ChaosWalker: Non-Repeating Pseudorandom Traversal for Robust Exploration of Large Keyspaces

J. Ekrami

Abstract

Traditional brute-force methods explore keyspaces in deterministic or memoryless random orders, both of which exhibit significant limitations at scale. Sequential traversal introduces strong positional bias, while random sampling suffers from redundancy and heavy-tailed discovery times.

We propose ChaosWalker, a non-repeating pseudorandom traversal framework based on Feistel permutations, designed to eliminate redundancy and distribute exploration uniformly across large keyspaces. While equivalent to any deterministic traversal in expected discovery time under a uniform target distribution, ChaosWalker demonstrates improved robustness in structured and biased domains by reducing variance and eliminating pathological worst-case behaviors.

Experimental results on keyspaces up to 2^{20} show that ChaosWalker produces tightly bounded discovery distributions, avoids inefficiencies of random sampling, and achieves behavior comparable to full-period permutation generators such as LCGs, while additionally providing stronger pseudorandom ordering properties. The method is memoryless, parallelizable, and suitable for large-scale search under uncertainty.

1 Introduction

Exhaustive search is a fundamental method for exploring large discrete domains, including cryptographic keyspaces and combinatorial search spaces [1]. Standard approaches rely on either sequential traversal or random sampling.

Sequential traversal introduces strong positional bias: early regions are explored exhaustively before later regions. Random sampling distributes exploration globally but suffers from redundancy and heavy-tailed discovery times due to repeated sampling [2].

In many practical settings, such as password search or structured combinatorial problems, the target distribution is not uniform [5, 6]. Under such conditions, the ordering of exploration significantly affects early discovery performance.

We introduce **ChaosWalker**, a non-repeating pseudorandom traversal strategy that eliminates ordering bias while avoiding redundancy. The method constructs a permutation of the keyspace using a Feistel network [3], enabling uniform exploration without storing visited elements.

2 Background

2.1 Sequential and Random Search

Sequential search guarantees discovery within L steps but depends entirely on ordering. Random sampling avoids bias but suffers from redundancy; the expected number of samples to cover the entire space grows as $L \log L$ due to the coupon collector effect [1].

2.2 Non-Repeating Traversals

Memory-efficient permutation generators include:

- Linear congruential generators (LCGs)
- Linear feedback shift registers (LFSRs) [**?**]
- Feistel-based permutations

LCGs and LFSRs are efficient but exhibit structural correlations. Feistel networks provide a general construction for pseudorandom permutations [3, **?**].

2.3 Finite Domains

When the domain size is not a power of two, cycle walking can be used to restrict outputs to the desired range [4].

3 ChaosWalker Algorithm

3.1 Construction

Let L be the keyspace size and $m = 2^{\lceil \log_2 L \rceil}$.

We define a Feistel permutation π over $[0, m-1]$. For each counter i:

$$y = \pi(i)$$

If $y \geq L$, we apply cycle walking:

$$y \leftarrow \pi(y)$$

until $y < L$.

3.2 Feistel Network

We use a balanced Feistel network with r rounds:

$$L_{k+1} = R_k, \quad R_{k+1} = L_k \oplus F(R_k, K_k)$$

The round function F is instantiated using a truncated cryptographic hash (e.g., SHAKE128).

3.3 Properties

- No repetition (permutation)
- Constant memory
- Uniform coverage
- Order independence

4 Theoretical Analysis

4.1 Expected Discovery Time

For any permutation, the rank of a uniformly random target is:

$$\mathbb{E}[R] = \frac{L+1}{2}$$

Random sampling with replacement follows a geometric distribution with expectation L, but includes redundancy.

4.2 Variance

Permutation-based traversal:

$$\text{Var} = \frac{L^2 - 1}{12}$$

Random sampling:

$$\text{Var} \approx L^2$$

Thus, ChaosWalker significantly reduces variance.

5 Experimental Setup

We compare:

- Sequential
- Random sampling
- LCG
- ChaosWalker

Keyspace sizes:

$$L = 10^4, 10^6, 2^{20}$$

We run multiple seeds for stochastic methods.

5.1 Target Distributions

- Uniform
- Clustered
- Multi-cluster

6 Results

6.1 Uniform Case

All methods match expected performance. ChaosWalker shows lower variance than random sampling.

6.2 Clustered Case

ChaosWalker improves early discovery probability compared to sequential traversal.

6.3 Distribution Analysis

Random sampling exhibits heavy tails; ChaosWalker produces bounded distributions.

7 Discussion

ChaosWalker eliminates redundancy and ordering bias, improving robustness and predictability. It behaves similarly to a random permutation while maintaining deterministic coverage.

8 Limitations

- No improvement in expected time under uniform targets
- Cycle walking overhead
- Computational cost of hash-based rounds

9 Conclusion

ChaosWalker reframes brute-force search as a problem of exploration quality. It provides a robust and efficient traversal strategy for large keyspaces under uncertainty.

References

[1] W. Feller, *Probability Theory*, 1968.

[2] R. Motwani, *Randomized Algorithms*, 1995.

[3] M. Luby and C. Rackoff, 1988.

[4] C. Shannon, 1949.

[5] J. Black and P. Rogaway, 2002.

[6] M. Weir, 2009.

[7] J. Bonneau, 2012.

[8] S. Golomb, 1967.