# Thinking Outside the Simplex

Signed compositions of optimization algorithms,
with negative weights as anti-inspirations

Peter Cotton

July 6, 2026

**Abstract**

A companion note [1] blends established optimizers by placing them at the vertices of a simplex: a recipe is a vector of non-negative mixing weights, realized as a working program by a language model. Here the weights are allowed to go negative. A recipe such as $-100\%$ algorithm A, $+200\%$ algorithm B means: build B, maintain a cheap shadow of A's proposal logic that spends no evaluations, and steer away from where A would go. On the selection suite the best signed recipe beat every unsigned draw (regret 0.179 against 0.326), and the generated program faithfully implements the avoidance semantics. Then an ablation assigns the credit: disabling the repulsion in that same program improves it, to 0.130 on identical instances. The win belonged to the host architecture the recipe happened to elicit, and the signed term subtracted value. This note records the construction, the trap, and the protocol that avoids it: no signed recipe counts until it beats its own ablated twin.

## 1 The construction

A simplex recipe over $K$ algorithms has weights $w_k \geq 0$ summing to 1. Drop the non-negativity and keep the sum: the recipes now range over an affine slice of which the simplex is the all-positive face.

The semantics of a negative weight is avoidance. The positive part of the recipe is realized exactly as before [1]: the heaviest vertex hosts the architecture and the rest graft their ideas in proportion.

Each negatively weighted vertex becomes an anti-inspiration. The generated program must maintain a shadow of that algorithm's proposal logic, computing where it would move next without spending any objective evaluations, and must penalise or resample candidates near the shadow's suggestion, with repulsion strength and radius scaling with the magnitude of the weight.

Figure 1 shows the actual prompt at the winning recipe. As with the unsigned construction, compliance is checkable by reading the output: the winning program contains a shadow Differential Evolution population evolved at zero evaluation cost, a repulsion radius proportional to the anti-strength, and penalty and push-away logic wired into every evaluation.

## 2 Results

Twenty-two recipes were generated and scored on the eight-problem selection suite of the companion note, at 100 evaluations, two seeds, with panel-normalised regret: twelve random signed

```
Build a black-box numerical optimizer from a SIGNED recipe of base
algorithms. Positive weights are inspirations to blend, exactly as in
asymmetric blending: the highest-weight method is the HOST architecture
and the others donate grafted ideas in proportion to their weight.

NEGATIVE weights are ANTI-INSPIRATIONS. For each negatively-weighted
method, maintain a cheap SHADOW of its proposal logic (spend NO objective
evaluations on it; just compute where it WOULD move next from the current
state) and steer AWAY from that suggestion: penalise or resample
candidates that fall within a repulsion radius of the shadow's
suggestion. Repulsion strength and radius scale with the magnitude of the
negative weight. The shadow must be real, working code, not a comment.

Signed recipe: NelderMead +200%, DifferentialEvolution -100%

HOST architecture (100% of positive mass): NelderMead - downhill simplex:
reflect/expand/contract/shrink a set of n+1 points
ANTI-INSPIRATION (strength 100%): DifferentialEvolution - shadow its
logic (population + difference-vector mutation and crossover) and avoid
where it would sample.

[followed by the fixed interface contract of the unsigned construction]
```

Figure 1: The prompt at the winning signed recipe, +200% Nelder–Mead, −100% Differential Evolution.

recipes, four hand-picked ones of the −1A+2B form, and six unsigned random controls.

Table 1 shows the head of the leaderboard. The three best recipes are all signed, and the winner is the canonical form itself: Nelder–Mead hosting, Differential Evolution repelled, at 0.179, the best single generation observed on this suite by any method in the companion study. Taken at face value this is a victory for the extension. It should not be taken at face value.

| recipe | signed | regret |
|---|---|---|
| +200% NM, −100% DE | yes | 0.179 |
| random signed 2 | yes | 0.262 |
| random signed 6 | yes | 0.306 |
| best unsigned control | no | 0.326 |
| +150% SA, −50% PS | yes | 0.341 |

Table 1: Selection leaderboard, best five of twenty-two generations. Five of the sixteen signed generations produced degenerate programs scoring 1.0; the anti-inspiration machinery is harder to write than a blend.

The ablation is the decisive step, and it reverses the story. Setting the program's anti-strength to zero, which empties the repulsion radius and makes the shadow inert, and rescoring on the identical deterministic instances, improves the regret from 0.179 to 0.130. The repulsion was a handicap. The recipe won because its prompt elicited an unusually good Nelder–Mead-style host, and the signed term this note exists to test made that host worse.

The out-of-sample evidence reads consistently once the credit is assigned. On the fourteen-way race over twenty-nine problems never used in selection, the signed program sits in the same band as the generated pure Nelder–Mead control, ranking eighth of fourteen overall, competitive on problems of eight dimensions or fewer and collapsing above twenty-four, where its $\sqrt{n}$-scaled exclusion zones swallow the search space.

Racing the ablated twin on the same 580 held-out instances settles the attribution there too: it ranks better than the signed original at every budget, including the low-dimensional slice where the signed program had looked like a specialist (mean rank 5.8 against 6.4), and wins the paired comparison on ties-excluded instances 57% to 43%. The avoidance mechanism subtracted value everywhere it was measured.

The trap generalises and is worth stating plainly. A single-draw leaderboard rewards whatever the label elicited, not what the label claims. Hand-picked recipes here received cleaner prompts than the random signed recipes, five of sixteen of which produced degenerate programs, so the canonical form's win partly measured promptability. Only the paired ablation, the same program with the signature feature switched off on the same instances, assigns credit causally.

## 3 What follows

What survives is the machinery, not yet any evidence it helps. Signed recipes are expressible, the model implements avoidance faithfully, and the extension is cheap to search. Whether any point outside the simplex beats its own projection back inside is now a well-posed question, and the first candidate answered it in the negative.

The protocol for a second round follows from the trap. Every signed candidate is scored against its ablated twin on identical instances, and only the paired difference counts; selection runs on a dimension-spread suite; and the repulsion is prompted to scale per-coordinate rather than with $\sqrt{n}$. Variants worth pricing under that protocol include slot-specific avoidance (repel A's moves only where A is weak) and time-varying signs (repel early, embrace late).

*Reproducibility.* The harness is `papers/dfo_recommender/e13_signed.py` in the humpday repository [2]; runs are `runs/e13_signed.json` and `runs/e13b_oos.json`, with every generated program in `runs/e13_code/`.

## References


[1] P. Cotton. Alloy: a machine-designed derivative-free optimizer, and a full account of the search that found it. Working paper, `https://humpday.microprediction.org/papers.html`, 2026.

[2] P. Cotton. HumpDay: recommendations and comparisons of derivative-free optimizers. `https://humpday.microprediction.org`, 2026.

[3] F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13(5):533–549, 1986.