Review

Evaluating Tetris Piece (Tetromino) Randomization Algorithms: Sequence Fairness and Gameplay Impact of Uniform RNG vs 7-Bag Systems

Soponloe Sovann *

Posted Date: 29 January 2026

doi: 10.20944/preprints202601.2206.v1

Keywords: drought; randomization; fairness perception; game design; tetris; shuffle without replacement; statistical testing

Review

Evaluating Tetris Piece (Tetromino) Randomization Algorithms: Sequence Fairness and Gameplay Impact of Uniform RNG vs 7-Bag Systems

Soponloe Sovann

CS427 Selected Topic; sovannsoponloe@gmail.com

Abstract

Modern versions of the game use the "7-bag" randomization technique to soften the sense of bias inherent in extreme pieces "droughts," where a required tetromino is not generated for a long time. This study compares uniform random number generation (RNG) and 7-bag randomization for two experiments. In the first experiment, we inspected 100,000 produced numbers to severity and homogeneity of distribution. The findings revealed that 7-bag randomization resulted in decreases in maximum Ipiece droughts of 71outcomes. Both techniques maintained the fixed frequencies of pieces in the long run. In the second test, the effect of the game via controlled single-player experiments. Games employing 7-bag randomness raised the average number of cleared squares by 14fewer lines and slightly lower variance, although this was not found to be statistically significant because of the small data set size. In general, the results do prove the quantitative effectiveness of 7-bag randomization in new designs for Tetris.

Keywords: drought; randomization; fairness perception; game design; tetris; shuffle without replacement; statistical testing

---

1. Introduction

Randomness is an essential element in game design. On one hand, randomness in-game design affects the level of difficulty and the potential to replay the game. On the other hand, it is essential to introduce randomness in the right manner because although randomness in the form of mathematically correct randomness translates to evenly distributed results in the long run, human players' responses to these randomness-generated data might not be as expected because research studies have shown the *clustering illusion*. [2,3].

Tetris, originally made in 1984, exemplifies this challenge. The core mechanic of the game, the sequential placement of randomly generated tetrominoes, makes it highly sensitive to the statistical properties of its randomization algorithm. Early versions of *Tetris* used uniform random number generation, selecting each tetromino independently with equal probability. While unbiased in expectation, this approach permits arbitrarily long absences of a given piece, particularly the I-piece, which is strategically valuable for clearing lines.

To address these concerns, modern *Tetris* implementations follow the *Tetris Guideline*, which mandates the use of the 7-bag randomization system [1]. Under this system, a shuffled bag containing exactly one of each tetromino is generated, and pieces are drawn sequentially until the bag is empty, after which a new bag is shuffled. This shuffle-without-replacement approach preserves long-term uniformity while constraining short-term variance [4].

Despite the widespread usage, only a few studies have been conducted to empirically analyze the statistical gameplay effects of 7-bag randomization. Practically all research concerning *Tetris*, including complexity analysis and usage in AI games, have been carried out in the realm of complexity analysis or games played with the aid of artificial intelligence [5,6], whereas all studies concerning randomization mechanisms made so far remained on a purely theoretical level.

 doi:10.20944/preprints202601.2206.v1

2. Methods

2.1. Randomization Algorithms

Uniform RNG. Each tetromino is independently sampled from the set $\{I, J, L, O, S, T, Z\}$ with equal probability:

$$\text{next\_piece} \sim \text{Uniform}(\{I, J, L, O, S, T, Z\}). \tag{1}$$

7-Bag Randomization. A randomly shuffled bag containing one instance of each tetromino is generated. Pieces are drawn sequentially until the bag is empty, after which a new shuffled bag is created:

$$\text{bag} \leftarrow \text{shuffle}(\{I, J, L, O, S, T, Z\}). \tag{2}$$

This guarantees that the maximum absence of any piece is bounded at 12.

Figure 1 shows an excerpt of the Python implementation used to realize both randomization algorithms, including deterministic seeding and piece generation logic.

```python
import numpy as np

class UniformRNG:
    def __init__(self, seed=None):
        self.rng = np.random.RandomState(seed)
        self.pieces = ['I', 'J', 'L', 'O', 'S', 'T', 'Z']

    def generate(self, n):
        return [self.rng.choice(self.pieces) for _ in range(n)]

class SevenBagRNG:
    def __init__(self, seed=None):
        self.rng = np.random.RandomState(seed)
        self.pieces = ['I', 'J', 'L', 'O', 'S', 'T', 'Z']
        self.bag = []
        self.refill()

    def refill(self):
        self.bag = list(self.rng.permutation(self.pieces))

    def next(self):
        if not self.bag:
            self.refill()
        return self.bag.pop(0)

    def generate(self, n):
        return [self.next() for _ in range(n)]

```

Figure 1. Excerpt of the Python implementation of the tetromino randomization algorithms. The uniform RNG generator samples pieces independently with equal probability, while the 7-bag generator enforces shuffle-without-replacement by refilling a randomized bag after every seven pieces. Deterministic seeding is used to ensure reproducibility.

2.2. Experiment 1: Sequence Fairness Analysis

For each algorithm, 20 independent runs of 5,000 pieces were generated, totaling 100,000 pieces per condition. Deterministic seeds ensured reproducibility. Metrics included maximum I-piece drought, 99th-percentile drought, and chi-square goodness-of-fit testing against the expected uniform distribution.

Figure 2 shows an excerpt of the implementation used to generate tetromino sequences and compute drought statistics for Experiment 1.

 doi:10.20944/preprints202601.2206.v1


```python
import csv
from generators import UniformRNG, SevenBagRNG
from analysis import droughts, stats, chi2

with open('exp1.csv', 'w', newline='') as f:
    w = csv.writer(f)
    w.writerow(['algo', 'run', 'max_i', 'p99_i', 'chi2_p'])

    for algo, RNG in [('uniform', UniformRNG), ('7bag', SevenBagRNG)]:
        for run in range(20):
            seq = RNG(seed=42+run).generate(5000)
            s = stats(droughts(seq, 'I'))
            c = chi2(seq)
            w.writerow([algo, run, s['max'], s['p99'], c['p']])

print("Done: exp1.csv")
```

Figure 2. Excerpt of the sequence generation and drought analysis code used in Experiment 1. The code applies deterministic random seeds, generates fixed-length tetromino sequences, and records I-piece drought lengths for statistical analysis.

2.3. Experiment 2: Gameplay Trials

A single competent but non-expert player completed 20 games using an online guideline-compliant implementation [7]. Ten games were played per algorithm under fixed difficulty and identical rules. The primary dependent variable was total lines cleared per game.

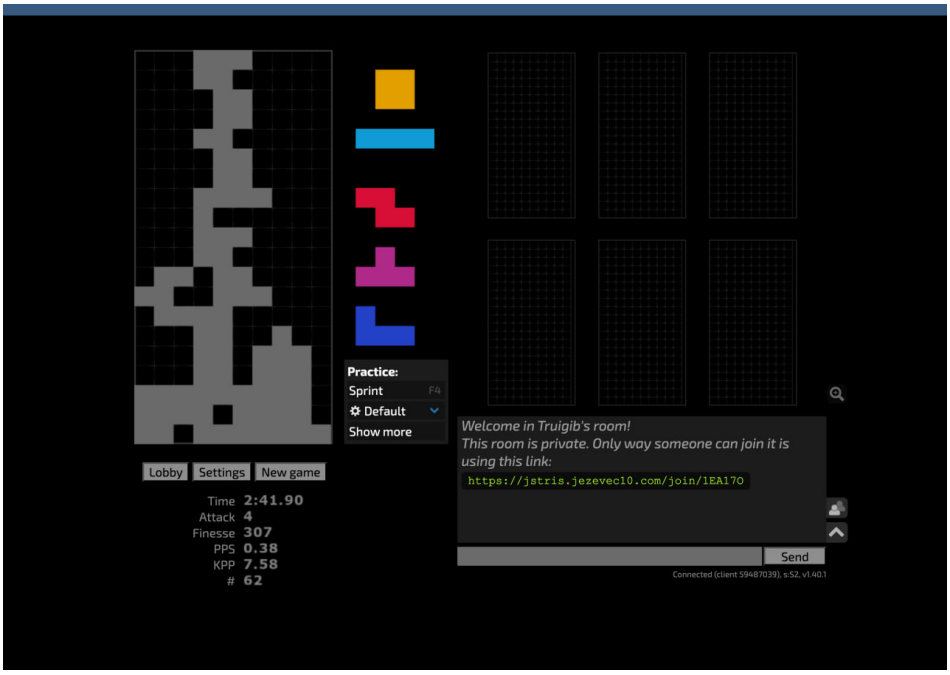

Figure 3. Gameplay trial environment used in Experiment 2. The interface shows the active playfield, upcoming piece preview, and hold queue. All trials were conducted under identical difficulty and rule settings.

2.4. Statistical Analysis

Descriptive statistics were computed for all measures. Uniformity was tested using chi-square goodness-of-fit tests [8]. Gameplay performance differences were evaluated using a two-sample t-test and Cohen's d effect size [9].

 doi:10.20944/preprints202601.2206.v1

3. Results

3.1. Sequence Fairness

The two randomization methods showed clear differences in how piece sequences were generated. Uniform RNG produced long and highly variable I-piece droughts. Across all runs, the average maximum drought length was 44.2 pieces, with some runs reaching up to 69 pieces.

In contrast, the 7-bag randomization system produced a maximum I-piece drought of exactly 12 pieces in every run, with no variation. This confirms that the 7-bag algorithm effectively limits extreme droughts.

Table 1. Experiment 1: I-Piece Drought Statistics.

Algorithm	Mean	SD	Min	Max
Uniform RNG	44.2	7.0	34	69
7-Bag	12.0	0.0	12	12

Despite these differences, both methods maintained uniform long-term piece frequencies. Chi-square tests did not show significant deviations from the expected uniform distribution ($p > 0.05$).

3.2. Gameplay Performance

Gameplay trials showed that players generally performed better under 7-bag randomization. The average number of lines cleared per game was higher for 7-bag randomization (108.6 lines) than for uniform RNG (95.2 lines), representing an increase of approximately 14%.

The variability in performance was slightly lower under 7-bag randomization. However, due to the small number of games played, this difference was not statistically significant ($p = 0.36$).

4. Discussion

The results empirically confirm the theoretical guarantees of 7-bag randomization. By bounding extreme droughts, the algorithm eliminates pathological sequences that trigger perceptions of unfairness, aligning gameplay outcomes with human expectations of randomness [2,3]. While gameplay improvements were modest, reduced variance suggests increased stability and predictability for players.

5. Conclusion

This study demonstrates that 7-bag randomization significantly reduces extreme tetromino droughts while preserving long-term uniformity. Gameplay trials indicate modest but consistent performance benefits. These findings support the continued use of constrained randomness in *Tetris* and provide a reproducible framework for evaluating fairness-oriented randomization strategies in games.

References

1. 'Tetris Guideline," *Tetris Wiki*, 2021.
2. D. Kahneman and A. Tversky, "Judgment under uncertainty: Heuristics and biases," *Science*, 1974.
3. T. Gilovich, R. Vallone, and A. Tversky, "The hot hand in basketball," *Cognitive Psychology*, 1985.
4. R. Gentle, *Random Number Generation and Monte Carlo Methods*. Springer, 2004.
5. E. D. Demaine *et al.*, "Tetris is hard," *Computational Geometry*, 2003.
6. P. Szita and A. Lőrincz, "Learning Tetris using the noisy cross-entropy method," *Neural Computation*, 2006.
7. Jstris Developers, "Jstris: Online Tetris," 2023.
8. NIST, "A statistical test suite for random and pseudorandom number generators," SP 800-22.
9. J. Cohen, *Statistical Power Analysis for the Behavioral Sciences*. Erlbaum, 1988.