Discovering a Derivative-Free Optimizer by Evolution Against a Memorisation-Proof Real-World Benchmark

Peter Cotton

June 18, 2026

Abstract

We *discover* rather than hand-design a derivative-free optimizer by evolving a parameterised DE/ES-hybrid template against a memorisation-proof suite of real-world objectives. A genetic algorithm searching 14 behavioural knobs reaches a normalised regret of 0.110 versus a Nelder–Mead/Differential-Evolution/CMA-ES panel (mean rank 1.47 of 4), against 0.191 for the original 12-knob template. The gain comes from one added mechanism — a cheap, R^2-gated separable-quadratic trust-region jump — which we validate three ways: an A/B re-evolution, a single-gene toggle ablation (disabling it more than doubles regret), and independent re-discovery by the production search, which drives the jump's firing probability to $\approx$1. The discovered optimizer is a hedged hybrid, found by grounding fitness in *real* (disguised) evaluation rather than a synthetic proxy.

1 The question

Can a competitive derivative-free optimizer be *discovered* by search, rather than hand-designed — and discovered against a benchmark it cannot game? We parameterise a single unified DE/ES-hybrid template and let a genetic algorithm tune it, scoring every candidate by normalised regret against a fixed baseline panel (Nelder–Mead, Differential Evolution, CMA-ES) on *disguised* real-world objectives.

2 The space of optimizers (the genome)

A genome is a point in $[0,1]^{14}$ decoding to a concrete optimizer that blends, in tunable proportion: DE moves (`rand/1` and current-to-best, mixed), SHADE-style self-adaptation of F/CR, local moves (Gaussian or coordinate pattern search with a 1/5-success step), simulated-annealing acceptance, restart-on- stagnation, and — the two genes added here — a **separable-quadratic trust-region jump**: with probability `p_surrogate` a generation fits a diagonal quadratic $f(x) \approx c + \sum_i b_i x_i + \sum_i a_i x_i^2$ to the nearest $\sim(4n{+}2)$ evaluated points and jumps to its per-coordinate vertex $x_i^\star = -b_i/(2a_i)$ (only where $a_i > 0$), within an adaptive trust radius, gated by a minimum model R^2 (`r2_min`). The fit is $O(n)$ — a poor-man's NEWUOA step the pure DE/ES moves lack.

3 The discovery loop

A (μ+λ) genetic algorithm with gene-wise crossover, Gaussian mutation, elitist survival, and a warm-started, diverse initial population searches the genome space. Fitness is *continuous* normalised regret (not integer rank) so the search gets a gradient even when the candidate is merely

close behind the panel. The benchmark is the `example_applications` suite, each objective relocated by a seeded cube→cube diffeomorphism so the optimum cannot be memorised.

4 Result: the surrogate earns its place

optimizer	normalised regret	mean rank (of 4)
base template GA (12 knobs)	0.191	1.73
surrogate prototype (A/B re-evolution)	0.112	—
canonical 14-knob GA	**0.110**	**1.47**

Table 1: Same scale throughout (15 disguised demos, 2 seeds, 100 trials). Lower regret is better; mean rank is among {candidate, NM, DE, CMA}.

Adding the surrogate lowers regret from 0.191 to 0.110 (Table 1). Three independent checks attribute the gain to the mechanism itself, not to extra search:

1. **Single-gene ablation.** On the evolved genome, toggling only `p_surrogate`: at 0.625 (evolved) regret is 0.112; at 0.0 (jump off) it jumps to 0.238; at 1.0 (always) it is 0.129. Disabling the jump *more than doubles* regret — and leaves the genome *worse than the base template* (0.238 vs 0.191), showing the base knobs *co-adapted* to rely on the jump rather than treating it as a bolt-on.

2. **Independent re-discovery.** The production search, run fresh on the 14-knob genome, reaches 0.110 and drives `p_surrogate` to ≈ 0.998 — given free rein, evolution makes the jump fire almost every generation. (It found a different co-adapted basin than the prototype's 0.625; both score ≈ 0.11, so the "optimal" firing rate is not a constant — it co-adapts with the base knobs.)

3. **Behaviour preserved.** The mechanism was folded into the shipped template additively; the original API and its tests are unchanged.

5 Caveats

- **In-sample.** Fitness is measured on the same 15 demos and 2 seeds the GA trains on, so the winner could overfit those instances. A train/test split (evolve on demo-set A, score on disjoint set B and unseen seeds) is running; generalisation is not yet established.

- **Regime-specific.** This is low-budget DFO on *real-world* objectives. The same curvature-exploiting idea is neutral-to-harmful on smooth *synthetic* functions (where production CMA dominates), so the surrogate's value is conditional on the messy, structured, low-budget regime, not universal.

- **Honest framing.** What evolution yields is a strong tuned configuration of an existing hybrid template plus one genuinely new mechanism — a new optimizer *instance*, not a new algorithm *family*.