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

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

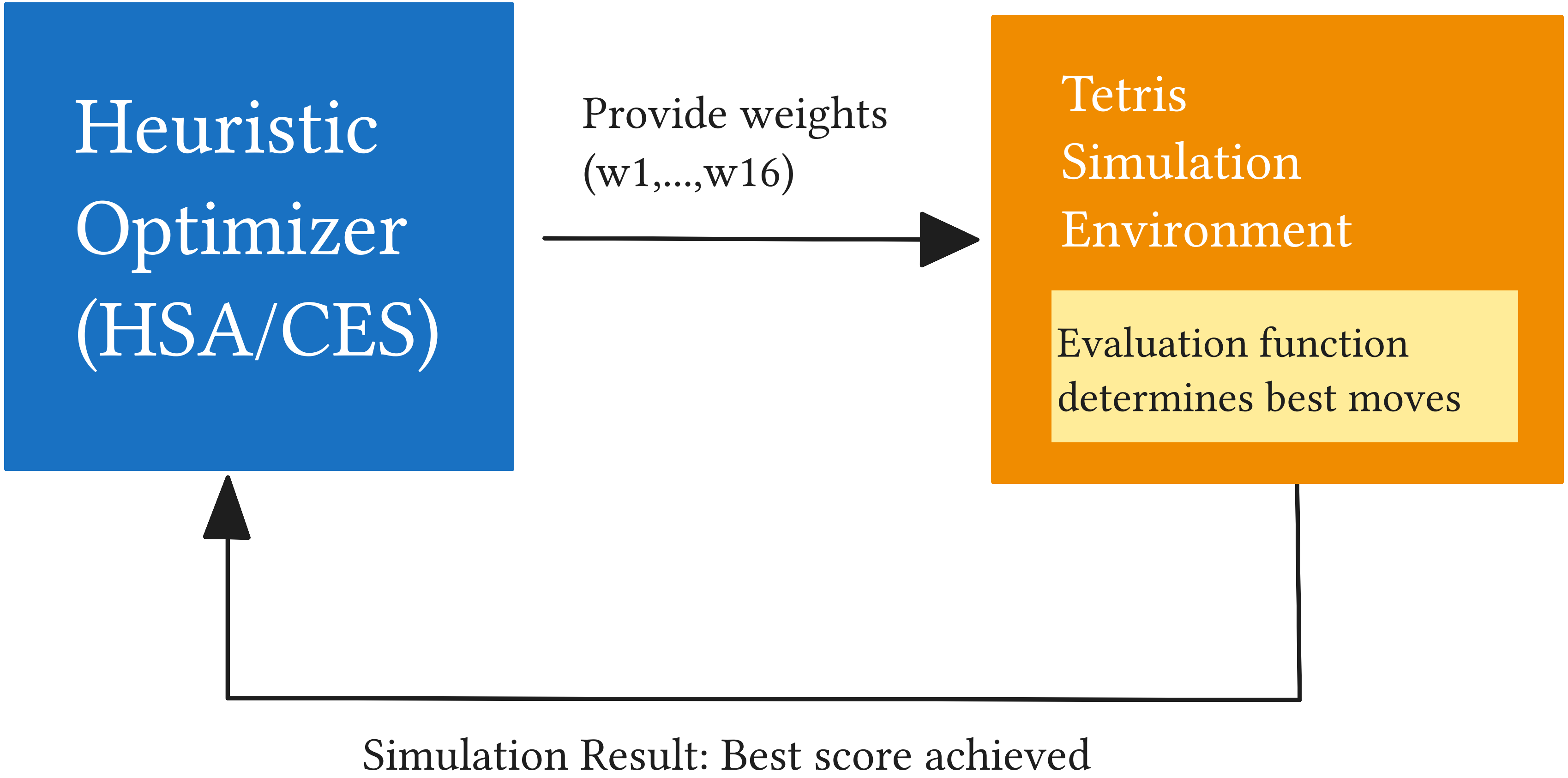
Motivation

- 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
- **Goal:** automatically discover weight vectors that maximize rows cleared

Research Questions

- **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?

Agent Architecture



Agent Architecture

The agent that plays Tetris consists of the following components: Board ⟶ 16 evaluation functions ⟶ weighted sum ⟶ 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

Agent Architecture

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

Agent Architecture

ef16	Smoothness
ef18	Row Holes
ef19	Hole Depth

Approach & Results

Algorithms

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.

Core Mechanisms of HS

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_{accept}
Pitch Adjustment	Perturbing a value with probability r_{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.

HS in Tetris Weight Optimization

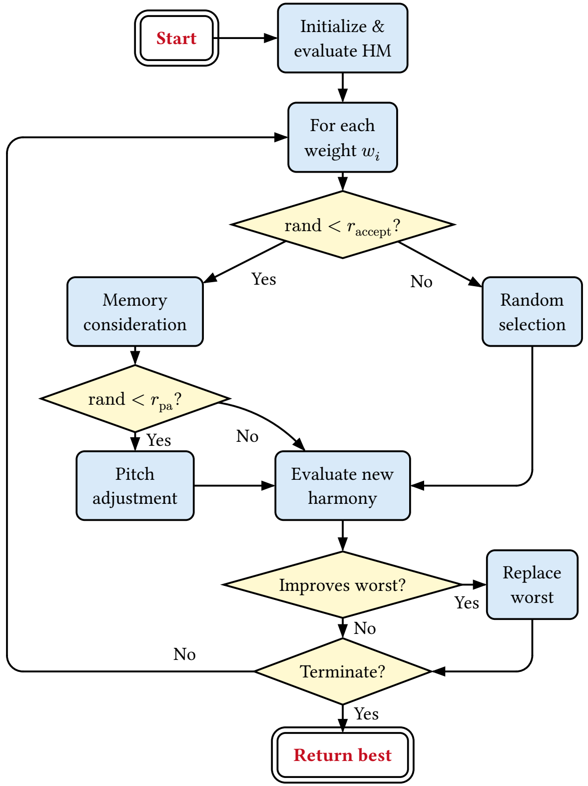
Romero et al. (2011) pioneered the use of HS for Tetris:

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

Results: - Efficiently discovered high-quality weight configurations.

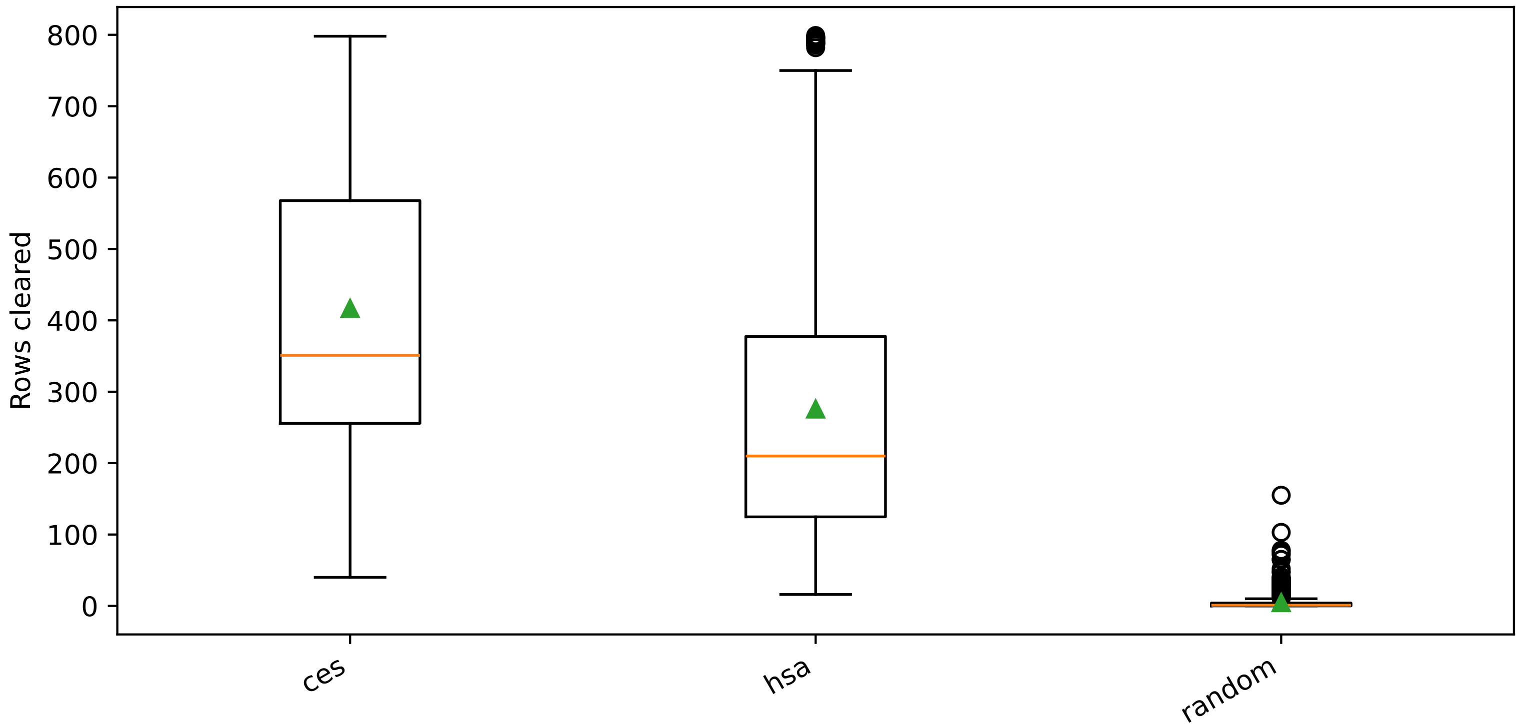
- Achieved a spawned-pieces-to-cleared-rows ratio near the **theoretical optimum of 2.5**.

Flowchart



Start
Initialize & evaluate HM
For each weight w_i
rand $< r_{\text{accept}}$?
Yes
No
Memory consideration
Random selection
rand $< r_{\text{pa}}$?
Yes
No
Pitch adjustment
Evaluate new harmony
Improves worst?
Yes
Replace worst
No
No
Terminate?
Yes
Return best

Performance



Performance

HSA

Mean: 275.737 rows

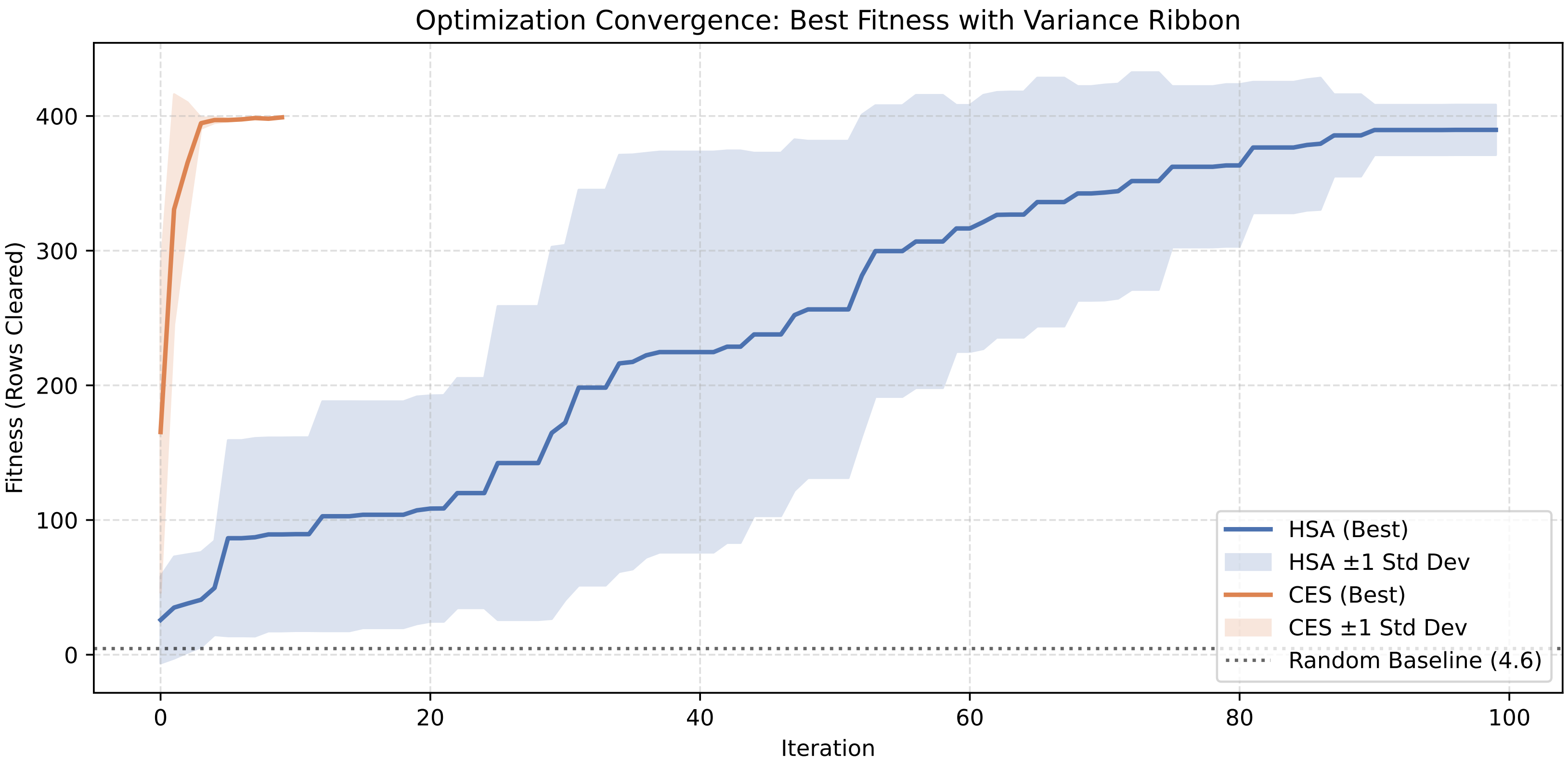
Median: 210.000 rows

CES

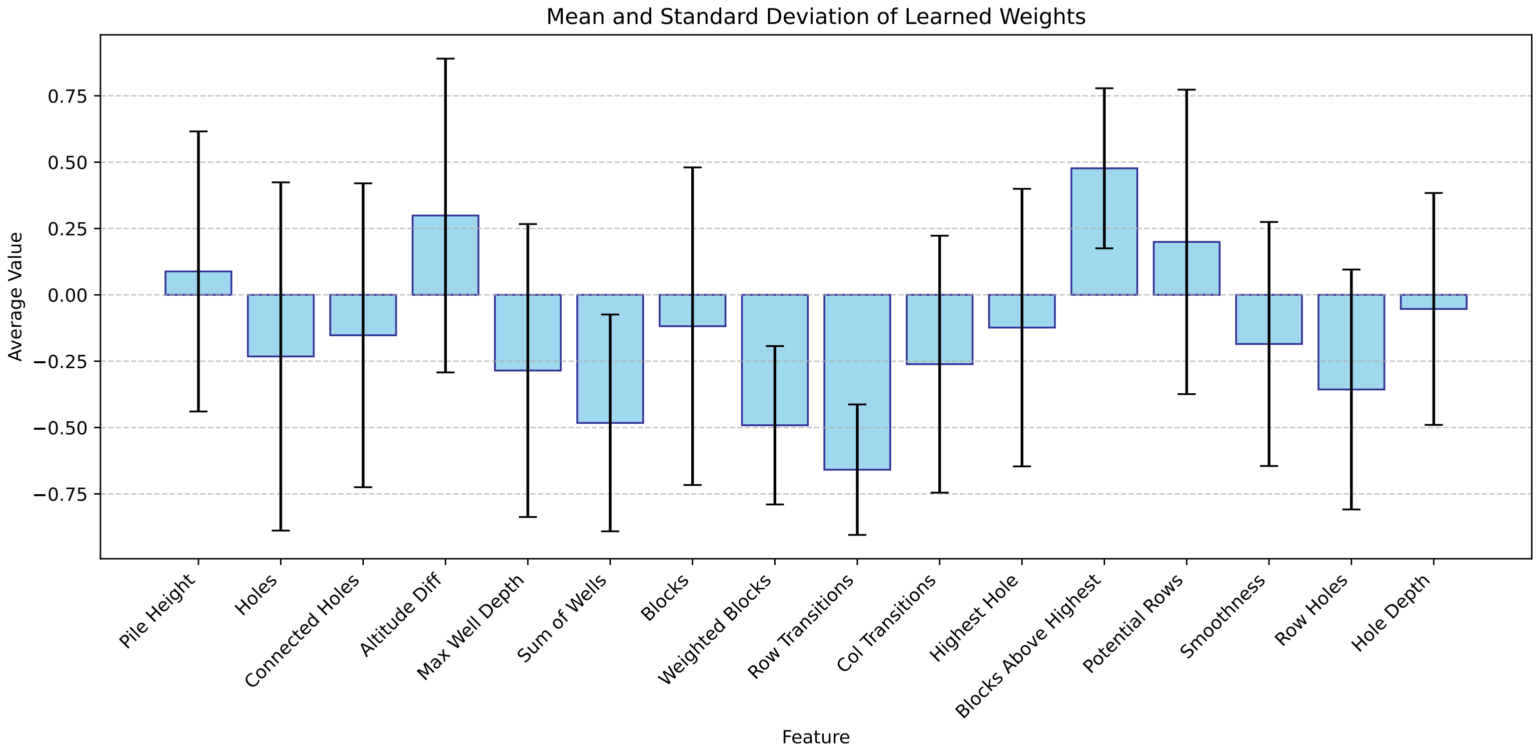
Mean: 416.633 rows

Median: 351.000 rows

Convergence



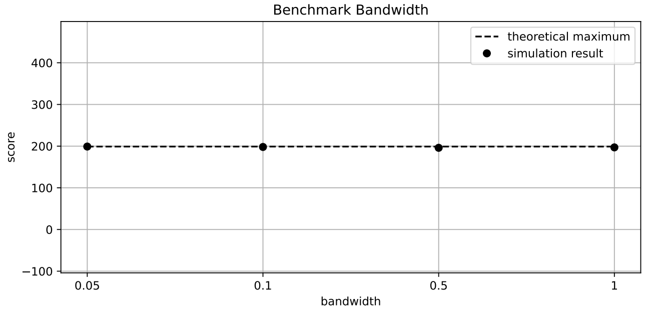
Weight Analysis

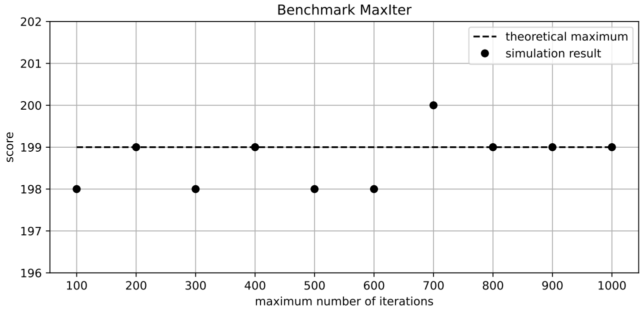


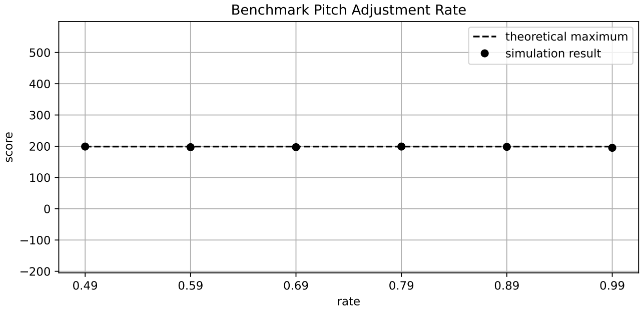
Most stable: w_9 (Row Transitions), w_8 (Weighted Blocks), and w_{12} (Blocks Above Highest)

High variance: w_2 (Holes)

Parameter Sensitivity







Conclusion

Findings

Contributions

- Both HSA and CES successfully optimize Tetris agent weights
- 16 feature evaluation covers pile, hole, well, row, and block metrics
- Subset of weights converge to stable values across runs
- Automated pipeline enables reproducible experiments

Limitations

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

Future Work

- 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?