer executions; the resource-aware and distributional analyses therefore require no additional optimizer runs. The seven evaluated methods and their methodological roles follow standard random, quasi-random, model-based, and multi-fidelity HPO formulations (Bergstra & Bengio, 2012; Sobol, 1967; Bergstra et al., 2011; Snoek et al., 2012; Li et al., 2018; 2020; Falkner et al., 2018).

A.1 Study-Level Experimental Settings

Table 3 gives the complete evaluation configuration of the studies used in the paper. "Generated" denotes the number of benchmark instances created by the study configuration, whereas "Eval." denotes the held-out instances on which the reported optimizer comparison is computed. The difference between these quantities results from the benchmark split policy rather than from discarded experimental runs.

Table 3: Experimental settings for the released studies. Budget denotes the nominal number of optimizer evaluation opportunities; fidelity-dependent evaluations may have different computational costs.

Category	Study	Split	Generated	Eval.	Budget	Rep.
Core	Core benchmark	Extrapolation	96	72	44	5
Split	In-distribution	ID	72	4	40	5
	Interpolation	Interpolation	72	36	40	5
	Extrapolation	Extrapolation	72	54	40	5
	Cross-family	Cross-family	72	36	40	5
Stress	Structural search	Interpolation	48	16	36	5
	Fidelity trustworthiness	Extrapolation	48	40	40	5
	Budget realism	Extrapolation	48	29	36	5
	Shift generalization	Cross-family	64	32	40	5
	Protocol robustness	Interpolation	48	24	32	5
	Backend diversity	ID	32	4	28	5
Ablation	Misleading fidelity	Extrapolation	48	19	40	5
Extension	Real-task evaluation	ID	12	3	12	3
	External connector	ID	12	2	12	3

The principal studies retain the same seven optimizers and five independent repetitions, allowing changes in the reported conclusion to be associated primarily with the manipulated benchmark regime rather than with changes in the comparison set. The real-task and connector experiments are intentionally smaller extensions and are therefore used as diagnostic evidence of backend generality rather than as large-scale estimates of optimizer performance.

The nominal evaluation budget in Table 3 should not be interpreted as a common computational cost. A low-fidelity evaluation can be substantially cheaper than a maximum-fidelity evaluation. This distinction motivates the cumulative-cost analyses defined below.

A.2 Execution Records and Final Full-Fidelity Evaluation

For optimizer m on benchmark instance b, one execution produces the trace

$$\mathcal{T}_{m,b} = \{(\mathbf{x}_t, r_t, y_t, \kappa_t)\}_{t=1}^{T_{m,b}}, \tag{10}$$

where $\mathbf{x}_t \in \mathcal{X}_b$ is the queried configuration, r_t is the allocated fidelity or resource level,

$$y_t = f_b(\mathbf{x}_t, r_t), \qquad \kappa_t = c_b(\mathbf{x}_t, r_t), \tag{11}$$

are respectively the observed objective and incurred evaluation cost, and $T_{m,b}$ is the number of completed evaluations.

The cumulative cost after step t is

$$C_{m,b}(t) = \sum_{\tau=1}^{t} \kappa_\tau. \tag{12}$$

The framework additionally stores the best incumbent available after every completed evaluation. Consequently, final-quality, fixed-cost, and anytime analyses are all reconstructed from the same underlying execution record.

After the search budget is exhausted, optimizer m returns configuration $\hat{\mathbf{x}}_{m,b}$. Irrespective of the fidelities used internally during search, the returned configuration is evaluated at the maximum fidelity:

$$F_{m,b} = f_b(\hat{\mathbf{x}}_{m,b}, r_{\max}), \qquad C_{m,b} = C_{m,b}(T_{m,b}). \tag{13}$$

This common full-fidelity evaluation prevents a method from receiving an apparent advantage merely because its final reported observation was obtained at a cheaper or less informative fidelity.

A.3 Instance Calibration and Aggregate Quality

Let $F_{m,b}^{(s)}$ denote the full-fidelity outcome from repetition s and

$$\bar{F}_{m,b} = \frac{1}{K} \sum_{s=1}^{K} F_{m,b}^{(s)} \tag{14}$$

the mean terminal outcome for method m on instance b. The released reporting pipeline defines the empirical calibration anchors

$$F_b^{\min} = \min_{m \in \mathcal{M}} \bar{F}_{m,b}, \qquad F_b^{\max} = \max_{m \in \mathcal{M}} \bar{F}_{m,b}. \tag{15}$$

The normalized regret reported in the main paper is then

$$R_{m,b} = \frac{\bar{F}_{m,b} - F_b^{\min}}{F_b^{\max} - F_b^{\min} + \epsilon}, \tag{16}$$

with lower values indicating better full-fidelity performance. The study-level summary is

$$\bar{R}_m = \frac{1}{|\mathcal{B}_{\text{eval}}|} \sum_{b \in \mathcal{B}_{\text{eval}}} R_{m,b}. \tag{17}$$

The normalization is deliberately instance-specific because the generated objectives need not share a common raw scale. The resulting mean is used only to summarize the explicitly declared evaluation collection. Instance-level distributions and conditional slices are retained whenever the aggregate alone would hide relevant heterogeneity.

A.4 Conditional Conclusion Sensitivity

Let $\rho(m)$ denote the quality rank obtained from $\bar{R}_m$ over a declared study, and let $\rho_k(m)$ denote the corresponding rank within benchmark slice k, such as a scenario, task family, protocol, or backend. We summarize the displacement of the complete ordering as

$$\Delta_k = \frac{1}{|\mathcal{M}|} \sum_{m \in \mathcal{M}} |\rho_k(m) - \rho(m)| . \tag{18}$$

A value $\Delta_k = 0$ means that the complete quality ordering is unchanged within slice k. Larger values indicate that the aggregate ordering is less representative of that slice. Importantly, Δ_k compares rankings formed from the same normalized-regret criterion and therefore does not combine quality and cost.

For the method-specific sensitivity heatmap in Figure 2, we use

$$D_{j,m} = \frac{1}{|\mathcal{K}_j|} \sum_{k \in \mathcal{K}_j} |\rho_{j,k}(m) - \rho_j(m)| , \tag{19}$$

where $\mathcal{K}_j$ is the set of scenario slices in study j. This quantity identifies whether the conditional movement is concentrated in particular optimizers rather than reducing all changes to one study-level number.

Rank displacement is therefore interpreted as a diagnostic of conclusion sensitivity, not as an additional optimization objective. In particular, a low value in one study means that the ordering is internally consistent across that study's slices; it does not imply agreement with the ordering observed in another study.

A.5 Fixed-Cost Evaluation

Evaluation count and computational expenditure are distinct when fidelity levels have different costs. We therefore construct an instance-specific reference budget from the observed terminal costs. Let

$$\bar{C}_{m,b} = \frac{1}{K} \sum_{s=1}^{K} C_{m,b}^{(s)} \tag{20}$$

be the mean terminal cost of method m on instance b. The reference full cost is

$$B_{\mathrm{ref},b} = \max_{m \in \mathcal{M}} \bar{C}_{m,b}. \tag{21}$$

For

$$\alpha \in \{0.25, 0.50, 1.00\}, \tag{22}$$

the fixed-cost evaluation retrieves the best completed incumbent available before cost $\alpha B_{\mathrm{ref},b}$ for every method, repetition, and instance.

More explicitly, let $\hat{\mathbf{x}}_{m,b}^{(s)}(c)$ denote the incumbent available in repetition s immediately after the latest completed evaluation satisfying

$$C_{m,b}^{(s)}(t) \le c. \tag{23}$$

The fixed-cost comparison evaluates

$$\hat{\mathbf{x}}_{m,b}^{(s)} \left(\alpha B_{\mathrm{ref},b} \right)$$

at the corresponding checkpoint and then aggregates over repetitions and benchmark instances.

If a method terminates before a later common checkpoint, its final completed incumbent remains available thereafter. This convention answers the practical question of which solution would be available after spending at most a specified computational budget, rather than penalizing methods simply because they finish early.

A.6 Anytime Evaluation

The fixed checkpoints provide interpretable snapshots of a more general anytime process. For cumulative cost c, define

$$t^\star(c) = \max\{t : C_{m,b}(t) \le c\}. \tag{24}$$

The anytime incumbent is the best configuration retained at step $t^\star(c)$. Evaluating its normalized regret over a common cost grid gives

$$R_m(c) = \frac{1}{|\mathcal{B}_{\text{eval}}|} \sum_{b \in \mathcal{B}_{\text{eval}}} R_{m,b}(c), \tag{25}$$

with averaging additionally performed over repeated runs.

These trajectories are the basis of the cumulative-cost plot in Figure 1. They are particularly important for comparing multi-fidelity and single-fidelity methods because two methods can receive the same nominal number of evaluation opportunities while consuming very different amounts of computational resource.

A.7 Terminal Quality–Cost Pareto Relation

For each optimizer, define its aggregate terminal quality and cost as

$$Q_m = \bar{R}_m, \qquad \bar{C}_m = \frac{1}{|\mathcal{B}_{\text{eval}}|} \sum_{b \in \mathcal{B}_{\text{eval}}} \bar{C}_{m,b}. \tag{26}$$

Method m_i dominates method m_j when

$$Q_{m_i} \le Q_{m_j}, \qquad \bar{C}_{m_i} \le \bar{C}_{m_j}, \tag{27}$$

and at least one inequality is strict. The empirical Pareto set is therefore

$$\mathcal{P} = \{m \in \mathcal{M} : \nexists\, m' \text{ such that } m' \prec m\}. \tag{28}$$

Pareto membership is not itself a ranking. It indicates only that, among the evaluated alternatives, improving one of the two reported quantities requires accepting a deterioration in the other. Selecting among non-dominated methods therefore requires an explicit computational budget or decision preference.

A.8 Repeated-Run Variability

To preserve run-level variation, the same calibration anchors in Eq. equation 15 can be applied to each repetition:

$$R^{(s)}_{m,b} = \frac{F^{(s)}_{m,b} - F^{\min}_b}{F^{\max}_b - F^{\min}_b + \epsilon}. \tag{29}$$

For method m and instance b, the within-instance run-to-run dispersion is

$$\sigma_{m,b} = \mathrm{SD}\left(R^{(1)}_{m,b}, \ldots, R^{(K)}_{m,b}\right). \tag{30}$$

The fundamental repeatability evidence is the collection

$$\Sigma_m = \{\sigma_{m,b} : b \in \mathcal{B}_{\text{eval}}\}, \tag{31}$$

rather than a single standard deviation computed after pooling heterogeneous benchmark instances. We summarize Σ_m using its median and interquartile range and inspect its full empirical distribution in Appendix B.9.

This interpretation separates two properties that need not coincide. A method may have low mean regret but substantial seed-to-seed variation, or it may be highly repeatable while consistently obtaining poor objective values. Repeatability is therefore reported alongside, rather than incorporated into, the final-quality criterion.

A.9 Reporting Hierarchy

The evaluation outputs form a hierarchy rather than a collection of interchangeable scores. The run-level records are the finest observations; these are aggregated first within benchmark instances and then, where appropriate, across a declared evaluation collection. Conditional slices, cumulative-cost trajectories, and per-instance distributions are retained to test whether an aggregate summary hides relevant structure.

Accordingly, the quantities above have distinct roles. Mean normalized regret provides a compact quality summary; rank displacement identifies conditional changes in the same quality criterion; fixed-cost and anytime analyses expose resource dependence; Pareto analysis represents terminal quality–cost trade-offs without scalarization; and Σ_m characterizes repeatability across benchmark instances. None of these quantities is intended to replace the others with a single universal optimizer score.

This reporting hierarchy follows the broader principle from empirical algorithmics that aggregate performance summaries should be interpreted together with distributions, trajectories, and conditions of success (Hooker, 1995; Barr et al., 1995; Rardin & Uzsoy, 2001; Bartz-Beielstein et al., 2020). In Beyond Final Regret, these established evaluation principles are operationalized through the common HPO generation and trace-retention pipeline defined in Sections 3 and 4.

B Additional Experimental Results

This appendix provides the detailed numerical evidence underlying the conclusions summarized in Section 5. The purpose is not to introduce additional claims, but to expose the study composition, instance-level behavior, cost-conditioned comparisons, split- and scenario-specific results, fidelity ablations, and repeated-run variation that are necessarily compressed in the main paper. Throughout, aggregate summaries are interpreted together with their corresponding conditional or distributional evidence rather than as context-independent optimizer rankings.

B.1 Benchmark Split Inventory

Table 4 reports the train/validation/test composition of the released study families. These partitions describe benchmark-instance construction and evaluation; they are not training splits for the HPO optimizers themselves. The main empirical analysis is organized into three layers: a core benchmark that provides the broad aggregate comparison, four split-family studies that test transfer across benchmark distributions, and targeted stress studies that perturb individual properties of the evaluation regime. The real-task and external-connector experiments are smaller backend-oriented extensions intended primarily to verify that the same evaluation interface can be applied beyond the controlled surrogate setting.

Table 4: Released study families and benchmark split sizes.

Category	Study	Split	Train	Val.	Test
Core	Core benchmark	Extrapolation	0	24	72
Split	In-distribution	ID	64	4	4
	Interpolation	Interpolation	36	0	36
	Extrapolation	Extrapolation	0	18	54
	Cross-family	Cross-family	18	18	36
Stress	Structural search	Interpolation	32	0	16
	Fidelity trustworthiness	Extrapolation	0	8	40
	Budget realism	Extrapolation	0	19	29
	Shift generalization	Cross-family	21	11	32
	Protocol robustness	Interpolation	12	12	24
	Backend diversity	ID	24	4	4
	Misleading fidelity	Extrapolation	10	19	19
Extension	Real-task evaluation	ID	6	3	3
	External connector	ID	8	2	2

The different evaluation-set sizes are a consequence of the split construction rather than an attempt to equalize statistical power across studies. For example, extrapolation and fidelity-trustworthiness

studies reserve comparatively large held-out sets because their purpose is to evaluate behavior away from the represented construction regime, whereas the backend-diversity study contains only four held-out instances because it is intended as a controlled diagnostic probe. Results from the latter are therefore interpreted more cautiously throughout the paper.

The distinction between benchmark "train", "validation", and "test" partitions should also be emphasized. These are partitions of generated benchmark instances; they are not training sets used to fit the HPO algorithms themselves. Their purpose is to define which benchmark conditions are represented during study construction and which are withheld for evaluation.

B.2 Core Benchmark: Aggregate and Instance-Level Results

Table 5 reports the complete comparison on the 72 held-out instances of the core benchmark. The table contains the two quantities that motivate the resource-aware analysis in the main paper: final normalized regret and cumulative evaluation cost.

Table 5: Core benchmark over 72 held-out instances. "Pareto" denotes membership in the empirical terminal quality–cost frontier.

Method	Mean norm. regret	Mean cost	Pareto
Surrogate BO	0.0886	893418	yes
TPE	0.1489	874503	yes
Sobol	0.6639	938844	no
Random	0.7564	946441	no
BOHB	0.8440	91956	yes
Hyperband	0.9082	92369	no
ASHA	0.9126	87572	yes

Two distinct operating regimes are evident. Surrogate BO and TPE achieve substantially lower final regret, but they do so at a mean cost close to 9×10^5. The multi-fidelity methods instead operate near 9×10^4 cost units, roughly an order of magnitude lower, while accepting markedly worse full-budget quality. This separation explains why ranking methods only by final regret or only by computational cost gives incomplete information.

The Pareto column makes this distinction explicit. ASHA is non-dominated because it occupies the minimum-cost extreme even though its terminal regret is high. BOHB achieves better quality at a slightly higher cost, while TPE and surrogate BO occupy progressively lower-regret regions at substantially larger computational expenditure. Thus, Pareto membership should not be interpreted as evidence that these methods are equally desirable; it means that no other reported method is simultaneously no worse in both dimensions.

B.3 Instance-Level Distribution of Core Performance

Aggregate means can conceal whether one method consistently outperforms another or whether the average is produced by a heterogeneous collection of instance-level outcomes. We therefore examine the paired core-instance results for the two strongest final-quality methods.

Surrogate BO has lower normalized regret than TPE on 43 of the 72 held-out instances, while TPE is lower on the remaining 29. Defining

$$d_b = R_{\mathrm{TPE},b} - R_{\mathrm{BO},b},$$

the observed mean paired difference is

$$\bar{d} = 0.0603.$$

A nonparametric bootstrap over benchmark instances gives a descriptive 95% interval of approximately

$$[0.002,\ 0.119].$$

The interval is reported to characterize uncertainty across the particular benchmark instances in the study, rather than to make a population-wide significance claim about the two optimizer families.

In particular, the $43/72$ comparison shows that the aggregate advantage of surrogate BO is not equivalent to instance-wise dominance: TPE remains preferable on a substantial fraction of the held-out collection.

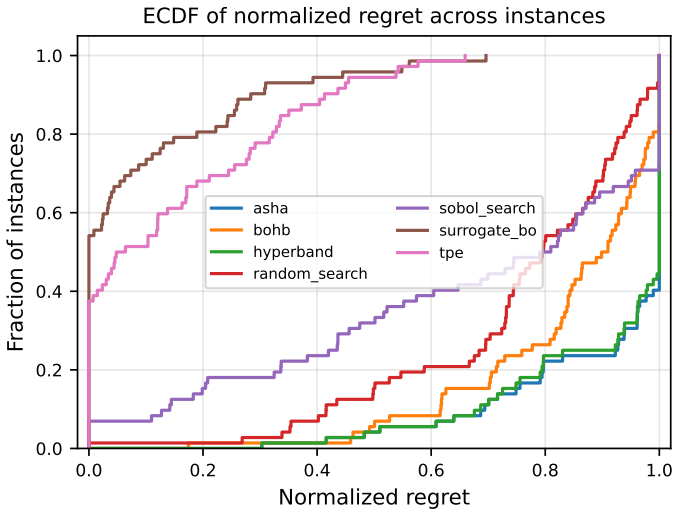


Figure 3: ECDF of instance-normalized regret over the core benchmark. The distributional view complements the aggregate values in Table 5. Curves shifted toward lower regret indicate that a larger fraction of benchmark instances is solved with comparatively small normalized regret.

Figure 3 makes the same point distributionally. Rather than reducing the benchmark to one mean value per optimizer, the ECDF shows the fraction of instances lying below any given regret threshold. The separation between methods at low regret supports the strong aggregate performance of model-based search, while overlap between the curves illustrates that the relative advantage is not uniform over all instances.

B.4 Fixed-Cost and Anytime Analysis

The resource-aware analysis uses the complete optimization traces rather than the scalarized regret–cost score used in the original benchmark report. For benchmark instance b, the reference resource level is defined as

$$B_{\mathrm{ref},b} = \max_{m \in \mathcal{M}} \bar{C}_{m,b},$$

where $\bar{C}_{m,b}$ is the mean terminal cost of method m on that instance across repeated runs.

For each budget fraction

$$\alpha \in \{0.25, 0.50, 1.00\},$$

we recover the best incumbent completed by each method before $\alpha B_{\mathrm{ref},b}$. This creates a common cost constraint while allowing methods to differ in the number and fidelity of evaluations completed within that constraint.

Table 6: Conclusion at common cumulative-cost checkpoints.

Cost checkpoint	Lowest-regret method	Next relevant method
25%	BOHB	Hyperband / ASHA operate in the same low-cost regime
50%	Surrogate BO	TPE
100%	Surrogate BO	TPE

Table 6 reveals a qualitative crossover that is hidden by terminal rankings. BOHB is favored at the tightest cost checkpoint because the low cost of partial-fidelity evaluations allows it to make substantial progress before expensive model-based methods have consumed their full search budget. At the

50% checkpoint, surrogate BO becomes the lowest-regret method and TPE becomes competitive. The same model-based regime remains preferable at the full reference cost.

The important observation is therefore not simply that different methods occupy different ranks. Rather, the evidence supports different conclusions under different resource constraints. This is why the main paper treats the result as a budget-indexed conclusion instead of constructing a single cost-adjusted total ordering.

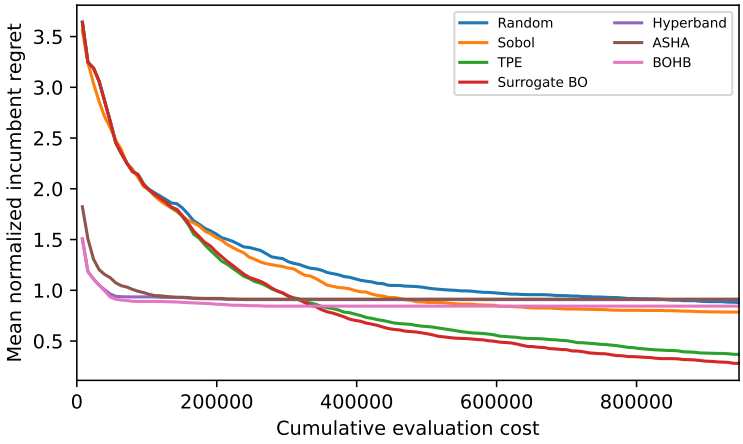


Figure 4: Anytime normalized regret versus cumulative evaluation cost for the core benchmark. Each curve represents the mean incumbent quality available after spending a given amount of cumulative evaluation cost.

Figure 4 provides the continuous version of the checkpoint analysis. The horizontal axis measures actual incurred evaluation cost rather than the number of trials, which is important because low- and high-fidelity evaluations have substantially different costs. Multi-fidelity methods therefore progress through many inexpensive evaluations early in the trajectory, whereas model-based methods eventually reach lower-regret regions after substantially more computation becomes available. The fixed checkpoints in Table 6 should be read as interpretable snapshots of this more complete anytime behavior.

B.5 Split-Family Results

Table 7 examines whether conclusions obtained from the aggregate benchmark persist under different ways of partitioning the benchmark distribution. The first three columns describe the aggregate outcome under each split. The final two columns show whether that aggregate conclusion remains unchanged inside the individual task-family slices.

Table 7: Aggregate and conditional conclusions across split families.

Split	**Aggregate leader**	**Regret**	**Conditional change**	$\Delta_{\max}$
ID	Surrogate BO	0.0749	tabular/core-MLP slice favors TPE	1.143
Interpolation	Surrogate BO	0.1096	text/core-Transformer slice favors TPE	0.571
Extrapolation	Surrogate BO	0.0934	text/core-Transformer slice favors TPE	0.286
Cross-family	Surrogate BO	0.1308	text/core-Transformer slice favors TPE	0.286

At the aggregate level, the conclusion is notably persistent: surrogate BO has the lowest normalized regret in all four split families. This is an important negative result for the benchmark's central hypothesis because it demonstrates that controlled perturbation does not automatically manufacture a different leading method.

The conditional columns, however, show that aggregate persistence and local persistence are different properties. TPE becomes preferable on the core-MLP slice of the in-distribution study and

on the core-Transformer slice under interpolation, extrapolation, and cross-family evaluation. The largest displacement occurs in the in-distribution study, where $\Delta_{\max} = 1.143$.

Consequently, two statements can simultaneously be true: the aggregate final-quality conclusion is robust to the split family, while the detailed ordering remains task-family dependent. This distinction motivates reporting both the aggregate result and the corresponding conditional slices.

B.6 Detailed Stress-Test Results

The targeted studies isolate individual properties of the evaluation regime while preserving the common optimizer and reporting interfaces. Table 8 reports the aggregate lowest-regret method, its aggregate regret, a representative scenario-level change, and two measures of rank displacement. Here $\Delta_{\max}$ denotes the largest mean absolute displacement from the study-level ordering among the reported slices, while $d_{\max}$ is the largest movement of any individual optimizer.

Table 8: Detailed conclusion sensitivity across targeted stress studies.

Stress	Aggregate leader	Regret	Representative conditional result	$\Delta_{\max}$	$d_{\max}$
Structure	TPE	0.0554	continuous-SVM slice favors Surrogate BO	0.857	3
Fidelity	Surrogate BO	0.0808	heterogeneous-CNN slice favors TPE	0.857	3
Budget realism	Surrogate BO	0.1314	Transformer cost-spike slice favors TPE	0.571	2
Shift	Surrogate BO	0.1413	cross-family CNN and extrapolated text favor TPE	0.286	1
Protocol	Surrogate BO	0.0797	leading method persists; lower ordering changes	0.571	2
Backend	Surrogate BO	0.0009	HPOBench connector favors BOHB; real-task SVM favors Sobol	2.286	6

The structural stress study provides the clearest aggregate change relative to the main core result. TPE obtains the lowest overall regret, yet the continuous-SVM slice favors surrogate BO. This indicates that the structural composition of the search problem can influence not only lower leaderboard positions but also the leading quality conclusion.

The fidelity and budget-realism studies display a different pattern. Their aggregate conclusions continue to favor surrogate BO, but specific fidelity or cost regimes favor TPE. These cases demonstrate why the benchmark treats fidelity reliability and cost behavior as explicit experimental variables: changing them can alter the local evidence even when the aggregate conclusion remains unchanged.

The protocol study provides an instructive counterexample. Surrogate BO remains the leading method, while lower-ranked methods move enough to produce $\Delta_{\max} = 0.571$. Hence, "same winner" and "same ranking" should not be treated as equivalent notions of robustness.

Backend diversity generates by far the largest movement. Its $\Delta_{\max} = 2.286$ is substantially larger than the values observed for the other targeted perturbations, and one method moves as many as six positions. This supports treating backend realization as part of the experimental condition rather than assuming that a surrogate, native task, and external connector are interchangeable representations of the same optimization problem.

The backend result should nevertheless be interpreted cautiously. This study contains only four held-out instances, so its role is diagnostic: it demonstrates that substantial sensitivity is possible under backend replacement, but does not estimate how frequently such changes occur across the broader population of HPO tasks.

B.7 Interpreting the Conclusion-Sensitivity Heatmap

Figure 2 in the main paper expands the aggregate quantities above into method-specific sensitivity. For study j and optimizer m, each heatmap cell represents the average absolute displacement of that optimizer's scenario-specific quality rank from its aggregate rank within the same study.

The heatmap therefore answers a different question from Table 8. The table asks how much the *overall ordering* can move within a study; the heatmap shows *which optimizers* are responsible for that movement.

Several patterns are visible. Backend variation affects nearly every optimizer and produces the largest overall displacements. Structural stress particularly changes the position of TPE, whereas extrapolation and cross-family studies leave several methods almost unchanged while altering the

relative positions of the strongest model-based methods. In the in-distribution study, Sobol and ASHA move more strongly than surrogate BO. Thus, conclusion sensitivity is not uniformly distributed across optimizers.

A zero-valued heatmap cell should not be interpreted as agreement with the core benchmark. It means only that the method's rank is stable across the scenario slices *within that study*. The misleading-fidelity ablation illustrates this distinction: its within-study ordering is highly consistent, producing near-zero displacement, even though the aggregate ordering differs substantially from the main core benchmark.

B.8 Misleading-Fidelity Ablation

The misleading-fidelity experiment isolates a failure mode that cannot be diagnosed solely from terminal cost. Here, low-resource evaluations are deliberately constructed to provide poor guidance about full-fidelity performance.

Table 9: Results under deliberately misleading low-fidelity information.

Method	Mean normalized regret	Mean total cost
TPE	**0.1040**	640149
Surrogate BO	0.1719	692856
Sobol	0.6144	764458
Random	0.8135	804350
Hyperband	0.9804	70085
ASHA	0.9804	**70036**
BOHB	0.9804	70085

Table 9 shows a particularly clear separation between cheap search and informative search. Hyperband, ASHA, and BOHB remain approximately an order of magnitude cheaper than the model-based methods, demonstrating that their resource-allocation mechanisms continue to reduce computational expenditure. Nevertheless, all three obtain normalized regret close to 0.98, whereas TPE and surrogate BO obtain substantially lower values.

The result therefore cannot be summarized as "multi-fidelity methods are inefficient." They are computationally efficient in this regime. The failure occurs because the inexpensive observations used to guide resource allocation are no longer sufficiently informative about the full-fidelity objective. This is precisely the type of mechanism-level condition that would be hidden by reporting only terminal cost or only terminal regret.

The ablation also helps distinguish two questions that are sometimes conflated: whether a method can save computational resources and whether those savings preserve solution quality. Under reliable fidelity signals these goals can align; under misleading fidelity they need not.

B.9 Repeated-Run Variability

The main studies contain five independent repetitions per method. The original reporting artifact computes run-to-run standard deviations separately for each method–instance pair before averaging them. Because that average can obscure whether variability is broadly distributed or concentrated on a subset of instances, the revised analysis retains the per-instance quantity

$$\sigma_{m,b} = \mathrm{SD}\left(R_{m,b}^{(1)}, \ldots, R_{m,b}^{(K)}\right)$$

as the fundamental observation.

Figure 5 shows that repeated-run variability is not well represented by a single global "stability score." Some methods exhibit narrow dispersion on many instances and much larger variation on others. This heterogeneity is important because the same average standard deviation could arise from moderate variation on every task or from severe instability on a relatively small subset of tasks.

Sobol provides an especially useful reference case. Because the implementation uses a fixed low-discrepancy sequence, many method–instance combinations are effectively deterministic, producing

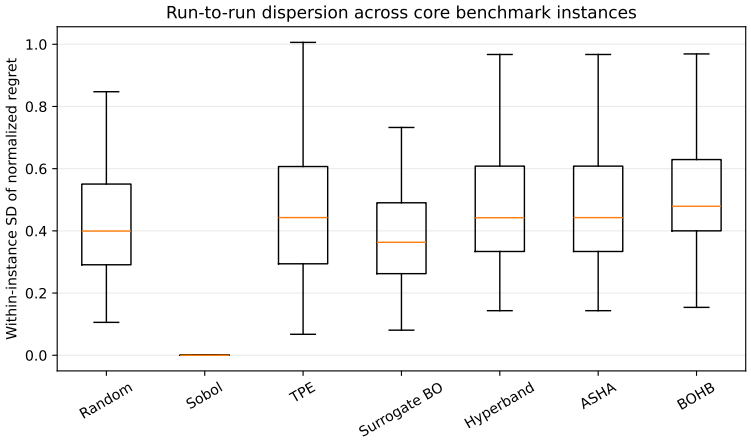


Figure 5: Distribution of within-instance run-to-run normalized-regret dispersion across the core benchmark. Each observation is computed from repeated runs on one fixed benchmark instance before any aggregation across instances.

near-zero within-instance dispersion. Stochastic model-based and multi-fidelity methods exhibit broader distributions because changes in random seed can affect model fitting, candidate selection, and resource allocation.

The interpretation is therefore deliberately descriptive. Lower dispersion means that repeated runs on the same benchmark instance produce more similar normalized-regret outcomes; it does not imply that the method has better optimization quality. Repeatability and final performance are separate properties and should be reported as such.

B.10 Backend-Oriented Extension Studies

The benchmark also contains two small extension studies that exercise the common optimizer interface outside the principal controlled surrogate experiments.

Table 10: Backend-oriented extension studies.

Study	Methods	Lowest-regret method	Mean regret
Real-task extension	3	TPE	0.3333
External connector extension	3	Sobol	0.1316

In the real-task extension, TPE has the lowest aggregate regret among the three evaluated methods, whereas Sobol has the lowest regret in the external-connector experiment. The fact that the leading method differs across these small studies is consistent with the broader observation that backend realization can influence an empirical conclusion.

However, these studies contain only two or three held-out instances and evaluate only random search, Sobol, and TPE. They should therefore not be used to claim that TPE is generally preferable on native real tasks or that Sobol is generally preferable on connector-backed benchmarks. Their primary value is methodological: they demonstrate that the same configuration, execution, audit, and reporting machinery can be applied to objective realizations that differ from the principal surrogate backend.

B.11 Summary of Supplementary Evidence

The additional results support four distinctions that are central to the benchmark design.

First, *aggregate advantage is not instance-wise dominance*. The core benchmark favors surrogate BO in mean final quality, while the paired instance results show substantial overlap with TPE.

Second, *quality and cost describe different aspects of performance*. The Pareto and anytime analyses reveal low-cost multi-fidelity and low-regret model-based operating regimes that cannot be faithfully reduced to one universal scalar ordering.

Third, *aggregate persistence and conditional persistence are different*. Surrogate BO remains the aggregate leader across all four principal split families, while particular task-family and stress slices favor different methods or substantially reorder the remaining methods.

Finally, *repeatability is an instance-level property before it is an aggregate property*. Reporting distributions of within-instance dispersion preserves information that is lost when all variation is compressed into one stability statistic.

Taken together, these supplementary analyzes reinforce the intended interpretation of BEYOND FINAL REGRET: aggregate summaries remain useful for navigating a benchmark, but the scientific conclusion should be accompanied by evidence about the conditions, budgets, instances, and backend regimes under which that conclusion continues to hold.