

# Harmonomino: Tetris Agent Optimization

Using Harmony Search & Cross-Entropy Methods

Ezra Cerpac · Andrea Tomatis

Faculty of Electrical Engineering and Computer Science

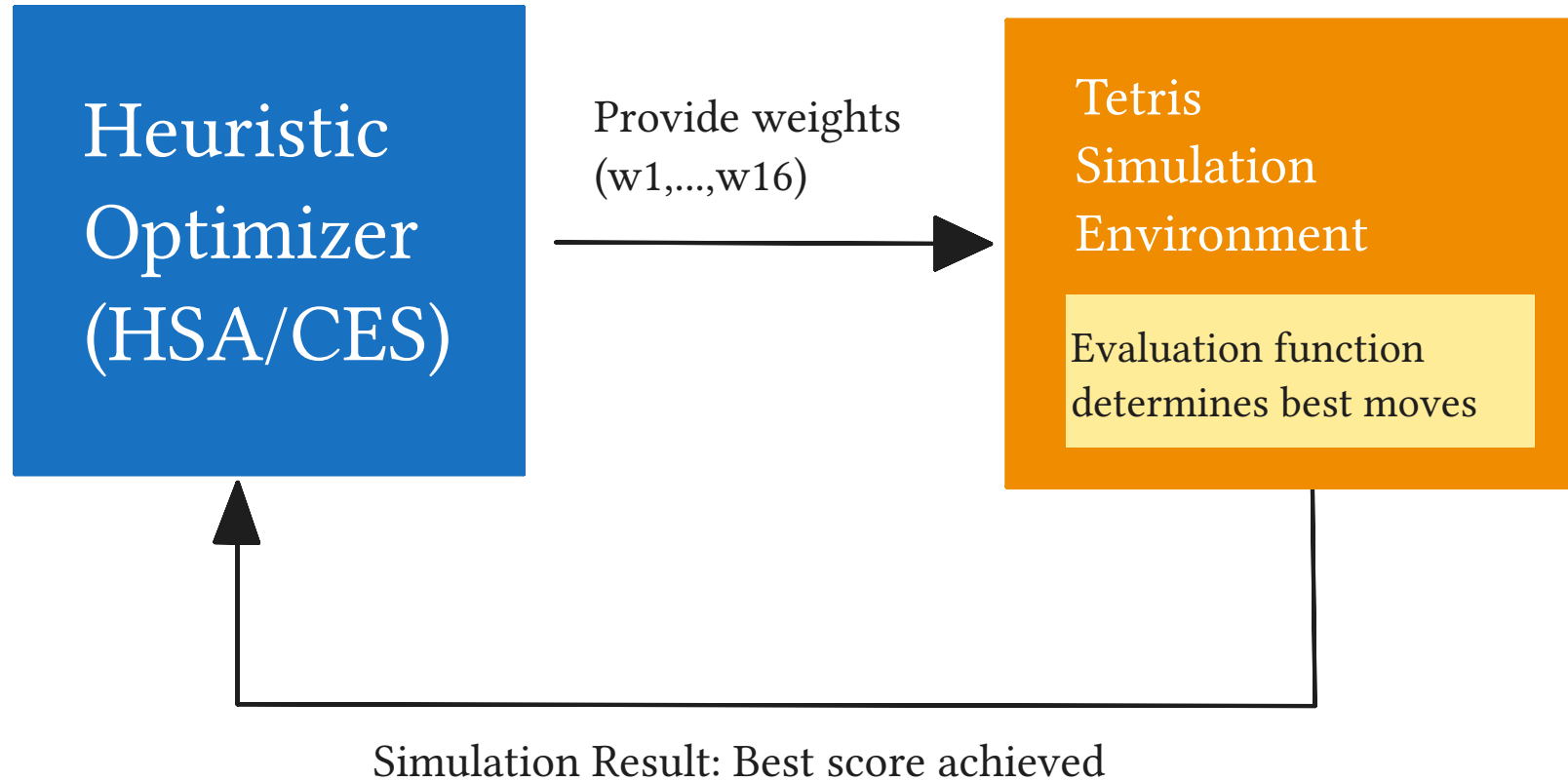
2026-02-08

# Background & Motivation

- Tetris is a subproblem in the category of tiling problems and is a well-studied AI benchmark with NP-hard piece placement
- The game is simple to understand but difficult to master, making it an ideal testbed for optimization techniques.
- Hand-tuning evaluation weights is infeasible for 16 features
- Metaheuristic search offers a practical alternative to exhaustive optimization

- Tetris is a subproblem in the category of tiling problems and is a well-studied AI benchmark with NP-hard piece placement
- The game is simple to understand but difficult to master, making it an ideal testbed for optimization techniques.
- Hand-tuning evaluation weights is infeasible for 16 features
- Metaheuristic search offers a practical alternative to exhaustive optimization
- **Goal:** automatically discover weight vectors that maximize rows cleared

- **RQ1** — How effective is Harmony Search (HSA) at optimizing Tetris agent weights?
- **RQ2** — How does Cross-Entropy Search (CES) compare to HSA?
- **RQ3** — Do the optimized weights converge to stable values?



The agent that plays Tetris consists of the following components: Board  $\rightarrow$  16 evaluation functions  $\rightarrow$  weighted sum  $\rightarrow$  best placement

Mathematically the objective function to maximize with respect to the weight vector  $w$  can be expressed as:

$$V(s) = \sum_{i=1}^{16} w_i \cdot f_i(s)$$

where every evaluation function  $f_i$  maps a board state  $s$  to an integer value.

ID	Evaluation Function
ef01	Pile Height
ef02	Holes
ef03	Connected Holes

ef05	Altitude Diff
ef06	Max Well Depth
ef07	Sum of Wells
ef09	Blocks
ef10	Weighted Blocks
ef11	Row Transitions
ef12	Col Transitions
ef13	Highest Hole
ef14	Blocks Above Highest
ef15	Potential Rows

ef16	Smoothness
ef18	Row Holes
ef19	Hole Depth

# ***Approach & Results***

## Harmony Search (HSA)

- Population of 5 weight vectors
- 100 improvisation rounds
- HMCR: 0.95, PAR: 0.99
- Bandwidth: 0.1

## Cross-Entropy Search (CES)

- 50 candidates per generation
- Top 10 elite selection
- 100 generations
- Gaussian sampling + shrinkage

## Harmony Search (HS) Algorithm

Introduced by Geem et al. (2001), HS is a metaheuristic inspired by the **musical improvisation process**.

**Musicians seek “pleasing harmony” through three strategies:**

1. **Memory:** Playing a known piece from memory.
2. **Variation:** Playing something similar with slight adjustments.
3. **Randomness:** Composing freely from random notes.

The algorithm maintains a **Harmony Memory (HM)** containing a population of solution vectors.

Mechanism	Description
HM Consideration	Copying a value from HM with probability $r_{\text{accept}}$
Pitch Adjustment	Perturbing a value with probability $r_{\text{pa}}$
Randomization	Sampling a completely new random value

> **Key Advantage:** Unlike Genetic Algorithms (which use two parents), HS considers **all** solutions in the HM simultaneously.

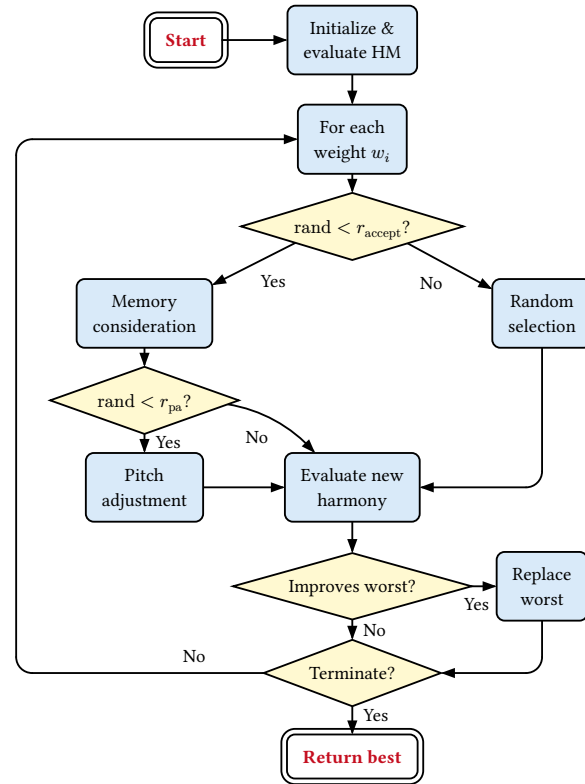
Romero et al. (2011) pioneered the use of HS for Tetris:

Feature Set: **16 board feature functions**. **Configuration:** Harmony Memory size of 5.

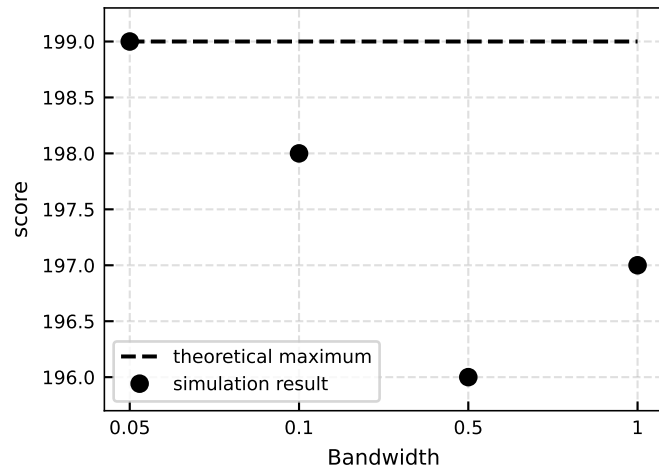
Results: - Efficiently discovered high-quality weight configurations.

- Achieved a spawned-pieces-to-cleared-rows ratio near the **theoretical optimum of 2.5**.

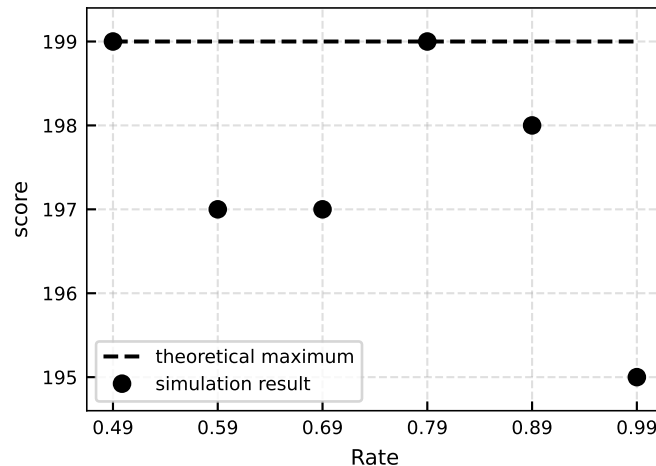




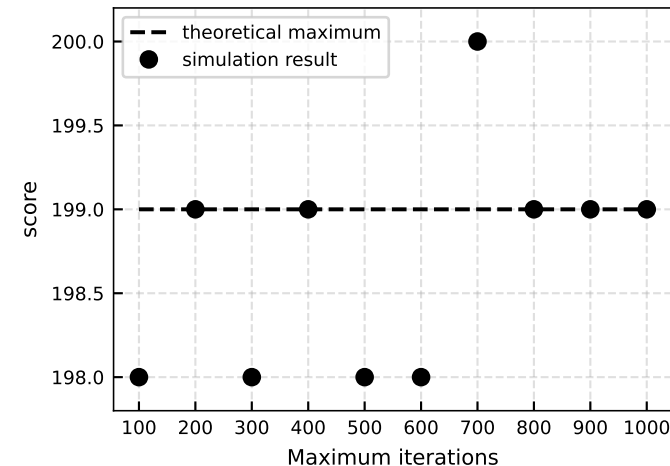
Analysis of Bandwidth, Pitch Adjustment Rate ( $r_{pa}$ ), and Max Iterations.



**Bandwidth:** Excessive values disrupt fine-tuning; very small values risk local optima.



**Pitch Adj. Rate:** Shows minimal impact on final performance in this configuration.



**Max Iterations:** Clear diminishing returns observed beyond roughly 170 iterations.

**Note:** While relative performance is stable, these parameters primarily influence the **rate** of convergence and search robustness.

Are the agents finding the same solution, or many different ones?

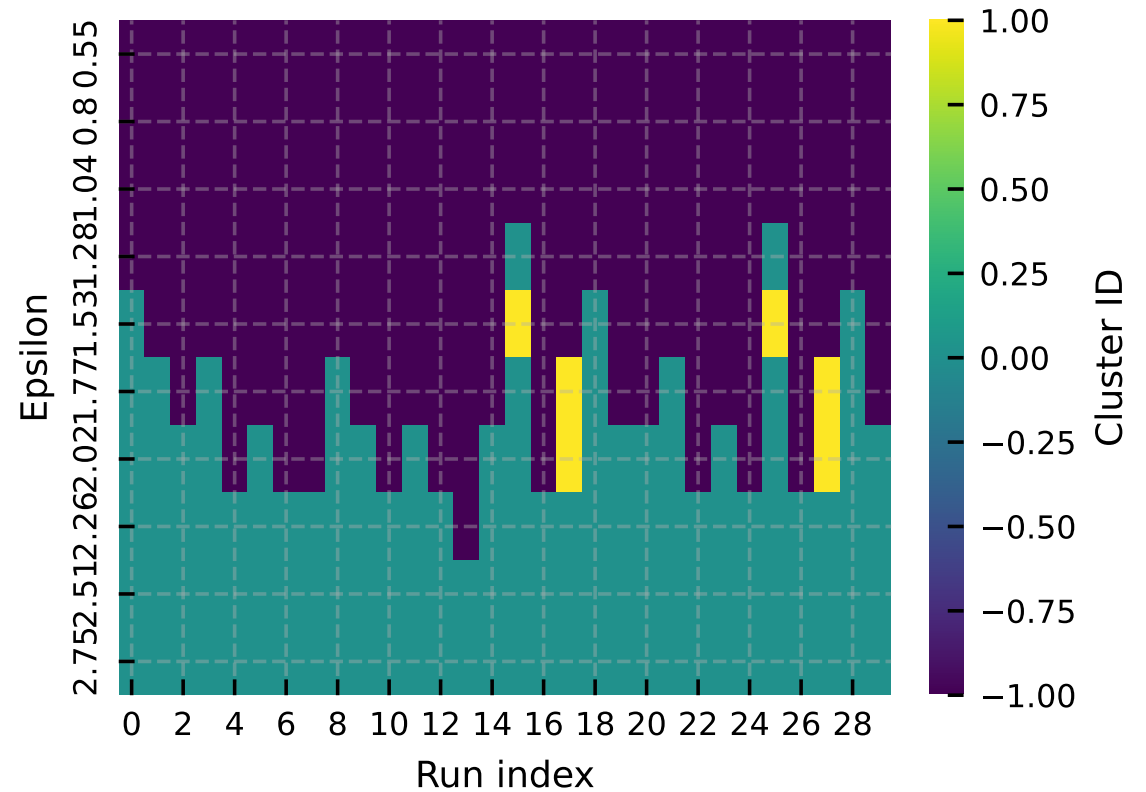
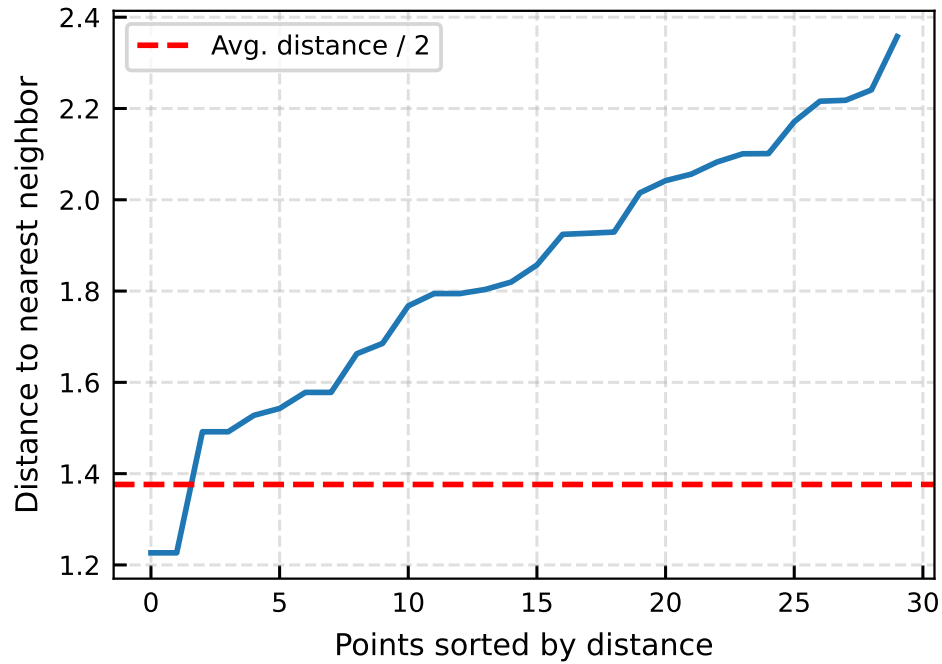


Figure 5: DBSCAN identifies a single primary cluster of “good” solutions.

- **The “Elbow”:** Identified at  $\varepsilon \approx 1.35$ .

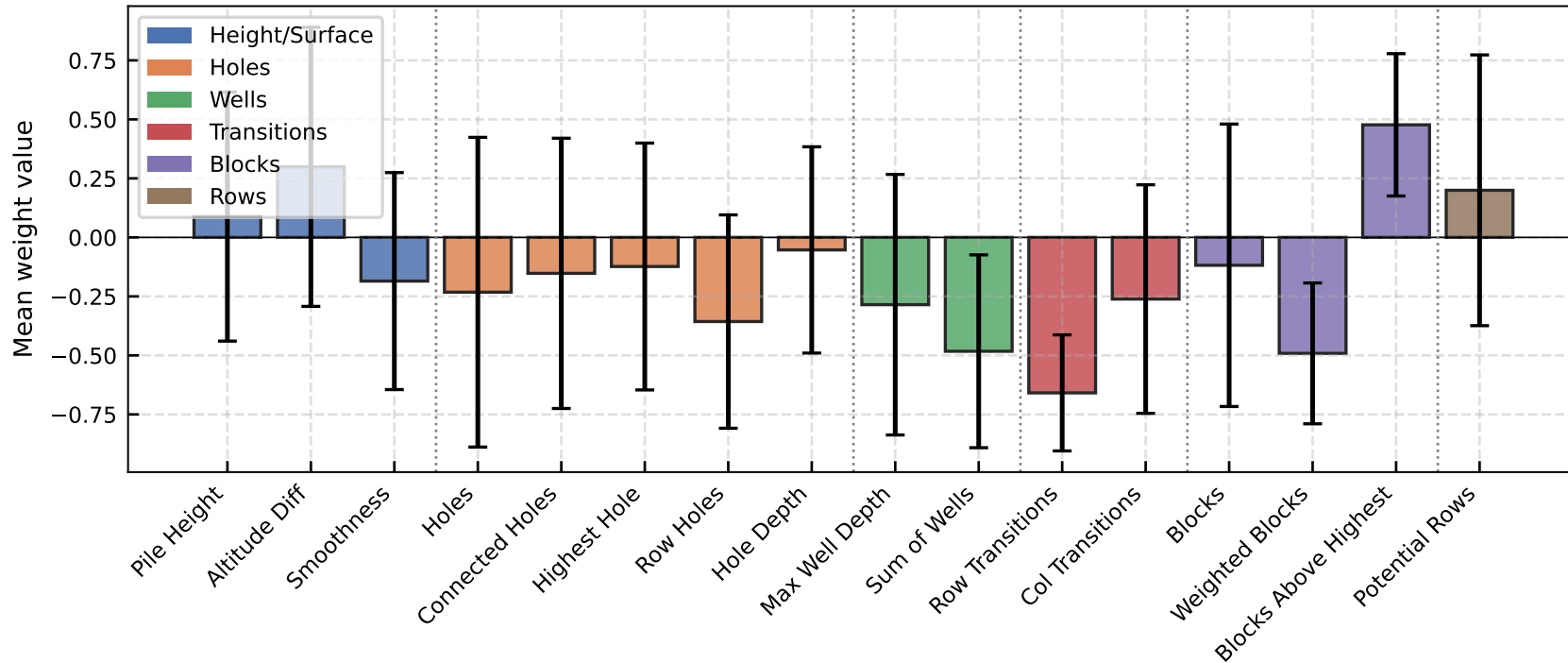


Figure 6: Most likely category found with DBSCAN per weight.

- **Primary Cluster:** Most seeds converge to a similar region in the high-dimensional weight space, validating the robustness of the heuristic set.

As games scale, the “Absolute Error” relative to the theoretical maximum increases.

## The Plateau Effect:

- **Short games (< 500 lines):** Error remains near zero.
- **Long games (> 750 lines):** Error grows sharply, exceeding 1750 by line 5000.

**Conclusion:** Current linear heuristics cannot fully compensate for board “exhaustion” in long-horizon play.

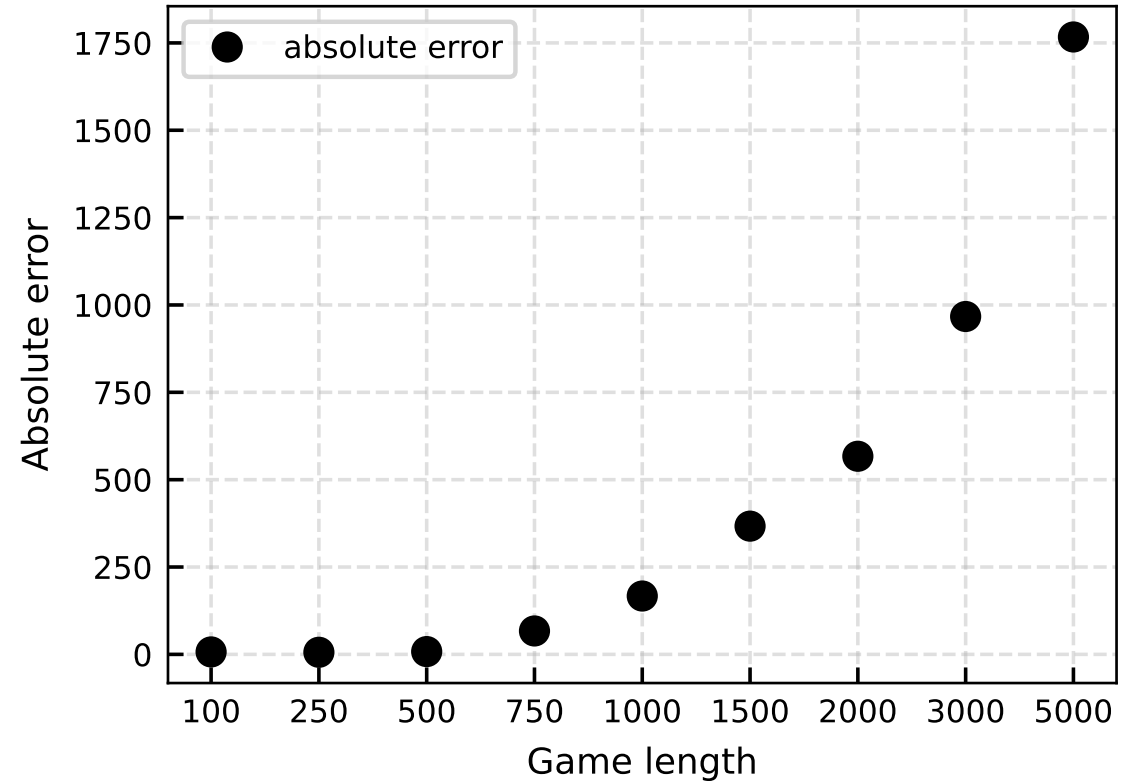
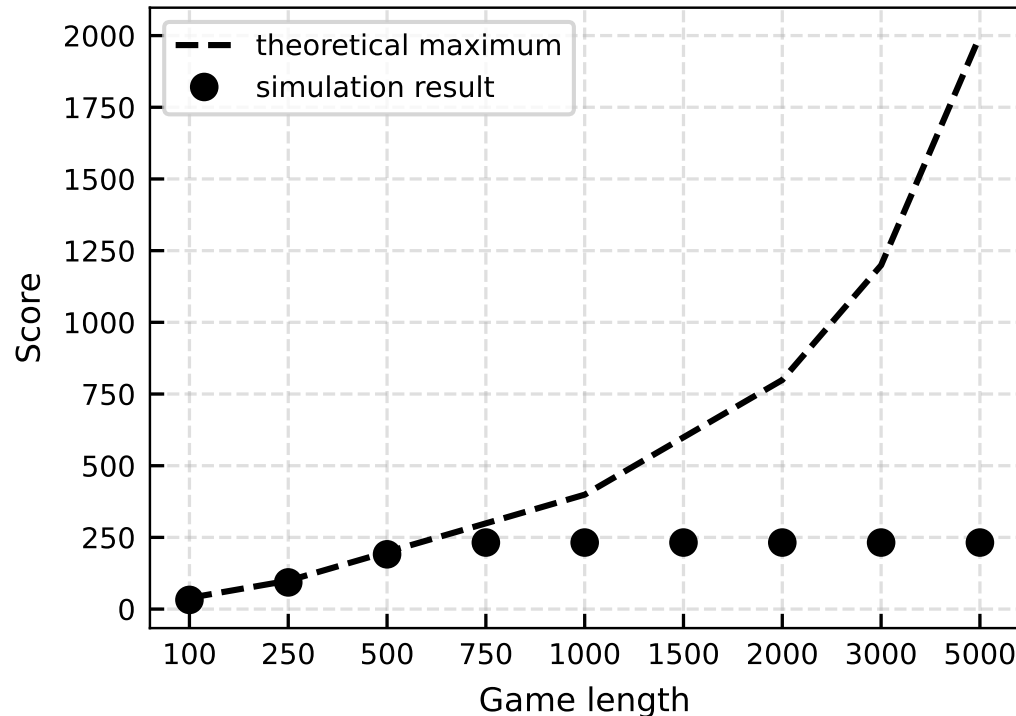


Figure 7: Absolute error vs. game length.

We tested the agent against a theoretical performance model to check for “plateaus.”



## Findings:

- **Short Term:** Near-perfect alignment with theory up to 500 rows.
- **Long Term:** Error grows significantly after 750 rows.
- **Implication:** The agents encounter “unsolvable” board states or structural constraints not captured by the simple linear heuristic model.

Figure 8: Performance vs. Theoretical Max

Unlike HS, which tracks individual “members”, CES tracks a **probability distribution** (mean  $\mu$  and variance  $\sigma^2$ ) that represents where the best weights likely live.

1. **Sampling:** Generate a large batch of candidate weight vectors (e.g., 100) by sampling from the current Gaussian distribution.
2. **The “Elite” Selection:** Test every candidate in a Tetris simulation. Select the top 10% (the “Elite Set”).
3. **Distribution Shift:** Calculate the new  $\mu$  and  $\sigma^2$  based **only** on the Elite Set.

The distribution literally “moves” and “shrinks” toward the highest-scoring regions of the fitness landscape over multiple generations.

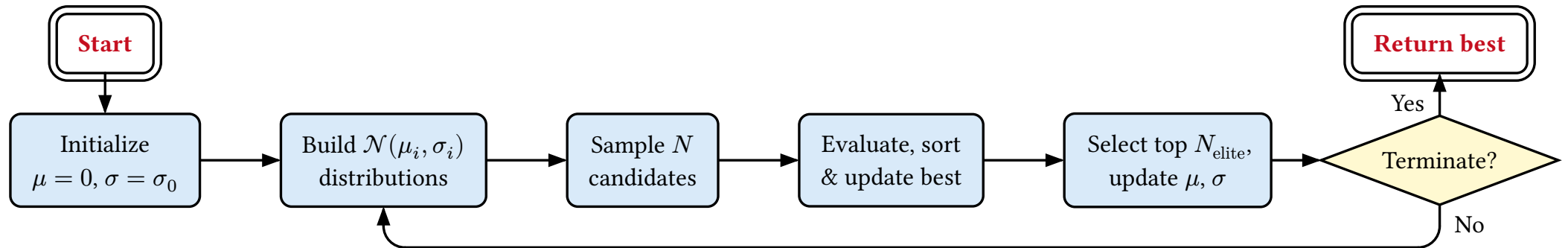
A major challenge in Tetris is the **stochastic noise**—a weight might perform well just because it got “lucky” pieces. This often leads to **Variance Collapse**.

The Failure: **The variance ( $\sigma^2$ ) shrinks to zero too quickly. The algorithm becomes “blind” to other possibilities and stops exploring. The Fix: Additive Noise.** We manually inject noise into the update rule:

$$\sigma_{t+1}^2 = \sigma_{\text{elite}}^2 + Z(t)$$

The Benefit: By ensuring the standard deviation never drops below a certain threshold, the search is forced to remain wide enough to find general, robust weights rather than “lucky” ones.

Feature	Harmony Search	Cross-Entropy
Representation	Individual vectors	Probability distribution
Improvement	Replaces worst member	Updates mean and variance
Diversity	Randomization ( $r_{\text{rand}}$ )	Additive noise ( $Z$ )
Strength	Simple, fast updates	Excellent in high dimensions

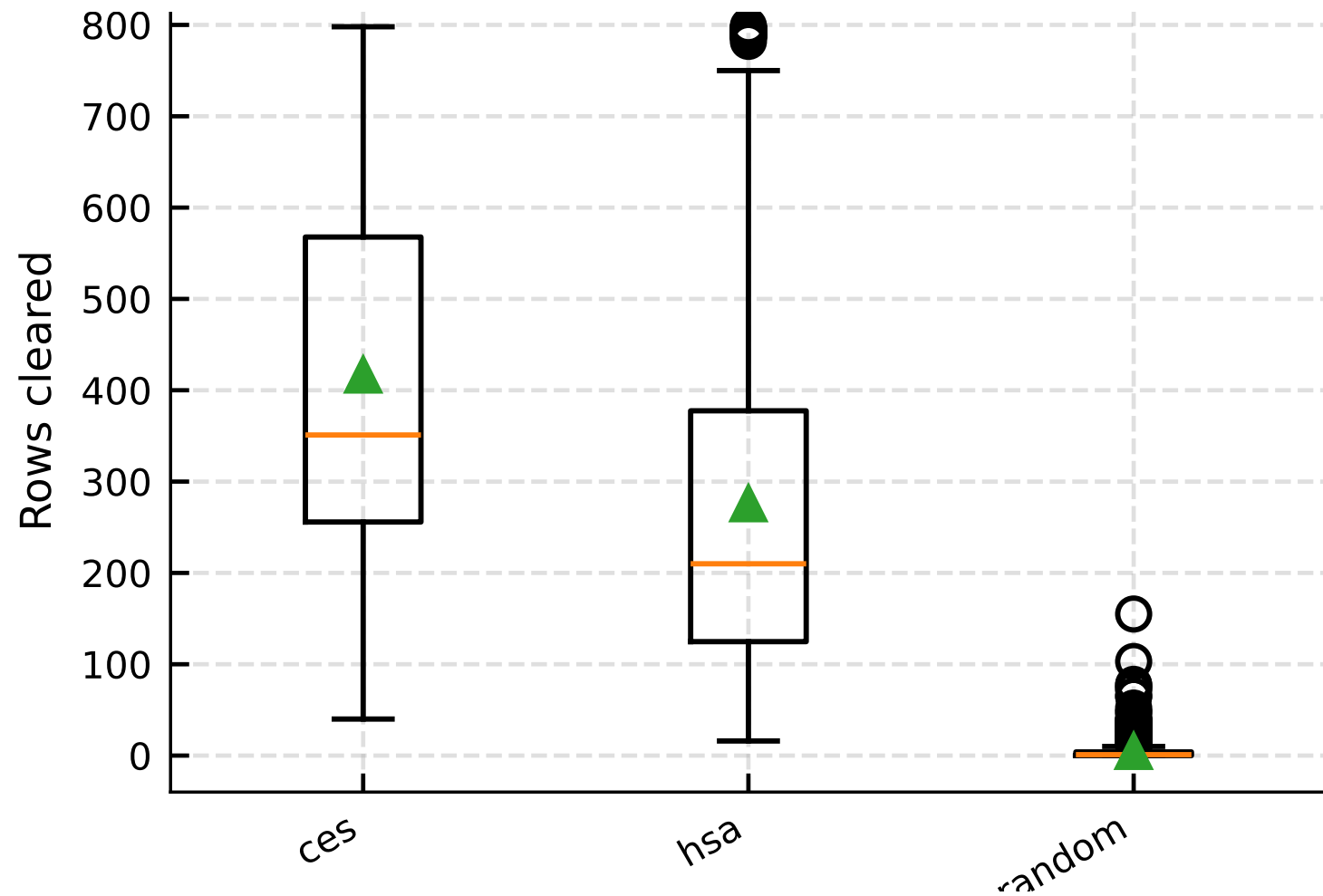


Both optimization methods significantly outperform the random baseline, with CES showing a slight edge in raw performance.

Method	Mean	Median
CES	416.633	351.000
HSA	275.737	210.000
Random	4.553	1.000

## Key Takeaways:

- CES and HSA distributions overlap significantly.
- Both maintain a high “floor”: lower quartiles exceed the best baseline results.
- Performance approaches theoretical limits for short-horizon games.



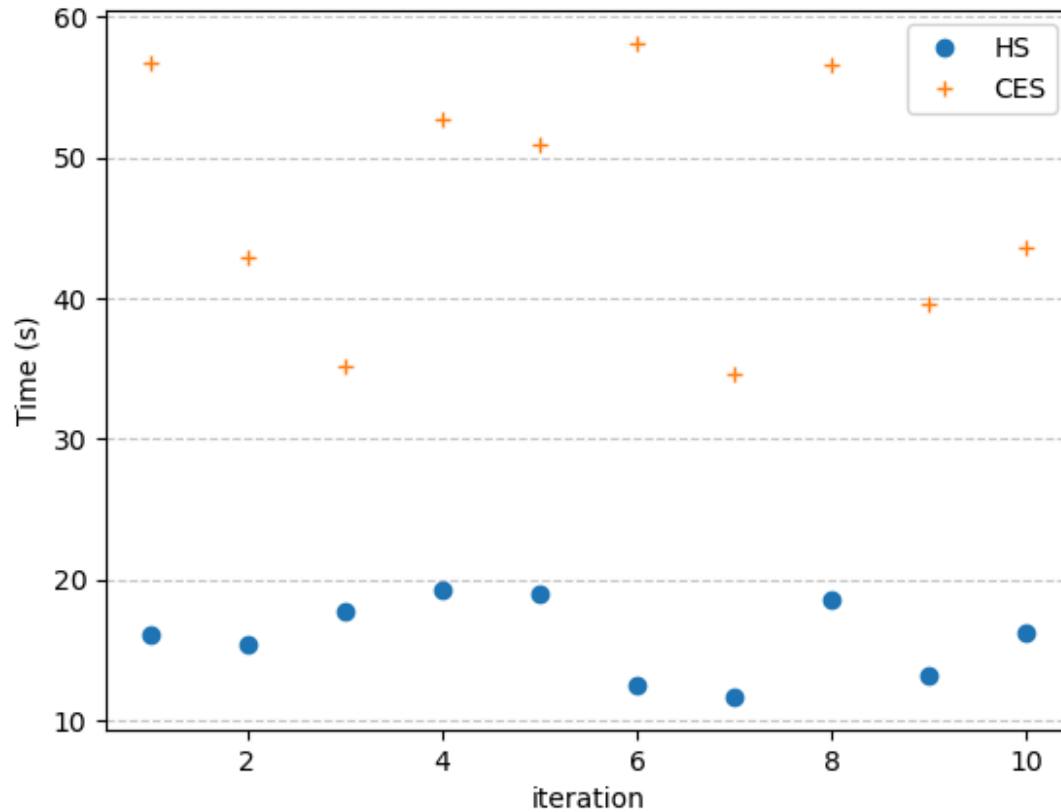


Figure 11: Execution time per iteration (seconds).

## Harmony Search (HSA):

- **Fast & Stable:** 12–19 seconds per iteration.
- Lower overhead allows for the high iteration counts (100) required for convergence.

## Cross-Entropy (CES):

- **Resource Intensive:** 35–58 seconds per iteration.
- The complexity stems from sampling and simulating large batches to update the distribution.

There is a massive disparity in how quickly each algorithm “solves” the weight space.

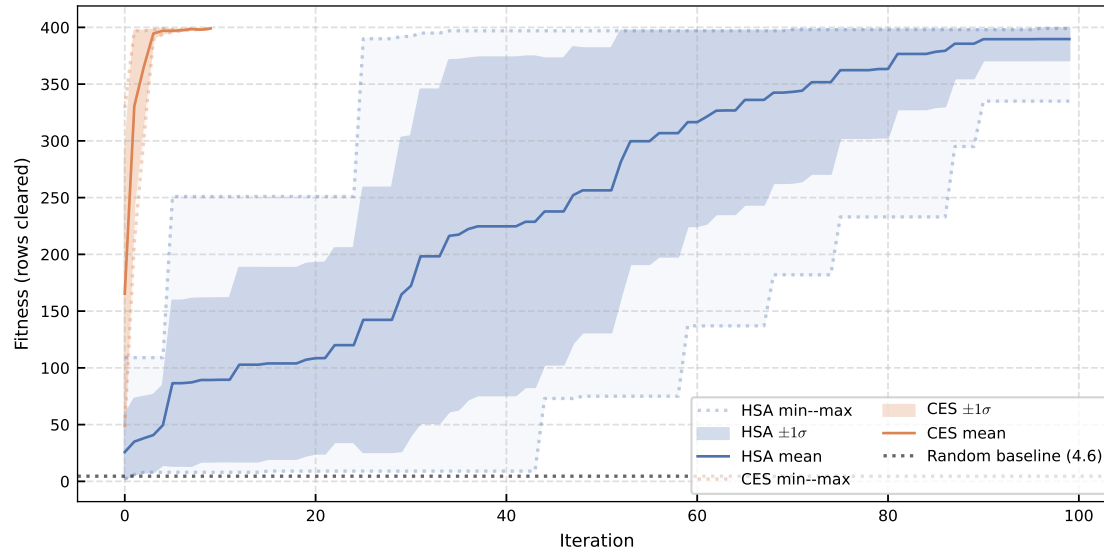


Figure 12: CES (Rapid) vs. HSA (Gradual)

## Cross-Entropy (CES):

- **Extremely Efficient:** Typically converges in  $< 5$  iterations.
- Rapidly narrows sampling distribution.
- **Trade-off:** Higher CPU cost per iteration (35–58s).

There is a massive disparity in how quickly each algorithm “solves” the weight space.

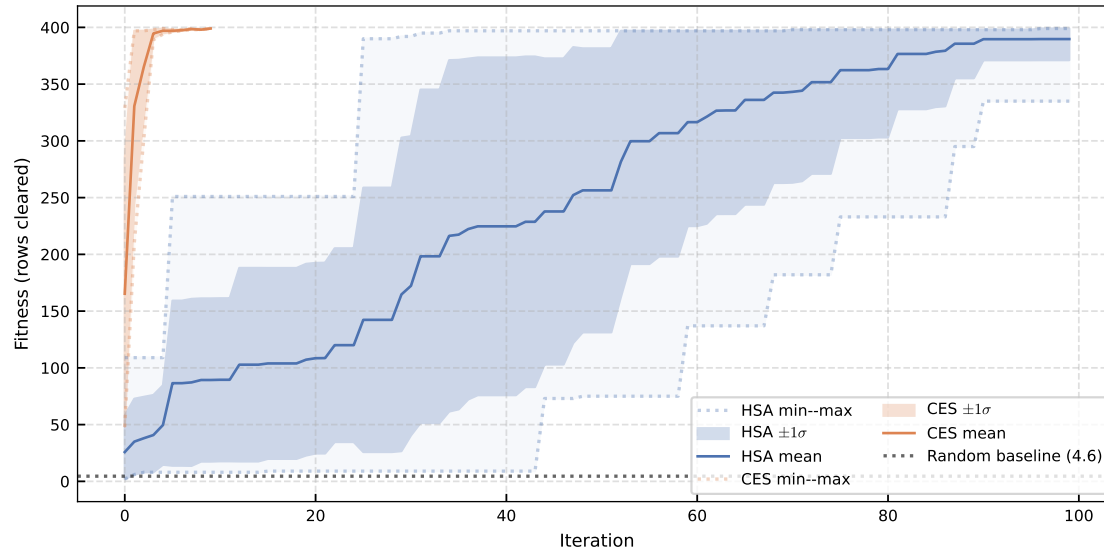


Figure 13: CES (Rapid) vs. HSA (Gradual)

## Harmony Search (HSA):

- **Steady Improvement:** Requires the full 100 budget.
- **Trade-off:** Lower CPU cost per iteration (12–19s).

Which board features actually matter for long-term survival?

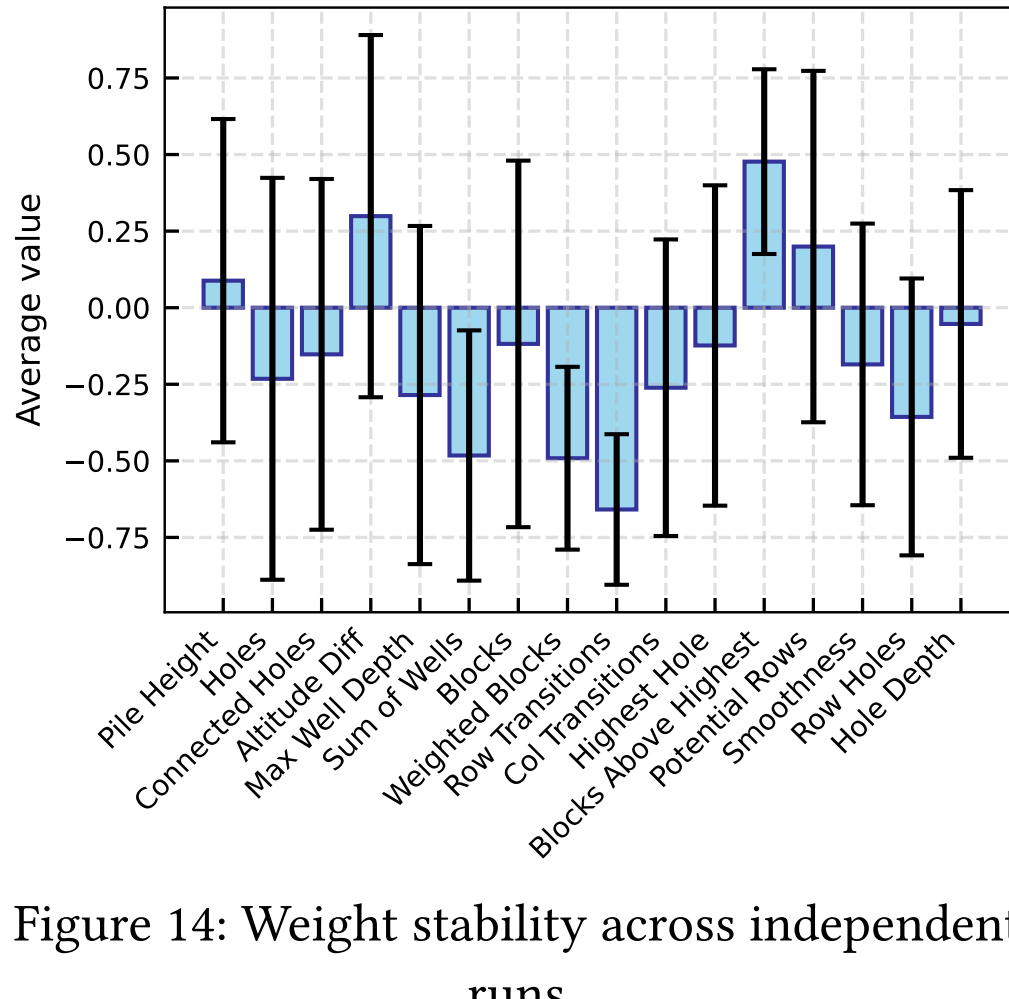


Figure 14: Weight stability across independent

## Stable Features (Low $\sigma$ ):

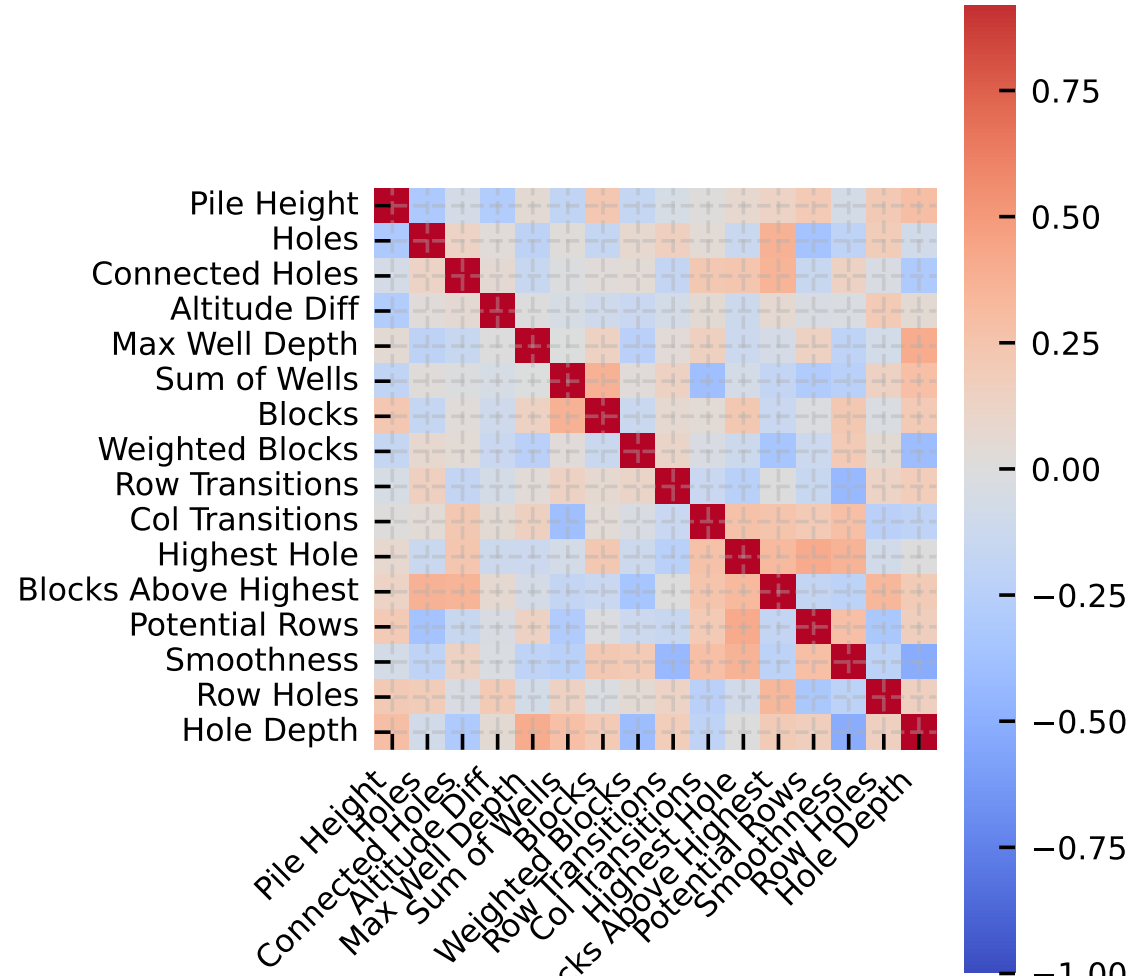
- $w_9$  (Row Transitions),  $w_8$  (Weighted Blocks), and  $w_{12}$  (Blocks Above Highest)
- These are universally critical for board quality.

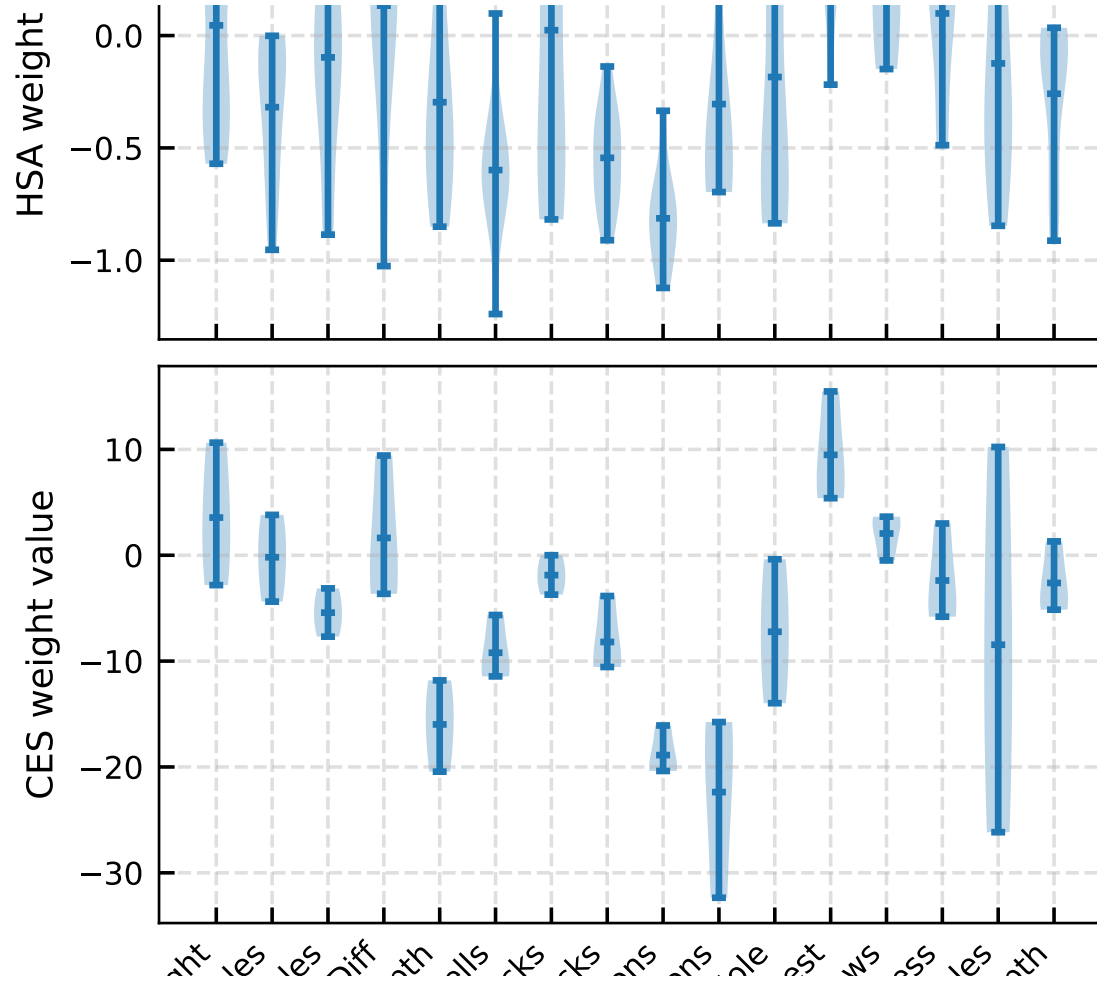
## Volatile Features (High $\sigma$ ):

- $w_2$  (Holes)
- High variance suggests the landscape has multiple “good” local optima or redundant features.

## Structure of the Solution:

- **Positive Correlation:** Height-related features (Pile Height & Blocks Above).
- **Negative Trends:** “Transition” features (Row/Col) consistently move toward negative weights to penalize surface instability.
- **Independence:** Most off-diagonal correlations are weak, validating the choice of features.





## Key Observations:

- **CES Consistency:** Generally produces tighter clusters, suggesting it finds a more precise “global” region.
- **HSA Diversity:** Wider distributions indicate HSA explores a broader range of the solution landscape.
- **Directionality:** Both agree on the polarity of key features (e.g., negative weights for transitions).

# Conclusion

## Research Questions & Findings

### RQ1: HSA

How effective is HSA at weight optimization?

**Verdict:** Suboptimal but Effective. Outperforms random weights and hand-tuned baselines significantly.

### RQ2: CES

How does CES compare to HSA?

**Verdict:** Superior Performance. Outperforms HSA in convergence speed and score, despite higher per-iteration costs.

### RQ3: Stability

Do optimized weights converge to stable values?

**Verdict:** Partial Convergence. Core weights are stable; secondary weights vary due to landscape stochasticity.

## Limitations

- Single-piece lookahead only
- Fixed game length caps observable performance
- No T-spin or hold-piece strategies

- Multi-piece lookahead and hold-piece integration
- Hybrid algorithms combining HSA exploration with CES exploitation
- Neural-network evaluation functions trained on optimized weights
- Transfer learning across board sizes and rule variants

**Thank You!**

Ezra Cerpac · Andrea Tomatis

Scientific Computing  
Technische Universität Berlin

*Questions?*