Neither Synthetic Benchmarks nor Common-Sense Reasoning Predicts Real-World Derivative-Free Optimizer Performance

Peter Cotton

June 18, 2026

Abstract

Practitioners choose a derivative-free optimizer either by consulting a benchmark leaderboard or by reasoning about the problem. We test both. Using a memorisation-proof suite of real-world objectives, each relocated by a seeded cube-to-cube diffeomorphism, we rank a panel of 22 optimizers and ask how well that ranking is predicted by the same panel's ranking on synthetic analytic benchmark functions, and by a language model asked to reason, problem by problem, about which optimizer should win. Both predictors are essentially uninformative: the rank correlation with real-world performance is statistically indistinguishable from zero in either case. A language-model selector that picks an optimizer per problem in fact does worse than a constant policy of always using one robust direct-search method. The two failures share a mechanism. Both the benchmark and the model over-trust model-based trust-region methods (Powell's NEWUOA and BOBYQA), which dominate smooth analytic functions yet rank near the bottom on the real suite. This is a draft, and several confirmatory runs are still in progress (Section 6).

1 The question

Derivative-free optimization (DFO) has dozens of credible algorithms and no universal winner. The two ways a practitioner actually picks one are to read a benchmark, such as a global-optimization test-function leaderboard, or to reason about the problem ("it is smooth and low-dimensional, so use a quadratic-model method"). We ask whether either route predicts what actually works on real problems.

The difficulty in answering honestly is that public benchmarks are mostly synthetic analytic functions such as sphere, Rosenbrock, and Rastrigin, and it is unknown whether a leaderboard on those transfers to messy real objectives. We therefore build a real-world instrument and treat the synthetic leaderboard and the model's reasoning as two competing predictors of it.

2 Instruments

A test suite motivated by real problems. Standard global-optimization benchmarks are analytic test functions, chosen for known optima and tunable difficulty rather than for resembling the problems practitioners face. We instead use `example_applications`, a collection of 52 distinct pure-Python objectives. Each is a reduced-order but faithful model of a genuine engineering, scientific, or operational task: placing wind turbines to maximise output under wake interaction, dispatching a grid battery against a price curve, throttling a booster to a soft landing, arranging atoms to minimise the Lennard-Jones potential, sizing a pressure vessel or cantilever, or tuning

a PID controller, among others.[1] Four of these expose a faithful size knob (number of turbines, dispatch horizon, atom count, control segments) that we scale to add 14 higher-dimensional instances, so the full instance set spans 2 to 100 dimensions, 66 instances in all. These carry the structure, coupling, ruggedness, and scale heterogeneity of real objectives, which is what analytic benchmarks omit and what we are testing. All objectives live on $[0,1]^n$, and lower is better.

Made memorisation-proof. Reusing a fixed set of named problems risks an optimizer, or its tuner, implicitly learning where the optima are. We close this by wrapping each objective in a seeded cube-to-cube diffeomorphism ϕ, so the optimizer minimises $f(\phi^{-1}(x))$ on $[0,1]^n$. This relocates the optimum to an instance-specific point while exactly preserving the landscape's difficulty, conditioning, multimodality, and critical-point structure; the value at each point is merely moved, not changed. Different seeds give different but equally hard instances of the same problem, so nothing is won by memorising a location, only by searching.

The panel and the empirical ranking. We score 22 optimizers, namely the 21 pure-Python `humpday` ports (Nelder-Mead, Powell, three PRIMA trust-region methods, CMA-ES, Differential Evolution, Particle Swarm, Pattern Search, Coordinate Descent, Simulated Annealing, and others) together with production `pycma`, on each disguised instance at fixed evaluation budgets. Per instance we rank the panel, and averaging gives a real-world leaderboard at each budget.

Predictor 1: synthetic benchmarks. The same panel is scored on rotated synthetic test functions (sphere, ellipsoid, cigar, Rastrigin, Rosenbrock) from the `nevergrad` library, at dimensions 5, 20, and 40, giving a synthetic leaderboard.

Predictor 2: model reasoning. For each real problem we give a language model only its natural-language description, its dimension, and the evaluation budget, together with textbook one-line descriptions of the candidate optimizers, and no benchmark results. It reasons about the likely landscape and ranks the candidates. We repeat with shuffled presentation order and aggregate the rankings with a Plackett-Luce (Bradley-Terry) model fit by Hunter's MM algorithm; only the induced order is used, so cross-call score-scale arbitrariness never enters.

3 Result 1: synthetic benchmarks do not predict the real ranking

Table 1 reports Kendall's τ and Spearman's ρ between the synthetic and real leaderboards, for the budgets completed so far. The correlation is weak and not statistically significant.

The tails are where it bites. The synthetic top performers are the model-based methods (NEWUOA, BOBYQA, UOBYQA, pycma, Powell), whereas the real-world top performers are humble direct-search and population methods such as Coordinate Descent, Pattern Search, Harmony Search, Ant Colony, and CMA-ES. NEWUOA ranks near the top of the panel on synthetic functions, around second, but near the bottom on real problems, around eleventh. The single best optimizer by synthetic benchmarks is one of the worst on real problems.

[1] Each problem has an interactive, visual demonstration at `https://humpday.microprediction.org/applications/`.

budget	Kendall τ	p	Spearman ρ
60	+0.15	0.32	+0.21
120	+0.21	0.17	+0.27

Table 1: Synthetic-versus-real leaderboard correlation (synthetic $n = 45$ instances, real $n = 90$). Both coefficients are indistinguishable from zero. These figures are preliminary, from an earlier panel that included a Bayesian-optimization baseline since removed for being intractable in high dimensions; they are being recomputed on the current panel, and the budget-240 row is still in progress.

4 Result 2: model reasoning does no better, and is worse than a constant

We elicited per-problem rankings for 18 real problems at budget 120. The predictive correlation with the real ranking (Table 2, left) is again about zero: the model, at $\tau = +0.07$, is indistinguishable from random at $+0.03$ and only marginally above synthetic at -0.02.

predictiveness (τ vs real)		selection: mean real rank of top pick	
predictor	τ	policy	mean rank
model reasoning	+0.07	oracle (best per problem)	1.0
synthetic	−0.02	always Pattern Search	3.78
random	+0.03	model's top pick per problem	5.72
		synthetic favourite (PRIMA)	6.39

Table 2: Left: the model predicts the real ranking no better than chance. Right: choosing the model's top pick per problem (lower is better, over a 10-candidate field) is worse than committing to one robust method.

The selection test (Table 2, right) is the sharper finding. Picking the model's recommended optimizer per problem gives a mean real rank of 5.72, worse than always using a single robust method (Pattern Search, at 3.78). The model's problem-by-problem reasoning subtracts value relative to a constant policy.

The mechanism is explicit in the picks: 15 of the 18 top recommendations were PRIMA, mostly BOBYQA. The model reliably reasons that a smooth-looking engineering objective calls for a quadratic trust-region method, the very methods Result 1 shows crater on the real suite, and it switches to CMA-ES or Differential Evolution only when the description overtly signals ruggedness, as in the financial and game-playing problems. The model reasons its way into the same trap the synthetic benchmark is built on.

5 The shared mechanism

Both failures have one cause. Smooth analytic functions reward an aggressively exploited curvature model, so both the benchmark and textbook reasoning crown model-based trust-region methods. Real objectives are rugged, coupled, noisy, or multi-scale, and at low budget they reward methods that hedge rather than commit to a model. A separate probe supports this directly: a covariance-damping intervention that reins in an over-trusted, undersampled covariance helps

on the structured real suite but is neutral to harmful on smooth synthetic functions. It pays exactly when the model should not be trusted.

6 Caveats and status

This is a living draft, and its numbers come with several qualifications. With a panel of around two dozen optimizers and modest instance counts, none of the reported τ values is statistically significant; the honest claim is that synthetic benchmarks and model reasoning carry no reliable signal about real-world performance, not that they are perfectly anti-correlated. Several confirmatory runs are also still computing and will harden or qualify these figures, including the budget-240 rank correlation, a multi-budget held-out optimizer comparison, a runtime optimizer-crossover study, and a train/test generalisation run.

Two further qualifications concern the baselines and the scope. The always-Pattern-Search policy was the best fixed method on these problems, which gives it a mild post-hoc advantage, though it is consistent with Pattern Search being a genuine top performer in Result 1. And the model results rest on a single model, a single budget, and three elicitation rounds, so they bound the effect rather than settle it.

7 Implication

If it holds, the practical message is uncomfortable and useful. One should not select a DFO algorithm from a synthetic leaderboard, and should not trust a language model to reason one out either, because both systematically over-recommend model-based methods that fail on real problems. Until selection is grounded in real, or at least memorisation-proof real-world-like, evaluation, committing to one robust and model-light optimizer beats both. The broader caution is that a benchmark and a language model trained on the same literature can share a blind spot, so agreement between them is not evidence of correctness.