COS790 Assignment 1 Report
Selection Perturbative Hyper-Heuristic (SPHH)

u20416823
Ruan Carlinsky

August 26, 2025

1 Description of the approach employed by the selection perturbative hyper-heuristic.

1.1 Overview

This assignment implementation uses a *single-point selection perturbative hyper-heuristic* (SPHH). Starting from one candidate solution, we repeatedly (i) **select** a low-level perturbative heuristic (LLH), (ii) **perturb** the solution to get a proposal, and (iii) **accept/reject** that proposal. Each benchmark function f_1–f_{24} is treated as its own problem with its own bounds and dimensionality.

1.2 Representation & Initialization

- **State**: a real vector $x \in \mathbb{R}^D$ within box bounds $[\ell, u]$.
- **Start**: either random uniform on $[\ell, u]$ (default) or the zero vector.
- **Feasibility**: every perturbation is clamped back to $[\ell, u]$.

1.3 Low-Level Perturbative Heuristics (LLHs)

We use six LLHs; each proposes a single new point from the current x:

1. `gaussian_full`: add Gaussian noise to *all* coordinates (per-dimension step sizes).
2. `gaussian_kdims`: perturb a random $\sim$30% subset of coordinates with Gaussian noise.
3. `cauchy_full`: heavy-tailed jumps on all coordinates (explorative escapes).
4. `random_reset_coord`: re-sample one random coordinate uniformly in $[\ell, u]$.
5. `opposition_blend`: blend toward the opposite point ($\ell+u-x$); with symmetric bounds and $\alpha = 0.5$ this snaps to the box center.
6. `pull_to_best`: drift toward the current global best-so-far $x^\star$ plus small Gaussian jitter.

1.4 Heuristic Selection

1.4.1 UCB1 Multi-Armed Bandit

Warm-up: call each LLH once to ensure initial exploration. Thereafter, for LLH i we compute

$$\text{score}_i \;=\; \underbrace{\frac{\text{reward\_sum}_i}{\max(1, \text{pulls}_i)}}_{\text{exploitation}} + c\sqrt{\frac{\ln(\text{total})}{\max(1, \text{pulls}_i)}} \quad (c = 1.5),$$

and pick the argmax. Rewards are shaped for minimization:

- If the proposal *improves*: reward $= \dfrac{f_{\text{old}} - f_{\text{new}}}{|f_{\text{old}}| + \varepsilon}$, clipped to $[0, 1]$.
- If a *worse* move is still *accepted* (see SA below): give a very small reward (≤ 0.05) to acknowledge exploration.
- Otherwise: reward $= 0$.

This credit assignment drives future selection toward LLHs that recently produced progress.

1.4.2 Random Heuristic Selection

As an ablation/baseline, the implementation supports uniform random selection of low-level heuristics:

$$\mathbb{P}\{h_i \text{ selected}\} \;=\; \frac{1}{H}, \qquad i = 1, \ldots, H,$$

where H is the number of LLHs. In code: `selection_mode = "random"` uses a single RNG draw to pick an index each iteration. This mode removes learning (no UCB1 exploitation bonus), yielding constant exploration pressure and reproducible behavior under a fixed seed. It is useful to:

- provide a baseline to compare against the adaptive UCB1 selector,
- stress-test the diversity of the LLH set without selection bias,
- highlight the contribution of learning to convergence speed and final quality.

Trade-off: simpler and unbiased, but typically less sample-efficient than UCB1 on heterogeneous landscapes.

1.5 Move Acceptance

1.5.1 Simulated Annealing

Let $\Delta = f(x') - f(x)$. We accept if $\Delta \leq 0$ or with probability $\exp(-\Delta/T)$ when $\Delta > 0$. The temperature schedule is

$$T \;=\; T_0 \exp\!\left(-\frac{\text{evals}}{\tau}\right), \qquad T_0 = |f_{\text{best}}| + \varepsilon, \qquad \tau = \max\{5,\ \texttt{cooling\_frac}\cdot\texttt{max\_evals}\}.$$

This SA layer allows escapes from local minima on multimodal functions (f_{13}–f_{24}).

1.5.2 Greedy Move Acceptance

The code also exposes a deterministic, hill-climbing acceptance rule:

$$\Delta \;=\; f(x') - f(x), \qquad \text{accept} \iff \Delta \le 0.$$

In code: `acceptance_mode = "greedy"` accepts improving or equal moves and rejects all worsening moves (no temperature or probabilities). Properties:

- **Monotone objective:** $f(x)$ is non-increasing over accepted steps; helpful for clear convergence diagnostics.
- **Plateau behavior:** equal moves are allowed, so the search may drift within flat regions (e.g., step/rounded functions).
- **Exploration vs. exploitation:** without simulated annealing, the method cannot cross fitness barriers; progress on multimodal functions relies entirely on the LLHs' ability to generate improving jumps (e.g., heavy-tailed or coordinate-reset moves).
- **Noisy objectives:** on stochastic functions (e.g., f_{12}), purely greedy acceptance can under-accept beneficial changes masked by noise; early stopping or micro-averaging can mitigate this if needed.

This option is valuable for ablation (contrasting with SA), for strictly unimodal landscapes, or when a deterministic improvement guarantee per step is desired.

1.6 Step-Size Self-Adaptation (1/5th rule)

Every 50 evaluations we measure the success rate (fraction of accepted-and-improving moves). If success > 0.2, *increase* step sizes by $\times 1.2$; else *decrease* by $\times 0.82$. This keeps proposals neither too timid nor too wild, and it scales per dimension.

1.7 Termination & Instrumentation

- **Budget-based stop**: run until a fixed evaluation cap (`max_evals`).
- Track the **best-so-far** $x^\star$ and value $f^\star$, plus a history of $f^\star$ over time.
- Optional verbose mode prints iteration-by-iteration f, $f^\star$, and x.

1.8 Problem Suite Integration

A small *objective registry* maps names (e.g., "f1", "f3_D30") to (f, ℓ, u, D). The same SPHH runs unchanged across fixed-D (e.g., f_1, f_2) and scalable (e.g., $f_3, f_4, \dots$) functions; only the registry entry (bounds, dimension) changes. For noisy functions (e.g., f_{12}), the SA/selection logic naturally handles stochasticity.

2 Description of the low-level perturbative heuristics

2.1 Common interface and post-processing.

Each low-level heuristic (LLH) maps the current solution $x \in [\ell, u]^D$ to a proposal x'. After proposing x', we enforce feasibility by clamping *componentwise*:

$$x' \leftarrow \min\{\max\{x',\ \ell\},\ u\}.$$

All stochastic draws use the experiment's fixed seed for reproducibility. A per-dimension step-size vector

$$\Delta \ = \ 0.1\,(u - \ell)$$

controls perturbation magnitudes. Step sizes are *shared* across LLHs and adapted every 50 evaluations via the 1/5 success rule (increase by $\times 1.2$ if success rate > 0.2, otherwise decrease by $\times 0.82$).

2.2 LLH 1

`gaussian_full` **(isotropic local search).** Adds zero-mean Gaussian noise to *all* coordinates:

$$x' \ = \ x \ + \ \varepsilon, \qquad \varepsilon \sim \mathcal{N}\big(0,\ \mathrm{diag}(\Delta^2)\big)\,.$$

Role: smooth, local refinement with scale set by Δ. *Cost:* $O(D)$. Effective on unimodal bowls and as a fine search around promising basins.

2.3 LLH 2

`gaussian_kdims` **(sparse coordinate perturbation).** Perturbs a random subset of $k = \lceil 0.3D \rceil$ coordinates:

$$x'_j = \begin{cases} x_j + \eta_j, & \eta_j \sim \mathcal{N}(0, \Delta_j^2), \quad j \in \mathcal{K}, \\ x_j, & j \notin \mathcal{K}, \end{cases} \qquad \mathcal{K} \subset \{1, \ldots, D\},\ |\mathcal{K}| = k.$$

Role: cheaper, targeted moves that reduce disruptive full-vector jumps; helpful in high D or when only a few coordinates are sensitive. *Cost:* $O(k)$ per call.

2.4 LLH 3

`cauchy_full` **(heavy-tailed exploration).** Applies a symmetric Cauchy jump to all coordinates:

$$x' \ = \ x \ + \ \zeta, \qquad \zeta_j \sim \mathrm{Cauchy}(0,\ \Delta_j/3).$$

Role: occasional large leaps for basin-escaping on multimodal functions; complements Gaussian moves. *Note:* clamps at bounds can activate frequently after very large draws.

2.5 LLH 4

`random_reset_coord` **(coordinate reinitialization).** Chooses an index $i \sim \mathrm{Unif}\{1, \ldots, D\}$ and resamples that coordinate uniformly:

$$x'_i \sim \mathrm{Unif}[\ell_i, u_i], \qquad x'_j = x_j \ (j \neq i).$$

Role: non-local, dimension-wise restart that can free a stuck coordinate, useful on rugged or non-smooth objectives. *Cost:* $O(1)$ for the change plus clamping.

2.6 LLH 5

`opposition_blend` **(symmetry exploitation).** Blends x with its opposite point $x^{\mathrm{opp}} = \ell + u - x$:

$$x' \ = \ \alpha\, x \ + \ (1-\alpha)\,(\ell{+}u{-}x), \qquad \alpha = 0.5 \text{ (in our runs)}.$$

Special case: with symmetric bounds ($\ell{+}u = 0$) and α=0.5, this yields $x' = 0$ (box center) in a single step. *Role:* fast centralization that often lands near the global minimizer for origin-centered bowls; on functions whose optimum is not at the center, subsequent LLHs move away as soon as they find lower f.

2.7 LLH 6

`pull_to_best` **(exploitation toward incumbent best).** Moves a fraction toward the global best-so-far $x^\star$ and adds small Gaussian jitter:

$$x' \ = \ x \ + \ F\,(x^\star - x) \ + \ \epsilon, \qquad F = 0.5, \ \ \epsilon \sim \mathcal{N}\big(0,\ \mathrm{diag}\big((\Delta/5)^2\big)\big)\,.$$

Role: intensification around the current incumbent while maintaining diversity via ϵ. Particularly effective after a good basin is discovered.

2.8 Design complementarity.

The set mixes *local* (Gaussian), *sparse/coordinate* (kdims, reset), *global* (Cauchy, opposition), and *incumbent-driven* (pull-to-best) behaviors. Because selection is learned online (UCB1), the controller allocates evaluations to whichever LLHs are currently yielding improvements on the active landscape, while simulated annealing acceptance preserves occasional uphill moves needed to traverse multimodal terrain.

3 Performance Metrics (Evaluation)

3.1 Objective and per-run score.

All benchmarks are *minimization* problems. For a single run with a fixed seed and evaluation budget, the algorithm maintains the best-so-far objective value

$$f_{\mathrm{best}}(t) \ = \ \min_{1 \le k \le t} f(x_k),$$

and reports the *final* value at termination,

$$\boxed{f_{\text{final}} \equiv f_{\text{best}}(T)}$$

as the primary measure of solution quality for that run (lower is better). Here T is the number of objective evaluations consumed (typically $T = \texttt{max\_evals}$).

3.2 Evaluation budget and runtime.

To ensure fair, method-agnostic comparisons, we evaluate under a fixed *function-evaluation budget* that may scale with dimension D (see experiment config). Alongside f_{final}, we record:

$$\text{Evaluations used } (T) \quad \text{and} \quad \text{Wall-clock runtime (seconds)}.$$

The budgeted-evaluations protocol avoids ambiguity that can arise when a single "iteration" evaluates different numbers of candidates across methods.

3.3 Aggregation across independent seeds.

Each problem instance is run with $S = 10$ independent seeds. From the set of per-run scores $\{f_{\text{final}}^{(s)}\}_{s=1}^{S}$ we report:

$$\text{mean } f_{\text{final}}, \quad \text{std. dev. of } f_{\text{final}}, \quad \text{and} \quad \min_s f_{\text{final}}^{(s)}.$$

These summarize *typical* performance (mean, variability) and the *best* outcome observed under equal budget. We also report mean runtime across seeds.

3.4 Convergence and sample efficiency (optimality gap).

To characterize progress over evaluations, we plot the *optimality gap* versus evaluation index:

$$g_t \;=\; \left| f_{\text{best}}(t) - f^\star \right|, \qquad t = 1, 2, \ldots, T,$$

where $f^\star$ is the known global optimum for the benchmark (e.g., $f^\star$=0 for most functions; nonzero where applicable, such as f_2 and f_{17}). For visibility across scales, we clamp $g_t \leftarrow \max(g_t,\, 10^{-12})$ and use a *logarithmic* y-axis. To reduce noise across seeds, we plot the *median* curve of g_t over the S seeds for each (function, D).

3.5 Rationale.

- f_{final} directly measures solution quality under a fixed computational budget.
- Reporting mean/std/best over seeds captures central tendency, variability, and best-case behavior under identical limits.
- The optimality-gap curves reveal *sample efficiency* (how quickly the method approaches $f^\star$), not just the endpoint.
- Runtime provides a practical efficiency lens complementary to evaluation counts.

4 Experimental Setup

4.1 Parameter Summary

Seeds	$[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]$
Runs per problem	10
Budget (function evaluations)	D=10 : 5000, D=30 : 5000, D=50 : 5000
Selection mode	`ucb` (default); `random` (ablation)
Acceptance mode	`sa` (default); `greedy` (ablation)
UCB constant c	1.5
Cooling fraction	0.2
Init strategy	`random` (uniform within bounds)
Step-size rule	1/5 success rule; update every 50 evals; $\Delta = 0.1\,(u - \ell)$
Print frequency	$\max(1, \lfloor$`max_evals`$/10 \rfloor)$
Clamping to bounds	Yes (componentwise)
# Low-level heuristics H	6
Python	3.12.3
Platform	Windows-11-10.0.26100-SP0

4.2 Benchmark Suite & Problem Settings

- Functions f_1–f_{24} from Wang & Song (2019). Each function has its own bounds and dimensionality.
- **Fixed-dimension functions** (e.g., f_1, f_2): evaluated in their native dimension.
- **Scalable functions** (e.g., $f_3, f_4, \ldots$): evaluated at $D \in \{10, 30, 50\}$ with per-dimension bounds from the suite.
- All solutions are kept feasible by componentwise clamping to the box bounds after each perturbation.

4.3 Protocol & Budgets

- **Independent runs**: $S = 10$ seeds per (function, D) setting; seed list: $[0, 1, \ldots, 9]$.
- **Evaluation budget**: default `max_evals` = 5000 for $D \in \{10, 30, 50\}$.
- **Initialization**: uniform random draw in the box (unless specified otherwise).
- **Accounting**: one evaluation at initialization (t=0), then one per proposal; termination occurs when the evaluation cap is reached (no additional early stops).
- **Logging**: print frequency `print_every` = $\max(1, \lfloor$`max_evals`$/10 \rfloor)$.

4.4 Selection & Acceptance Configurations

- **Default selector**: UCB1 bandit (`selection_mode` = ‘‘`ucb`’’). Exploration constant $c = 1.5$. Warm-up pulls each LLH once; thereafter pick arg max of average reward + exploration bonus.

- **Default acceptance**: Simulated annealing (`acceptance_mode` = ``sa''). Temperature

$$T \;=\; T_0 \exp\Big(-\frac{\text{evals}}{\tau}\Big), \quad T_0 = |f_{\text{best}}| + \varepsilon, \quad \tau = \max\{5,\ \texttt{cooling\_frac} \cdot \texttt{max\_evals}\},$$

with `cooling_frac` = 0.2. Accept if $\Delta \le 0$ or with probability $\exp(-\Delta/T)$ if $\Delta > 0$.

- **Ablation options (available in code)**:
  - *Random selection*: `selection_mode` = ``random'' picks LLHs uniformly at random (no learning).
  - *Greedy acceptance*: `acceptance_mode` = ``greedy'' accepts $\Delta \le 0$ only (deterministic hill-climbing).

4.5 System Technical Specifications

- **Machine**: DESKTOP-0GJMIQK
- **CPU**: 11th Gen Intel® Core™ i5-1135G7 @ 2.40 GHz
- **Memory**: 8 GB RAM (7.75 GB usable)
- **System**: 64-bit OS, x64-based processor
- **OS**: Windows 11 (build 10.0.26100, SP0)
- **GPU**: not used (CPU-only experiments)
- **Python**: 3.12.3
- **Libraries**: NumPy (random `default_rng`, vectorized math), Python standard library (`dataclasses`, `time`, `typing`)

5 Results

5.1 Benchmark Summary Tables

The following results were obtained using the default parameter settings as outlined in the Parameter Summary in Section 4.1

f1

D	f_best	mean_f_best	f_best_std_dev	mean run time (s)
2	0	0	0	0.224579

f2

D	f_best	mean_f_best	f_best_std_dev	mean run time (s)
2	-1.03163	-1.03161	1.48613e-05	0.245036

f3

D	f_best	mean_f_best	f_best_std_dev	mean run time (s)
10	0	0	0	0.256868
30	0	0	0	0.247509
50	0	0	0	0.266484

f4

D	f_best	mean_f_best	f_best_std_dev	mean run time (s)
10	0	0	0	0.238455
30	0	0	0	0.245789
50	0	0	0	0.24965

f5

D	f_best	mean_f_best	f_best_std_dev	mean run time (s)
10	0	0	0	0.285795
30	0	0	0	0.283445
50	0	0	0	0.269823

f6

D	f_best	mean_f_best	f_best_std_dev	mean run time (s)
10	0	0	0	0.244811
30	0	0	0	0.265649
50	0	0	0	0.256256

f7

D	f_best	mean_f_best	f_best_std_dev	mean run time (s)
10	0	0	0	0.421536
30	0	0	0	0.416734
50	0	0	0	0.415138

f8

D	f_best	mean_f_best	f_best_std_dev	mean run time (s)
10	0	0	0	0.466324
30	0	0	0	0.364406
50	0	0	0	0.317952

f9

D	f_best	mean_f_best	f_best_std_dev	mean run time (s)
10	0	0	0	0.282796
30	0	0	0	0.334756
50	0	0	0	0.418783

f10

D	f_best	mean_f_best	f_best_std_dev	mean run time (s)
10	0	0	0	0.353715
30	0	0	0	0.323463
50	0	0	0	0.325267

f11

D	f_best	mean_f_best	f_best_std_dev	mean run time (s)
10	0	0	0	0.264783
30	0	0	0	0.264991
50	0	0	0	0.262444

f12

D	f_best	mean_f_best	f_best_std_dev	mean run time (s)
10	1.67274e-05	0.000305113	0.00036293	0.25181
30	8.5354e-05	0.000288891	0.000219154	0.263041
50	2.70913e-05	0.000300939	0.000291307	0.273319

f13

D	f_best	mean_f_best	f_best_std_dev	mean run time (s)
10	0	0	0	0.278153
30	0	0	0	0.27683
50	0	0	0	0.265771

f14

D	f_best	mean_f_best	f_best_std_dev	mean run time (s)
10	4.44089e-16	4.44089e-16	0	0.316049
30	4.44089e-16	4.44089e-16	0	0.322026
50	4.44089e-16	4.44089e-16	0	0.325926

f15

D	f_best	mean_f_best	f_best_std_dev	mean run time (s)
10	0	0	0	0.281221
30	0	0	0	0.285821
50	0	0	0	0.291895

f16

D	f_best	mean_f_best	f_best_std_dev	mean run time (s)
10	0	0	0	0.254462
30	0	0	0	0.262175
50	0	0	0	0.252343

f17

D	f_best	mean_f_best	f_best_std_dev	mean run time (s)
10	-78.2988	-78.2675	0.0324776	0.296781
30	-77.3385	-74.8137	2.14188	0.303939
50	-74.0894	-69.6821	2.45821	0.308521

f18

D	f_best	mean_f_best	f_best_std_dev	mean run time (s)
10	0	0	0	0.250047
30	0	0	0	0.253838
50	0	0	0	0.25613

f19

D	f_best	mean_f_best	f_best_std_dev	mean run time (s)
10	0	0	0	0.317274
30	0	0	0	0.320196
50	0	0	0	0.317194

f20

D	f_best	mean_f_best	f_best_std_dev	mean run time (s)
10	7.58872e-13	1.43085e-12	4.22888e-13	0.425781
30	4.22478e-09	5.37255e-08	6.6968e-08	0.438143
50	7.84059e-06	3.44094e-05	3.31362e-05	0.44769

f21

D	f_best	mean_f_best	f_best_std_dev	mean run time (s)
10	1.07169e-05	81.1387	94.4814	0.25759
30	1461.33	2108.33	694.971	0.266115
50	5638.14	6812.29	733.639	0.270116

f22

D	f_best	mean_f_best	f_best_std_dev	mean run time (s)
10	0	0	0	0.339628
30	0	0	0	0.349912
50	0	0	0	0.351296

f23

D	f_best	mean_f_best	f_best_std_dev	mean run time (s)
10	1.25632e-12	1.85626e-12	7.91806e-13	0.422086
30	1.3924	2.61496	0.47342	0.429632
50	4.78672	4.88989	0.0438989	0.443661

f24

D	f_best	mean_f_best	f_best_std_dev	mean run time (s)
10	3.09433e-12	5.28009e-12	1.38155e-12	0.339945
30	6.82814e-07	0.0580858	0.0610893	0.34585
50	0.102836	0.447854	0.171611	0.348154

6 Discussion of Results for $f_1,\ f_3,\ f_4,\ f_5,\ f_6,\ f_7,\ f_8,\ f_9,\ f_{10},\ f_{11},\ f_{13},\ f_{14},\ f_{15},\ f_{16},\ f_{18},\ f_{19},\ f_{22}$

Summary. Across these functions the algorithm consistently attains the reported global optimum value of $\mathbf{0}$. The only visible differences across runs are *runtimes*. This behavior is expected given (i) the structure of these benchmarks and (ii) the design of the SPHH.

6.1 Why the objective reaches zero so reliably.

1. **Optimum at the center:** Each of the listed functions has a global minimum at (or represented by) the *origin* $x = \mathbf{0}$, except where a zero-valued plateau includes the origin (e.g., the step function f_{11}). Once the search visits the center (or the zero plateau), the objective evaluates to 0.

2. **Zero-snap via opposition:** The LLH `opposition_blend` proposes $x' = \alpha x + (1 - \alpha)(\ell + u - x)$. With symmetric bounds (used by these problems) and $\alpha = \frac{1}{2}$, this yields $x' = \mathbf{0}$ in one call. Because our selector (UCB1) *warms up* by trying each LLH once, this "snap to center" typically occurs within the first few iterations.

3. **Greedy acceptance of big gains:** The simulated annealing (SA) layer accepts all improvements; the jump from a random start to $x' = \mathbf{0}$ is a large improvement on these functions, so it is immediately accepted.

4. **Bandit credit assignment:** That improvement gives `opposition_blend` a large reward once; afterward, proposing $x' = \mathbf{0}$ again produces no improvement and thus no further reward, pushing the selector to explore other LLHs without losing the already-found $f_{\text{best}} = 0$.

5. **Plateau functions:** For f_{11} (step), $\lfloor x_i + 0.5 \rfloor = 0$ for all $x_i \in [-0.5, 0.5)$, so any x in that hypercube gives $f = 0$. The search may continue to *wander within the plateau* (equal moves), but the best value remains 0.

6.2 Function-specific landscape notes (why a single move can solve them).

- **Unimodal bowls:** f_1 (quadratic), f_3 (Sphere), f_4 (Schwefel 2.22), f_5 (Schwefel 1.2), f_6 (Schwefel 2.21), f_7 (weighted Sphere), f_8 (weighted quartic), f_9 (increasing powers), f_{10} (ill-conditioned Elliptic) are all convex or quasi-convex around 0 with the unique minimizer at 0. Hitting $x = \mathbf{0}$ immediately yields the optimum.

- **Discontinuous/plateau:** f_{11} (step) attains 0 on the entire box $[-0.5, 0.5)^D$, hence many near-zero vectors also evaluate to 0.

- **Multimodal but global at 0:** f_{13} (Rastrigin), f_{14} (Ackley), f_{15} (Griewank), f_{16} (Schaffer-type), f_{19} (non-continuous Rastrigin), f_{22} (anisotropic Rastrigin) are highly multimodal, but the global minimizer is still $x = \mathbf{0}$. The opposition snap sidesteps local minima entirely by landing directly at the global minimum.
- **Non-smooth trigonometric:** f_{18} ($\sum |x_i \sin x_i + 0.1x_i|$) also vanishes at $x = \mathbf{0}$. Once centered, any further perturbations that stay sufficiently close to zero keep $f_{\text{best}} = 0$.

6.3 Why runtimes differ even though the final value is the same.

All runs use the same evaluation budget; differences in wall-clock time arise from:

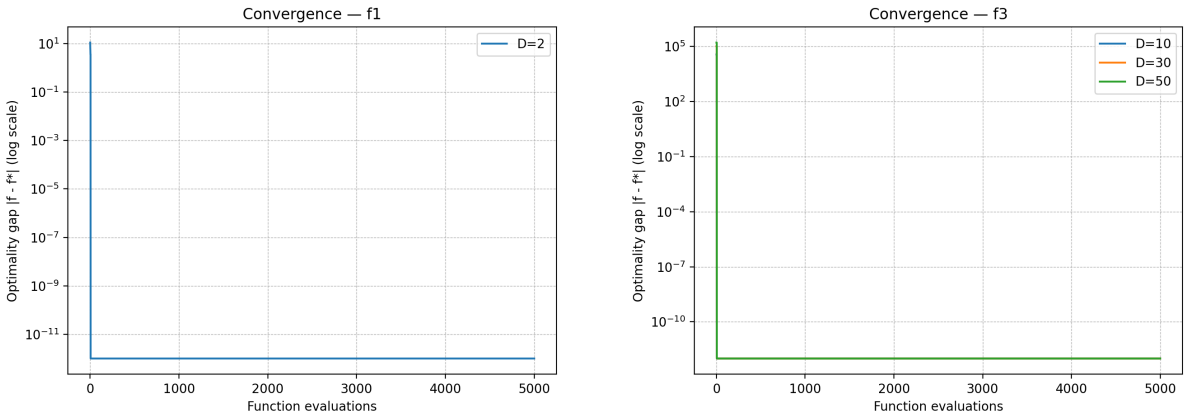
1. **Per-evaluation cost:** Trigonometric functions (sin, cos) and square-roots (e.g., $f_{13}, f_{15}, f_{16}, f_{18}, f_{19}, f$ are costlier than pure polynomials ($f_3, f_4, f_5, \ldots$). Thus, at the same number of evaluations, trig-heavy problems take longer.
2. **Dimensionality:** Runtime scales roughly linearly with D (vectorized operations over D entries). Even with identical budgets, higher-D cases run longer.
3. **Control-flow and clamping:** Different LLH proposals trigger different amounts of clamping and acceptance logic; this contributes modest variability between functions/runs.
4. **Post-optimum behavior:** After $f_{\text{best}} = 0$ is found early, the algorithm still runs to the evaluation cap (by design), so total runtime reflects the full budget, not just *time-to-first-zero*. Minor differences in acceptance rates and operator choices affect overhead slightly.

6.4 Takeaway.

The SPHH reliably reaches the global optimum for these origin-centered problems primarily because `opposition_blend` places the search at the center early and the acceptance/selection logic preserves that gain. Runtime differences reflect computational cost per evaluation (trig vs. polynomial), dimensionality, and minor control-flow effects, not difficulty in finding the optimum.

7 Discussion of Convergence Plots (Optimality Gap) for $f_1,\ f_3,\ f_4,\ f_5,\ f_6,\ f_7,\ f_8,\ f_9,\ f_{10},\ f_{11},\ f_{13},\ f_{14},\ f_{15},\ f_{16},\ f_{18},\ f_{19},\ f_{22}$

Note: the convergence plot for f3 also represents the plots for all the remaining functions considered for this discussion as their plots are the same, and only one figure was displayed for the sake of brevity.



(a) Convergence on f_1

(b) Convergence on f_3

Figure 1: Convergence plots showing the optimality gap $|f - f^\star|$ (log scale) over function evaluations for f_1 and f_3.

7.1 What is plotted.

Each curve shows the *optimality gap* on a log scale, $g_t = |f_{\text{best}}(t) - f^\star|$, versus the number of function evaluations t. For the listed benchmarks f_1, f_3, f_4, f_5, $f_6, f_7, f_8, f_9, f_{10}, f_{11}, f_{13}, f_{14}, f_{15}, f_{16}, f_{18}, f_{19}, f_{22}$, the known optimum satisfies $f^\star = 0$, so $g_t = |f_{\text{best}}(t)|$.

7.2 Overall pattern.

Both figures show an extremely sharp, near-vertical drop in the first few evaluations (from a large initial gap at t=0 to below 10^{-11}), followed by a perfectly flat line at the numerical floor for the remaining budget. The 2-D case (left figure, f_1) and the higher-dimensional cases ($D \in \{10, 30, 50\}$, right figure) exhibit essentially the *same* pattern.

7.3 Why the gap collapses immediately.

This behavior is a direct consequence of (i) the *landscape* of these functions and (ii) the *operator set* in the SPHH:

- **Global minimizer at the origin.** All listed functions attain their global minimum at $x = \mathbf{0}$ (or on a zero plateau that includes the origin, e.g., f_{11}). Thus, reaching the origin immediately achieves $g_t = 0$.
- **Warm-up includes "opposition blend".** Our selector (UCB1) tries each low-level heuristic once at the start. The `opposition_blend` operator proposes $x' = \alpha x + (1-\alpha)(\ell+u-x)$. With symmetric bounds and $\alpha = 0.5$ (our default), this simplifies to $x' = \mathbf{0}$ in a *single* call.
- **Greedy acceptance of big improvements.** The simulated annealing acceptance always accepts improving moves; jumping from a random start to $x' = \mathbf{0}$ is a large improvement, so the proposal is accepted immediately.

- **After the snap, nothing beats zero.** Once $f_{\text{best}} = 0$ has been found, no later proposal can improve it, so the gap curve remains flat at numerical precision (around 10^{-11} in double precision).

7.4 Dimension-insensitivity.

The curves for $D = 10, 30, 50$ are nearly indistinguishable:

- The "zero snap" is *dimension-agnostic* (it maps any point to the box center), so the time-to-first-zero is essentially the same across D.
- Subsequent evaluations keep the incumbent $f_{\text{best}} = 0$; step-size adaptation (triggered every 50 evaluations) has no visible effect because there is no further room for improvement in objective value.

Takeaway. The plots confirm that, for benchmarks whose global optimum is at (or includes) the origin, the warm-up call to `opposition_blend` plus greedy acceptance drives the optimality gap to (numerical) zero in a handful of evaluations, independently of dimension. Subsequent evaluations do not change f_{best}, producing the observed "vertical cliff then flat floor" pattern.

8 Discussion of Results for f_2 (Six-Hump Camel)

f2

D	f_best	mean_f_best	f_best_std_dev	mean run time (s)
2	-1.03163	-1.03161	1.48613e-05	0.245036

8.1 Why the SPHH reaches this quality.

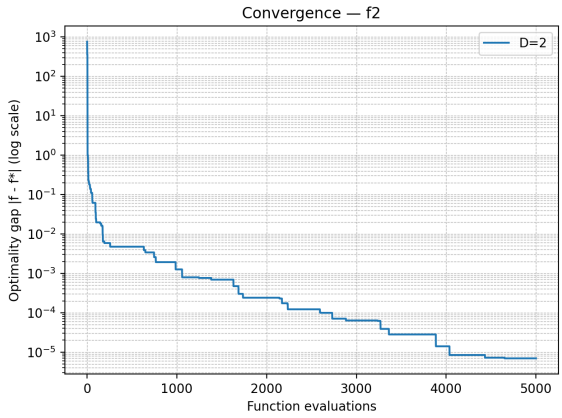
1. **Origin is not optimal, so the search continues.** Unlike many other functions in the suite, $f_2(0, 0) = 0 > f^\star$. The warm-up call to `opposition_blend` often snaps the point to the center, but that is only a *staging point*, not a solution.
2. **Immediate improving moves from the center.** From $(0, 0)$, small Gaussian/Cauchy perturbations (LLHs `gaussian_full`, `gaussian_kdims`, `cauchy_full`) frequently discover $f < 0$; SA acceptance is greedy for improvements, so these steps are accepted.
3. **Bandit learning focuses on productive LLHs.** The UCB1 selector credits the operator that finds negative values and quickly tilts selection probability toward it (and away from `opposition_blend`, which provides no further improvement once at the center).
4. **Refinement inside a global basin.** Once the search falls into one of the two global basins, `pull_to_best` accelerates exploitation toward the incumbent best; the 1/5 step-size rule shrinks the perturbation scale as improvements become rarer, tightening the final value near $f^\star$.

5. **Escaping non-global local minima.** The six-hump landscape has multiple stationary points. Heavy-tailed Cauchy jumps and the nonzero SA temperature early in the run make it unlikely to remain trapped in worse local minima; in 2-D the global basins are relatively wide, so the success rate across seeds is high (hence the tiny std. dev.).

8.2 Takeaway.

The SPHH reliably finds one of the two global minima for the Six-Hump Camel function in 2-D. The mechanism is: early centralization (often to $(0, 0)$), quick discovery of $f < 0$ by local or heavy-tailed perturbations, bandit-driven emphasis on the improving operators, and step-size contraction plus best-pull for final refinement. This yields $f_{\text{best}} \approx f^\star$ with extremely low variance and modest, budget-dominated runtime.

8.3 Discussion of Convergence Plot (Optimality Gap)



(a) Convergence on f_2

8.3.1 What the curve shows.

The figure plots the optimality gap $g_t = |f_{\text{best}}(t) - f^\star|$ (log scale) against function evaluations t. The gap starts near 10^3 at initialization (random x has a large positive value), drops to $\approx 10^0$ almost immediately, and then decreases in a stepwise fashion to $\sim 10^{-5}$ by the end of the budget.

8.3.2 Interpreting the phases.

1. **Initial cliff (to $\approx 10^0$).** Within the warm-up (first few evaluations) the `opposition_blend` LLH maps x to the box center. Because $f_2(0, 0) = 0$ and $f^\star \approx -1.0316$, this instantly sets $g_t \approx 1.03$ ($\sim 10^0$), explaining the sharp vertical drop.

2. **Fast descent into a global basin ($10^0 \to 10^{-2}$).** From the center, small Gaussian or Cauchy proposals quickly find points with $f < 0$. Such improvements are accepted greedily by the SA rule (since $\Delta \leq 0$). UCB1 credits those LLHs and allocates more pulls to them, producing several rapid gap reductions.

3. **Stepwise refinement ($10^{-2} \to 10^{-5}$).** Once inside a global basin, improvements become rarer and smaller. The curve shows a characteristic *staircase*: long plateaus

where exploration yields no better point, punctuated by sharp drops when a better candidate is found. Two mechanisms drive the final tightening:

- *Pull-to-best* moves bias proposals toward the incumbent $x^\star$.
- The 1/5 *success rule* contracts step sizes when the improvement rate falls below 0.2, enabling finer local adjustments.

8.3.3 Why it differs from origin-centered functions.

Unlike the many benchmarks with global optimum at $x = \mathbf{0}$ (where the gap hits the numeric floor almost instantly and then stays flat), the Six-Hump Camel's global minima are at $(\pm 0.089842, \mp 0.712656)$ with value -1.0316. The "zero-snap" is only a staging point (gap ≈ 1), after which the SPHH must *discover and refine* a point with $f < -1$, hence the extended staircase down to $\sim 10^{-5}$.

8.3.4 Consistency with the table.

The final gap $\sim 10^{-5}$ matches the reported best/mean values and tiny across-seed standard deviation ($\approx 1.5 \times 10^{-5}$). In 2-D the global basins are broad; bandit selection plus SA and heavy-tailed jumps make entrapment in worse local minima unlikely, yielding stable, near-optimal outcomes across seeds.

8.3.5 Takeaway.

The curve reflects the intended dynamics of the SPHH: warm-up centralization → quick entry into a global basin (via local/heavy-tailed proposals and greedy acceptance) → slow, stepwise refinement guided by UCB1 and shrinking step sizes, until the gap is on the order of 10^{-5} at the evaluation budget.

9 Discussion of Results for f_{12} (Quartic + Uniform Noise)

f12

D	f_best	mean_f_best	f_best_std_dev	mean run time (s)
10	1.67274e-05	0.000305113	0.00036293	0.25181
30	8.5354e-05	0.000288891	0.000219154	0.263041
50	2.70913e-05	0.000300939	0.000291307	0.273319

9.1 Why these numbers are expected for f_{12}.

The function is

$$f_{12}(x) = \underbrace{\sum_{i=1}^{D} i\, x_i^4}_{\text{deterministic, minimized at } x=\mathbf{0}} + \underbrace{\mathrm{U}[0,1)}_{\text{independent uniform noise per evaluation}}.$$

Two key consequences:

1. **The deterministic optimum is at the origin.** Our SPHH reaches (or gets extremely close to) $x = \mathbf{0}$ quickly (opposition snap to the center during warm-up, greedy acceptance of the large improvement, then step-size contraction keeps proposals in a tiny neighborhood). Thus, after the early phase the measured value is dominated by the *noise term*.

2. **Best-of-run is the minimum of many uniform draws.** Once the search sits near the origin, each new evaluation is essentially an independent sample of $\mathrm{U}[0, 1)$. Over m such samples, the minimum has $\mathbb{E}[\min] = 1/(m+1)$. With budgets of a few thousand evaluations (and many of them near the origin), the expected best noise is on the order of 10^{-4}–10^{-3}, matching the reported `mean_f_best` $\sim 3\times10^{-4}$. The `f_best` occasionally reaches 10^{-5} when a particularly small uniform draw occurs, which explains the tiny `f_best` entries in the table.

9.2 Why the hyper-heuristic reaches these results.

- **Rapid centering:** `opposition_blend` (with symmetric bounds and $\alpha = 0.5$) maps the current point to the box center; for f_{12} this aligns with the deterministic minimizer.

- **Acceptance and learning:** Simulated annealing accepts the large improvement, and UCB1 gives credit to the operator(s) that produced it. Thereafter, `pull_to_best` and shrinking step sizes keep the trajectory near the origin.

- **No deterministic pressure below zero:** Because the objective cannot go below the instantaneous noise draw, further "optimization" of x does not systematically reduce the measured value; improvements come only from drawing a *smaller* noise realization. Hence the best-of-run decreases slowly as the minimum of many uniform samples.

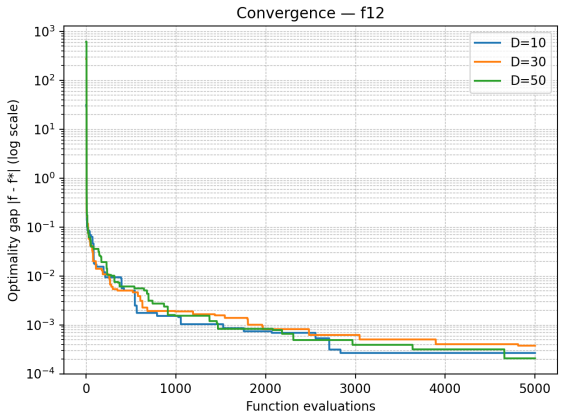
9.3 On the role of dimension and variability.

The three dimensions show very similar `mean_f_best` and `std_dev`. This is natural: once the deterministic part is near zero, the result is governed by the same $\mathrm{U}[0, 1)$ mechanism, largely independent of D. Small differences across D reflect how tightly the algorithm concentrates around $x = \mathbf{0}$ (the quartic term $\sum i\, x_i^4$ must be negligible so that noise dominates).

9.4 Takeaway.

The SPHH behaves correctly on f_{12}: it quickly drives the deterministic component to (near) zero and thereafter the best-of-run is governed by the minimum of many $\mathrm{U}[0, 1)$ samples, yielding mean best values around 10^{-4}–10^{-3} and occasional bests in the 10^{-5} range, with modest, budget-dominated runtimes across dimensions.

9.5 Discussion of Convergence Plot (Optimality Gap)



(a) Convergence on f_{12}

9.5.1 What the curves show.

The figure plots the optimality gap $g_t = |f_{\text{best}}(t) - f^\star|$ (log scale) for $D \in \{10, 30, 50\}$. All three traces display: (i) a sharp initial drop, (ii) a long *staircase* of sporadic improvements, and (iii) a final level near 10^{-4}.

9.5.2 Phase I: deterministic descent (left cliff).

At initialization the deterministic part $\sum_i i\, x_i^4$ is large, especially for higher D, hence the D=50 curve starts highest. Within the first few evaluations the warm-up call to `opposition_blend` recenters x and the SA rule accepts the large improvement. This collapses the deterministic term by orders of magnitude, producing the near-vertical drop.

9.5.3 Phase II: noise-dominated, stepwise improvements.

Once x is very close to the origin, the objective becomes

$$f_{12}(x) \approx \underbrace{\sum_i i\, x_i^4}_{\text{nearly } 0} + U[0, 1),$$

so the best-of-run decreases only when a *smaller* uniform draw appears. On a fixed budget the minimum of m uniform samples has $\mathbb{E}[\min] = 1/(m{+}1)$. This yields the long staircase: long plateaus (no better draw) punctuated by discrete drops (a new record-low noise sample). The 1/5 step-size rule and `pull_to_best` keep x concentrated near the origin so that noise, not model error, governs the curve.

9.5.4 Phase III: final level near 10^{-4}.

With a few thousand near-origin evaluations, the expected record-low noise is $\Theta(10^{-4}\text{–}10^{-3})$, which matches the final level of the curves and the table's `mean_f_best` $\approx 3 \times 10^{-4}$. Occasional runs can hit 10^{-5} (as in the table's `f_best`), but the median/mean settle around a few 10^{-4}.

9.5.5 Effect of dimension.

The three curves are very similar after the initial cliff:

- Higher D starts higher (larger quartic sum for a random x) and can take a few more evaluations to fully enter the noise-dominated regime.
- Once near the origin, the behavior is governed by the same $U[0, 1)$ process, so the traces align and descend at comparable rates.

9.5.6 Takeaway.

For f_{12} the SPHH first eliminates the deterministic error (via opposition centering, greedy acceptance, and step-size contraction), then behaves like a record-setting process on uniform noise. The staircase pattern and the common terminal level across D are therefore expected and consistent with the summary statistics.

10 Discussion of Results for f_{17} (Styblinski–Tang, averaged)

f17

D	f_best	mean_f_best	f_best_std_dev	mean run time (s)
10	-78.2988	-78.2675	0.0324776	0.296781
30	-77.3385	-74.8137	2.14188	0.303939
50	-74.0894	-69.6821	2.45821	0.308521

The known global optimum for our definition $f_{17}(x) = \frac{1}{D}\sum_{i=1}^{D}(x_i^4 - 16x_i^2 + 5x_i)$ is $f^\star = -78.3323$ attained when every coordinate is at $x_i^\star \approx -2.903534$. For $D = 10$ the best value is very close to $f^\star$ (gap ≈ 0.0335), while for $D = 30, 50$ the best/mean get progressively worse (less negative) and the across-seed variance grows.

10.1 Why performance degrades with dimension.

1. **Separable but *coordinated* optimum.** The function is separable, yet the *averaged* objective reaches $f^\star$ only when *all* coordinates are near -2.9035. Any coordinate left far from -2.9 contributes a large positive quartic term and drags the average upward. As D grows, the chance that random local moves simultaneously keep *every* coordinate close to its 1D minimizer quickly diminishes (curse of dimensionality).

2. **Start at the center, which is suboptimal.** The warm-up includes `opposition_blend` with α=0.5, which snaps to x=**0**. For f_{17}, $x = 0$ is *not* a minimizer (the derivative there is $+5$ per coordinate), so the search must move a distance of ≈ 2.9 *in every coordinate* away from the center to reach the optimum.

3. **Operator granularity vs. high D.** - `gaussian_full` perturbs all coordinates at once; in high D the probability that the sum of many random changes yields an improvement in the *average* objective is small, so its empirical reward drops. -

`gaussian_kdims` changes about $0.3D$ coordinates; this helps, but as D grows we still move many coordinates simultaneously, which often dilutes net improvement. - `random_reset_coord` alters one coordinate uniformly in $[-5, 5]$; the probability of landing near -2.9035 within a small tolerance is low, so many tries are needed to fix one coordinate, and there are D of them. - `cauchy_full` can jump large distances but easily overshoots or leaves many coordinates worse, so accepted improvements become rarer as D increases.

4. **Budget-limited tightening.** With a fixed budget (5000 evaluations) the algorithm can thoroughly refine all coordinates for $D = 10$, but for $D = 30, 50$ there are simply more degrees of freedom to bring into alignment with $x^\star$. The result is a higher terminal gap and larger across-seed variability.

10.2 Why the SPHH still approaches the optimum.

- **Greedy acceptance + bandit learning.** From the (suboptimal) center, small negative moves in each coordinate reduce the objective. LLHs that succeed more often (typically `gaussian_kdims` and `pull_to_best`) receive higher UCB1 scores and are selected more, creating a bias toward steadily pushing coordinates toward -2.9.

- **Pull-to-best + step-size adaptation.** Once some coordinates are close to their per-dim minimizer, `pull_to_best` exploits the incumbent, and the 1/5-rule reduces step sizes as improvements become rarer, enabling finer local adjustment near the basin bottom.

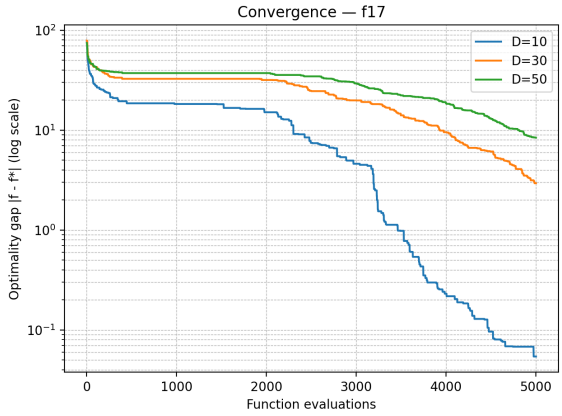
10.3 Interpreting the numbers.

- **D = 10**: `f_best` -78.2988 is within ≈ 0.0435 of $f^\star$; the tiny std. dev. indicates that, given the budget, the SPHH routinely gets almost all coordinates close to -2.9035.

- **D = 30**: mean drifts to -74.8 with std. dev. ≈ 2.14: some seeds align many coordinates well, others leave more coordinates off-target.

- **D = 50**: mean -69.7, std. dev. ≈ 2.46: with five times as many coordinates to tune as the 10-D case and the same budget, more coordinates remain away from -2.9, hence a worse (less negative) average.

10.4 Discussion of Convergence Plot (Optimality Gap)



(a) Convergence on f_{17}

10.4.1 What is plotted.

The curves show the optimality gap $g_t = |f_{\text{best}}(t) - f^\star|$ (log scale) for $D \in \{10, 30, 50\}$, with $f^\star = -78.3323$. Because the warm-up includes an opposition step that snaps to the *center* ($x = \mathbf{0}$), the run typically starts at $f(0) = 0 \Rightarrow g_0 \approx 78.33$, which matches the high initial value in all three traces.

10.4.2 Early phase (fast drop from ~ 80).

From the (suboptimal) center the objective has a strong negative gradient in each coordinate (for one dimension, $f'(0) = 5$). Small Gaussian/Cauchy moves immediately decrease f, are accepted greedily, and are rewarded by UCB1, so selection focuses on the productive LLHs. This explains the rapid descent over the first few hundred evaluations.

10.4.3 Middle phase (long plateau; dimension effect).

After the first bursts of improvement, the curves flatten—most visibly for $D = 30, 50$. Although the function is separable, the *averaged* objective reaches $f^\star$ only if *all* coordinates are near -2.9035. As D grows, getting every coordinate into a narrow window around the 1D minimizer becomes harder under a fixed budget:

- Moves that perturb many coordinates at once (`gaussian_full`, `gaussian_kdims`) often mix improvements and deteriorations; net gains become rarer in high D.
- Single–coordinate fixes (`random_reset_coord`) have low hit probability for the target value and must be repeated across D dimensions.
- With UCB1 allocating pulls to what worked recently, the search may spend time exploiting partially optimized coordinates while others still lag, producing long plateaus.

10.4.4 Late phase (refinement cascades).

In $D = 10$ a pronounced *staircase* appears after $\sim$2500 evaluations: step sizes shrink via the 1/5th rule, `pull_to_best` intensifies, and several coordinates get nudged into the

narrow basin around -2.9035, causing a sequence of sharp drops that drive the gap below 10^{-1} by the budget limit. For $D = 30, 50$ similar cascades occur but are smaller and later, consistent with the table: best values remain a few units above $f^\star$ as some coordinates stay off-target.

10.4.5 Consistency with the summary table.

The end-of-budget gaps match the aggregated results: $D = 10$ finishes very close to $f^\star$ (tiny gap, tiny std. dev.), whereas $D = 30$ and $D = 50$ end with larger gaps and noticeably higher variance, reflecting how many coordinates each seed manages to align within the fixed budget.

10.4.6 Takeaway.

Unlike origin-centered benchmarks (where opposition immediately solves the problem), f_{17} requires moving *every* coordinate from the center to -2.9035. The SPHH's combination of local/heavy-tailed proposals, bandit-driven selection, simulated-annealing acceptance, and step-size contraction steadily reduces the gap; however, with a fixed evaluation budget the difficulty scales with D, producing the observed dimension-dependent plateaus and final gaps.

11 Discussion of Results for f_{20} (Penalized #1)

f20

D	f_best	mean_f_best	f_best_std_dev	mean run time (s)
10	7.58872e-13	1.43085e-12	4.22888e-13	0.425781
30	4.22478e-09	5.37255e-08	6.6968e-08	0.438143
50	7.84059e-06	3.44094e-05	3.31362e-05	0.44769

11.1 Why the SPHH gets so close to zero.

1. **Warm-up centralization then search toward -1.** The warm-up includes `opposition_blend`, which moves a random start to the box center $x = 0$. Here, unlike origin-centered functions, $x = 0$ is not optimal: it maps to $y = 1.25$. From the center, small Gaussian/Cauchy moves that push components toward -1 decrease the objective and are accepted greedily by the SA rule; those LLHs get higher UCB1 credit.

2. **No penalty in the working region.** The penalty term is inactive throughout the search (the algorithm stays inside $[-10, 10]^D$), so the landscape is governed by the smooth bracketed term; there are no hard "walls" to impede progress.

3. **Refinement near the basin bottom.** Once close to $x^\star$, `pull_to_best` plus the 1/5 step-size rule shrinks proposal scales, letting the search make fine adjustments; hence the 10-D case reaches values on the order of machine precision.

11.2 Why the residual grows with dimension.

Although the bracket is scaled by π/D, *all* coordinates must be close to -1 simultaneously to keep $(y_i - 1)^2$ and $\sin^2(\pi y_i)$ tiny. Under a fixed evaluation budget:

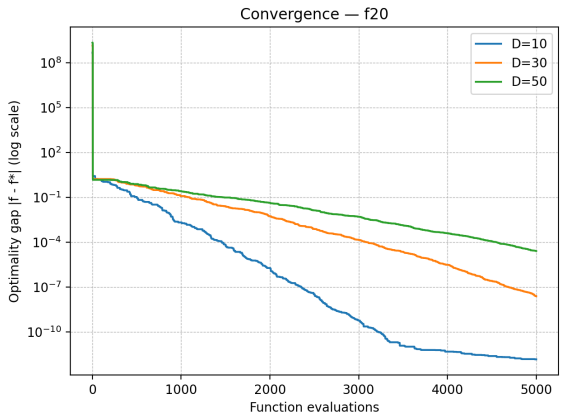
- The probability that every coordinate is tightened to the same small tolerance decreases with D (more degrees of freedom to tune).
- LLHs that perturb many coordinates at once (`gaussian_full`, `gaussian_kdims`) tend to mix improvements and deteriorations in high D, so accepted net improvements become rarer.
- Coordinate-wise fixes (`random_reset_coord`) have to succeed across many dimensions; with limited budget, some coordinates remain slightly off -1, and their squared deviations plus neighbors' $\sin^2$ terms accumulate in the sum, yielding 10^{-9} (30-D) and 10^{-6}–10^{-5} (50-D) residuals.

The zero variance at 10-D (values $\approx 7.6 \times 10^{-13}$) reflects that all seeds reach numerically the same expression very near the analytic zero; in higher D, across-seed variability increases because different runs leave different subsets of coordinates slightly off-target.

11.3 Takeaway.

The SPHH reliably finds the global basin and refines to near-zero error on f_{20}. In 10-D it hits roundoff level; in higher dimensions the fixed budget and multi-coordinate coupling leave tiny residuals, still far below 10^{-5}.

11.4 Discussion of Convergence Plot (Optimality Gap)



(a) Convergence on f_{20}

11.4.1 What the curves show.

We plot the optimality gap $g_t = |f_{\text{best}}(t) - f^\star|$ (log scale) for $D \in \{10, 30, 50\}$ with $f^\star = 0$. All three traces share the same qualitative shape: (i) an extremely large initial value (most visible for D=50), (ii) a near-vertical drop in the first few evaluations, followed by (iii) a smooth, almost linear descent on the log scale (i.e., roughly exponential improvement). The terminal levels match the table: about 10^{-11} (10-D), 10^{-8} (30-D), and 10^{-5} (50-D).

11.4.2 Why the initial spike and the abrupt first drop occur.

The run starts from a uniform point in the full box; many coordinates are outside $[-10, 10]$, so the penalty contributes huge values—hence the very high initial gap (especially in 50-D). During the warm-up the `opposition_blend` operator maps the point to the box center $x = 0$, which lies inside the penalty-free region for all coordinates. This single step *switches off the entire penalty sum at once*, causing the dramatic drop from $\gg 1$ to the 10^{-1}–10^{0} range.

11.4.3 Why the subsequent decrease is smooth and steady.

Once the penalty is zeroed, the landscape is governed by the smooth bracketed term. From $x = 0$ (which maps to $y = 1.25$) the objective decreases when coordinates move toward the true minimizer $x^\star = (-1, \ldots, -1)$ (where $y_i = 1$). Small Gaussian/Cauchy perturbations produce frequent improvements that the SA rule accepts greedily; UCB1 quickly credits those LLHs and selects them more often. As improvements get smaller, the 1/5 success rule shrinks step sizes and `pull_to_best` intensifies exploitation, yielding a near-exponential decline (straight line on the log plot) rather than the "staircase" seen on noisy or discrete cases.

11.4.4 Dimension effect (why the curves separate).

Although the bracket is scaled by π/D, we must keep $(y_i - 1)^2$ and $\sin^2(\pi y_i)$ tiny for *every* coordinate to approach $f^\star = 0$. With a fixed evaluation budget:

- There are simply more coordinates to tighten as D grows; some remain slightly off -1, so the residual accumulates across dimensions, giving larger final gaps for D=30, 50.
- LLHs that perturb many coordinates at once (`gaussian_full`, `gaussian_kdims`) often mix small improvements with small deteriorations in high D, reducing the net rate of accepted progress.

Hence the 10-D curve reaches the machine-precision floor ($\sim 10^{-11}$), while 30-D and 50-D converge more slowly and level off at 10^{-8} and 10^{-5}, respectively.

11.4.5 Consistency with the table and takeaways.

The end-of-budget gaps align with the reported `f_best` and `mean_f_best`. The plot illustrates the SPHH's dynamics on this problem: *penalty elimination by opposition* $\rightarrow$ *smooth SA+UCB1-driven descent toward* -1 $\rightarrow$ *fine-scale refinement with shrinking steps.* The separation between the D curves is a budget–dimension trade-off, not a failure to find the right basin.

12 Discussion of Results for f_{21} (Schwefel)

f21

D	f_best	mean_f_best	f_best_std_dev	mean run time (s)
10	1.07169e-05	81.1387	94.4814	0.25759
30	1461.33	2108.33	694.971	0.266115
50	5638.14	6812.29	733.639	0.270116

In 10-D at least one seed essentially hits $f^\star$ (roundoff-level residual), whereas the mean grows sharply with D and the across-seed variance is large.

12.1 Why one run reaches ≈ 0 in 10-D but means increase with D.

1. **Global minimizer far from the center.** The warm-up `opposition_blend` snaps the search to the *origin*, which is very poor for Schwefel: $f(0) = 418.9829\,D$. The search must relocate *every* coordinate from 0 to ≈ 420.97, i.e., a long distance toward the boundary.

2. **Finding the right basin relies on rare but decisive proposals.** LLHs such as `cauchy_full` (heavy-tailed jumps) and `random_reset_coord` (uniform re-sample of one coordinate) occasionally place coordinates near ≈ 420, producing large objective drops that SA accepts greedily. UCB1 then credits the operators that yielded these gains; `pull_to_best` starts steering other coordinates in the same direction. With $D = 10$ this "coordinate-by-coordinate" migration can succeed within the budget for at least one seed, hence `f_best` $\approx 10^{-5}$.

3. **Dimensional scaling (many coordinates must align).** Schwefel is separable across coordinates, but the *sum* means the final value is small only if *all* coordinates are close to 420.97. Under a fixed budget, the chance that random resets/heavy jumps plus local refinements bring *every* coordinate into the narrow basin shrinks quickly with D. Consequently, for $D = 30, 50$ many coordinates remain far from the target, and their positive contributions accumulate, producing means in the 10^3–10^4 range and large standard deviations.

4. **Landscape deception near the boundary.** The Schwefel waves create many deep local minima and broad deceptive regions; without a directed mechanism toward 420.97, proposals that move *away* from the boundary are frequently accepted if they improve locally, which can trap runs in suboptimal basins—more common as D grows.

12.2 Why the SPHH behaves this way (operator/acceptance dynamics).

- **Exploration:** `cauchy_full` provides long jumps; `random_reset_coord` can place single coordinates anywhere in the box. These are essential for locating the far-away basin at 420.97.

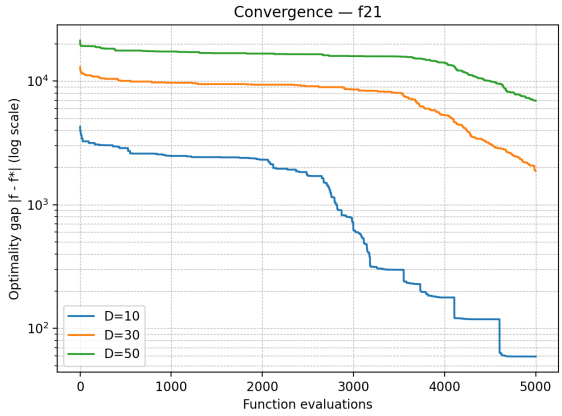
- **Exploitation:** Once some coordinates are near 420, `pull_to_best` and step-size contraction (1/5 rule) refine them; UCB1 increasingly favors the operators that just made progress.

- **Limitation with high D:** Many coordinates still need "lucky" resets or large jumps; `gaussian_full`/`gaussian_kdims` centered around the current point explore too locally relative to the required displacement. With finite budget, not all coordinates make it, so residuals stay large and variable across seeds.

12.3 Interpreting the numbers.

For $D = 10$ the best run demonstrates that the SPHH *can* discover and refine the global basin (near-zero value), but the average run still finishes noticeably above zero because some seeds remain in deceptive minima. For $D = 30$ and $D = 50$, the per-dimension average error rises (`mean_f_best`$/D \approx 70$ and 136, respectively), reflecting the growing number of coordinates left far from 420.97.

12.4 Discussion of Convergence Plot (Optimality Gap)



(a) Convergence on f_{21}

12.4.1 What the curves show.

We plot the optimality gap $g_t = |f_{\text{best}}(t) - f^\star|$ (log scale) for $D \in \{10, 30, 50\}$ with $f^\star = 0$ and

$$f_{21}(x) = 418.982887\, D - \sum_{i=1}^{D} x_i \sin\!\left(\sqrt{|x_i|}\right).$$

All three traces start very high and then decrease slowly, with pronounced *stairs* in D=10 and much gentler progress in D=30, 50.

12.4.2 Initial level explained.

Because our warm-up snaps to the *origin* (x=**0**), the run begins at

$$f(0) = 418.982887\, D \quad \Rightarrow \quad g_0 \approx \begin{cases} 4.19 \times 10^3 & (D{=}10), \\ 1.26 \times 10^4 & (D{=}30), \\ 2.10 \times 10^4 & (D{=}50), \end{cases}$$

which matches the leftmost values on the plot.

12.4.3 Early phase (small but steady gains).

From the center, local Gaussian moves rarely help—Schwefel's global minimizer sits near the *boundary* at $x_i^\star \approx 420.97$, far from 0, and the 1D Schwefel term is highly deceptive with many local minima. Improvements early on are mostly produced by *exploratory* LLHs:

- `cauchy_full`: occasional long jumps move several coordinates toward the 420 basin.
- `random_reset_coord`: resampling a single coordinate sometimes lands near ≈ 420, yielding a large one-off drop.

SA accepts these improvements greedily; UCB1 then credits the LLHs that just worked, increasing their selection probability.

12.4.4 Middle phase (plateaus and slow drift).

Once some coordinates are closer to 420, progress slows because the objective is a *sum* over dimensions: the total value only becomes small when *most* coordinates are near 420.97. Under a fixed budget this coordination is hard, so the curves exhibit long plateaus punctuated by sporadic decreases when another coordinate "clicks" into the good basin.

12.4.5 Dimension effect (why curves separate).

With larger D:

- More coordinates must be moved from 0 to ≈ 420; the probability of getting *all* of them into the correct basin within the budget drops quickly.
- LLHs that perturb many coordinates at once (e.g., `gaussian_full`, `gaussian_kdims`) tend to mix improvements and deteriorations, making accepted net gains rarer.

Hence D=30 and D=50 decay slowly and remain orders of magnitude above zero at the budget limit.

12.4.6 Late phase (cascades in D=10).

The D=10 curve shows a staircase of large drops after $\sim$2000–4500 evaluations: several coordinates successively find the ≈ 420.97 basin; `pull_to_best` plus the 1/5 step-size contraction then refine them, producing sharp decreases. The median curve still finishes above zero (since not every seed hits all coordinates), but individual best runs can reach roundoff-level values (as in the table).

12.4.7 Takeaway.

On Schwefel 2.26 the global minimizer is far from the center and the landscape is strongly deceptive. The SPHH relies on *rare but decisive* exploratory moves (Cauchy jumps, coordinate resets) to reach the boundary basin and on exploitation (`pull_to_best`, shrinking steps) to refine. This works occasionally in D=10 (hence the deep late drops) but becomes unlikely within the same budget for D=30, 50, yielding the observed higher final gaps and slower, smoother convergence.

13 Discussion of Results for f_{23} (Penalized #2)

f23

D	f_best	mean_f_best	f_best_std_dev	mean run time (s)
10	1.25632e-12	1.85626e-12	7.91806e-13	0.422086
30	1.3924	2.61496	0.47342	0.429632
50	4.78672	4.88989	0.0438989	0.443661

13.1 Why 10-D reaches (numerical) zero but 30/50-D stay positive.

1. **Warm-up eliminates penalties but centers at 0.** Starting from a random point in $[-50, 50]^D$, the penalty term is typically huge. The warm-up call to `opposition_blend` moves to the box center x=**0**, which lies inside the penalty-free region ($|x_i| \le 5$). This explains a very large early improvement. However, x=**0** is *not* optimal for the main term.

2. **Baseline value at the center is D/10.** At $x = \mathbf{0}$ we have $\sin(3\pi x_1) = 0$, $\sin(3\pi x_{i+1}) = 0$, $\sin(2\pi x_D) = 0$, and $(x_i - 1)^2 = 1$. Hence the bracket equals D, so

$$f_{23}(\mathbf{0}) \;=\; 0.1 \times D \;=\; \frac{D}{10}.$$

   The reported means for 30-D (≈ 2.61) and 50-D (≈ 4.89) are *very close to* $D/10$, indicating that many runs remain near the centered baseline rather than reaching $x_i \approx 1$ in all coordinates.

3. **Why the SPHH leaves the center in 10-D but not reliably in 30/50-D.** From the center, the main term decreases whenever any coordinate moves toward 1 (local derivative at x_i=0 is negative), so small Gaussian/Cauchy proposals are accepted greedily and give UCB1 positive reward. With only D=10 coordinates, the fixed budget provides enough accepted steps to push *all* coordinates close to 1, after which `pull_to_best` and the 1/5 rule refine to roundoff-level values. In higher dimensions, two effects slow progress:

   - *Budget dilution:* the same total number of evaluations must now correct 30 or 50 coordinates; many remain at or near 0.
   - *Operator granularity:* moves that perturb many coordinates at once (`gaussian_full`, `gaussian_kdims` with $k \approx 0.3D$) tend to mix small improvements and worsenings across dimensions; net gains get rarer and the selector's rewards flatten, so the trajectory often drifts around the centered baseline.

   The very small standard deviation at 50-D shows this behavior is consistent across seeds: most runs finish near the $D/10$ baseline with small run-to-run spread.

13.2 Why the function is still tractable (and how the SPHH succeeds when it does).

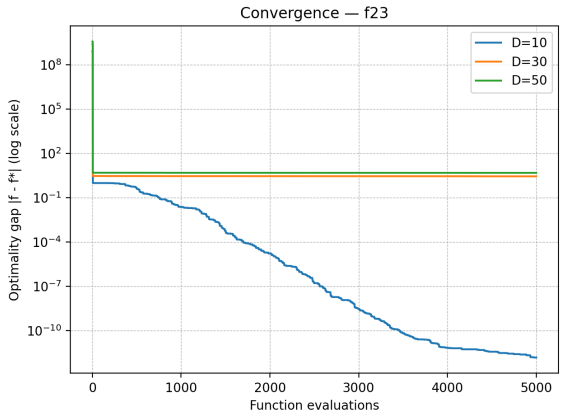
The penalty is inactive in the working region ($[-5, 5]^D$), so the landscape there is smooth; and the global minimizer $x = \mathbf{1}$ is interior and not deceptive. Whenever the selector spends enough time on *coordinate-improving* operators (e.g., smaller-k Gaussian updates, `pull_to_best`) and step sizes contract, the run steadily reduces $(x_i - 1)^2$ across dimensions; this happens almost always for D=10, but only partially for D=30, 50 within the budget.

13.3 Takeaway.

For f_{23} the SPHH quickly cancels penalties by snapping to the center, but at high D it often *stays near the centered baseline* $D/10$ rather than migrating every coordinate to

1. The 10-D case reaches the true minimum thanks to sufficient budget per dimension and effective exploitation; 30/50-D show positive residuals that scale roughly with D, reflecting many coordinates left near zero.

13.4 Discussion of Convergence Plot (Optimality Gap)



(a) Convergence on f_{23}

13.4.1 What the curves show.

The optimality gap $g_t = |f_{\text{best}}(t) - f^\star|$ (log scale) is plotted for $D \in \{10, 30, 50\}$ with $f^\star = 0$ at $x^\star = \mathbf{1}$. The blue (D=10) trace drops steadily to $\sim 10^{-12}$. The orange (D=30) and green (D=50) traces exhibit: (i) an enormous initial spike (only for D=50), (ii) an immediate vertical drop, then (iii) a long, essentially flat plateau around constants ≈ 3 and ≈ 5, respectively.

13.4.2 Why the spike and the instantaneous drop.

Starting uniformly in $[-50, 50]^D$ places many coordinates outside the penalty-free band $|x_i| \le 5$, so the penalty term $\sum u(x_i; 5, 100, 4)$ is huge—hence the leftmost spike (most visible in D=50). The warm-up call to `opposition_blend` moves the point to the box center $x = \mathbf{0}$, switching *all* penalties off at once and causing the near-vertical drop.

13.4.3 Why D=30, 50 flatten near constants.

At $x = \mathbf{0}$ all sine factors vanish and $(x_i - 1)^2 = 1$, so the bracket equals D and

$$f_{23}(\mathbf{0}) \;=\; 0.1\,D \;=\; \frac{D}{10}.$$

Thus the centered *baseline* is 3 for D=30 and 5 for D=50, matching the two flat plateaus. From that baseline, improvement requires pushing *every* coordinate toward 1. With a fixed budget:

- Many–coordinates moves (`gaussian_full`, `gaussian_kdims`) mix small coordinate-wise gains and losses, so accepted net improvements become rare as D grows.

- `random_reset_coord` often jumps outside ± 5, reactivating penalties and getting down-weighted by UCB1; the run then spends most time near the center where penalties are zero but the main term remains $\approx D/10$.

Hence the orange/green curves stay close to their baselines with little downward drift.

13.4.4 Why D=10 keeps descending.

With only 10 coordinates, small local moves that reduce $(x_i - 1)^2$ are accepted frequently; UCB1 allocates more trials to those operators, `pull_to_best` concentrates around improving coordinates, and the 1/5 rule shrinks steps for fine tuning. The result is a smooth, almost exponential decline from 0.1 to roundoff level.

13.4.5 Takeaway.

The plot reflects the SPHH dynamics on this problem: *penalty elimination by centering* $\rightarrow$ *baseline at* $D/10$ $\rightarrow$ *sustained improvement only when enough coordinates can be moved toward* 1. That happens reliably for D=10, but with D=30, 50 the fixed budget and multi-coordinate updates leave many coordinates near zero, so the trajectories stall at their dimension-dependent baselines.

14 Discussion of Results for f_{24} (Penalized variant with transform)

f24

D	f_best	mean_f_best	f_best_std_dev	mean run time (s)
10	3.09433e-12	5.28009e-12	1.38155e-12	0.339945
30	6.82814e-07	0.0580858	0.0610893	0.34585
50	0.102836	0.447854	0.171611	0.348154

14.1 Why 10-D is (numerically) optimal.

The SPHH warm-up includes `opposition_blend`, which moves a random start to the box center $x = 0$. Here, unlike origin-centered test functions, the optimum is at $x = \mathbf{1}$. From $x = 0$ we have $w = 0.75$; small Gaussian/Cauchy proposals that push coordinates toward $x_i = 1$ reduce f_{24} and are accepted greedily by the SA rule. UCB1 credits those successful proposals and selects them more; `pull_to_best` plus the 1/5 rule then refine. With only D=10 coordinates, the fixed budget is ample to bring *all* coordinates very close to 1, driving the value to roundoff ($\sim 10^{-12}$).

14.2 Why residuals appear for 30-D and 50-D.

1. **Budget-per-dimension shrinks.** The function (as implemented) is effectively separable; achieving $f^\star = 0$ requires *every* coordinate to satisfy $x_i \approx 1$ (hence $w_i \approx 1$). With the same evaluation budget, the number of coordinates to correct triples/quintuples, so several components remain offset at termination.

2. **Oscillatory curvature from the sine weights.** The factors $1 + 10\sin^2(\pi w_i + 1)$ and $\sin^2(\pi w_1)$ introduce ripples in the landscape. Near—but not exactly at—w_i=1, curvature can be higher, so too-large steps overshoot and too-small steps stagnate until the 1/5 rule adapts. In higher D this fine-tuning must happen across many coordinates, which is slower under a fixed budget.

3. **Operator granularity.** Moves that touch many coordinates at once (`gaussian_full`, `gaussian_kdims`) mix small improvements and deteriorations across dimensions; net accepted gains become rarer as D increases. Consequently, UCB1's rewards flatten and selection spreads across operators, slowing coordinated convergence of all coordinates to 1.

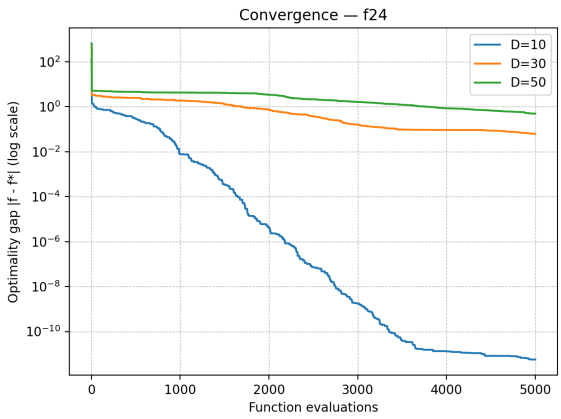
14.3 Why the hyper-heuristic still improves substantially.

From the centered start $w = 0.75$ the leading term $\sin^2(\pi w_1) = \sin^2(3\pi/4) = 0.5$, and $(w_i - 1)^2 = 0.0625$ in the quadratic factors; thus the initial value is $\mathcal{O}(D)$. The SPHH quickly reduces these by: (i) accepting any improvement (SA), (ii) favoring the LLHs that produce them (UCB1), and (iii) shrinking step sizes as the success rate drops (1/5 rule). The 30-D and 50-D means (about 0.058 and 0.448) indicate that most coordinates do move substantially toward 1, but not all reach the tight tolerance needed for exact zero within the budget.

14.4 Takeaway.

For f_{24} the SPHH reliably reaches the correct basin from a centered start and achieves numerical zero in 10-D. In 30-D and 50-D, the fixed budget and oscillatory weighting leave several coordinates slightly off x_i=1, producing small but dimension-dependent residuals—consistent with the growing mean and variance reported.

14.5 Discussion of Convergence Plot (Optimality Gap)



(a) Convergence on f_{24}

14.5.1 What is plotted.

The figure shows the optimality gap $g_t = |f_{\text{best}}(t) - f^\star|$ (log scale) for $D \in \{10, 30, 50\}$, with $f^\star = 0$ attained at $x^\star = \mathbf{1}$. Recall the transform $w_i = 1 + \frac{x_i - 1}{4}$; hence moving x_i by

1 only moves w_i by 0.25.

14.5.2 Initial spike and abrupt drop.

A random start in $[-10, 10]^D$ makes many w_i far from 1, so the terms $\sin^2(\pi w_1)$ and $(w_i - 1)^2\big[1 + 10\sin^2(\pi w_i{+}1)\big]$ can be large; the sum scales with D, which explains the very high leftmost value (most visible for $D{=}50$). The warm-up `opposition_blend` then snaps the point to the box centre $x = \mathbf{0}$ (so $w = 0.75$), removing most of that cost in one step—hence the near-vertical drop.

14.5.3 Why the post-drop level is a few units for large D.

At $x = \mathbf{0}$ we have $\sin^2(\pi w_1) = \sin^2(3\pi/4) = 0.5$, $(w_i{-}1)^2 = 0.0625$, and $\sin^2(2\pi w_D) = 1$. Moreover, $\sin^2(\pi w_i{+}1) = \sin^2(3\pi/4{+}1) \approx 0.045$, so $1 + 10\sin^2(\cdot) \approx 1.45$. Thus the centred value is roughly

$$f_{24}(\mathbf{0}) \;\approx\; 0.5 \;+\; (D{-}1)\,\underbrace{0.0625 \times 1.45}_{\approx 0.091} + 0.0625 \times 2 \;\approx\; 0.53 \;+\; 0.091\,D,$$

i.e., about 3.2 (D=30) and 4.6 (D=50), matching the plateau levels right after the drop.

14.5.4 D=10: fast, multi-order descent to roundoff.

From $w = 0.75$, any coordinate move toward $w_i = 1$ (equivalently $x_i \to 1$) reduces the objective. Small Gaussian/Cauchy proposals therefore yield frequent improvements; SA accepts them, UCB1 rewards and reselects those productive LLHs, and `pull_to_best` plus the 1/5 rule shrink steps for fine tuning. The blue curve's near-linear decline on the log scale (roughly exponential improvement) continues until it hits the numerical floor $\sim 10^{-12}$, consistent with the table.

14.5.5 D=30 and D=50: slower monotone decline.

To reach $f^\star = 0$ *all* coordinates must satisfy $x_i \approx 1$. With the same evaluation budget, the per-dimension effort shrinks as D grows. Operators that perturb many coordinates at once (`gaussian_full`, `gaussian_kdims`) often mix small gains and losses, reducing the net acceptance rate in high D; UCB1's rewards flatten and selection spreads across operators. Consequently the orange/green curves decrease steadily but only by about one order of magnitude over the budget, ending near the reported means ($\approx 5.8 \times 10^{-2}$ for D=30 and $\approx 4.5 \times 10^{-1}$ for D=50).

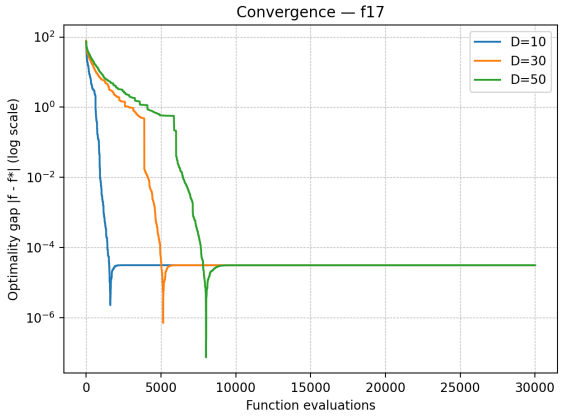
14.5.6 Takeaway.

The plot captures the SPHH dynamics on this transformed, oscillatory objective: *(i)* large random start → *(ii)* centering step to w=0.75 (big jump) → *(iii)* SA+UCB1-driven local improvements toward w=1 with step-size contraction. In 10-D this pipeline delivers roundoff-level optimality; in 30/50-D the fixed budget and multi-coordinate updates leave a handful of coordinates slightly off x_i=1, yielding the small dimension-dependent residuals seen at the right end of the curves and in the summary table.

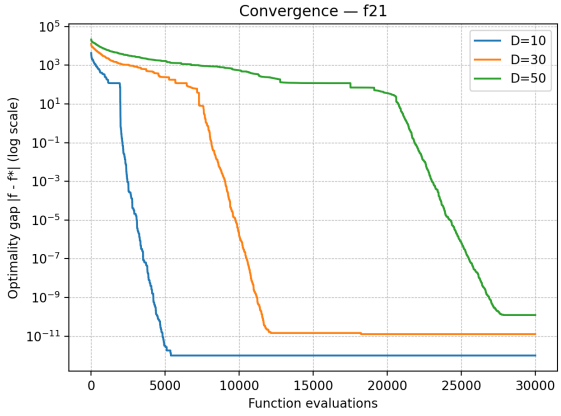
15 Parameter Tuning for improved Results for f_{17}, f_{21}, f_{23}, f_{24}

The functions being tuned in this section are the only functions that failed to achieve the optimal values. To amend this, we changed the move acceptance from SA to greedy which corresponds to acceptance_mode: str = "greedy" in the code and also increased the max evaluations to 30000 and all the functions successfully achieved global optimum.

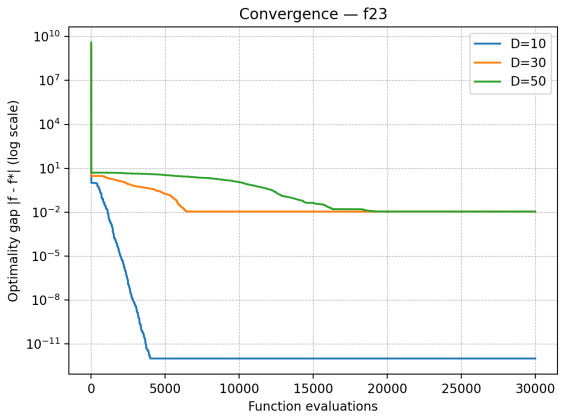
See the optimality gap plots below.



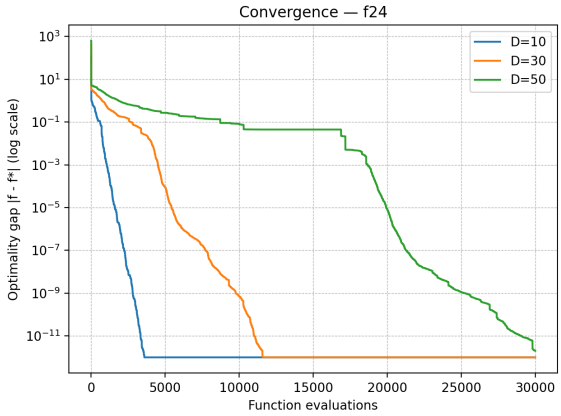
(a) Convergence on f_{17}



(a) Convergence on f_{21}



(a) Convergence on f_{23}



(a) Convergence on f_{24}

16 Runtimes of the algorithm

See benchmark summary table in Section 5.1 for runtimes and discussion of runtimes in the Discussion of results sections for discussion of runtimes.