Cross Sums Decompression:

Deterministic Elimination and the Game of Beliefs Protocol

Sam Caldwell  
Asymmetric Effort, LLC  
[scaldwell@asymmetric-effort.com](mailto:scaldwell@asymmetric-effort.com)

# Abstract

This paper introduces a new decompression framework for the Cross Sums Compression and Expansion (CRSCE) algorithm, focusing on a deterministic and parallelizable approach to reconstructing compressed binary data. The pipeline is composed of two stages: an initial Deterministic Elimination (DE) stage, followed by a probabilistic Game of Beliefs Protocol (GOBP). This is a significant improvement over the prior Radditz Sieve model. CRSCE decompression leverages cross-sum metrics (LSM, VSM, DSM, XSM) and cryptographic hash validation (LHASH) to iteratively reconstruct the original binary matrix (CSM) representing a single data block. Where the original Radditz Sieve was designed on paper and using an 80286 computer, this new algorithm is designed for use on much more powerful graphics processors (GPU) in much less time. GOBP combines Loopy Belief propagation and row-constrained cellular automation with fixed-bit budget constraints.

**Keywords:** lossless decompression, cross-sums, belief propagation, cellular automata, hash validation, parallel inference

Table of Contents

[Abstract 2](#_Toc198251694)

[Introduction 5](#_Toc198251695)

[Decompression Overview 6](#_Toc198251696)

[Deterministic Elimination (DE) 6](#_Toc198251697)

[Game of Beliefs Protocol (GOBP) 6](#_Toc198251698)

[Belief Field Initialization 7](#_Toc198251699)

[Row-Master Thread Model 7](#_Toc198251700)

[Cellular Automaton (CA) Phase 7](#_Toc198251701)

[Row Hash Validation 7](#_Toc198251702)

[Convergence Criteria 8](#_Toc198251703)

[Computational Complexity Analysis 9](#_Toc198251704)

[Deterministic Elimination (DE) 9](#_Toc198251705)

[Game of Beliefs Protocol (GOBP) 9](#_Toc198251706)

[Combined Complexity 10](#_Toc198251707)

[Memory Complexity Analysis 11](#_Toc198251708)

[Performance and GPU Acceleration 12](#_Toc198251709)

[Entropy-Independent Convergence Behavior 12](#_Toc198251710)

[Comparison to Traditional Compression 13](#_Toc198251711)

[References 14](#_Toc198251712)

# Introduction

The decompression phase of the CRSCE algorithm must invert a structure-preserving transformation that maps a binary input matrix into a compact representation of its lateral, vertical, diagonal, and anti-diagonal cross-sums along with row-wise cryptographic hashes using SHA-256. The prior decompression algorithm developed for CRSCE relied on brute-force search strategies (e.g., Sieve or Sift methods), which are neither scalable nor efficient. We propose a principled approach combining deterministic rule-based inference and probabilistic graphical modeling, using the Game of Beliefs Protocol (GOBP) as the backbone of post-elimination inference. The goal of this improved algorithm is to reduce the time required to decompress information and thereby make CRSCE a viable general-purpose compression alternative to traditional algorithms. Where the original sieve-and-sift decompression algorithm was too slow for anything beyond archival use, this approach enables broader applicability by significantly improving performance through deterministic inference and GPU-accelerated probabilistic resolution.

# Decompression Overview

Decompression proceeds as a loop of the following stages until convergence:

1. **Deterministic Elimination (DE)**: Apply exact inference rules to reduce entropy.
2. **Game of Beliefs Protocol (GOBP)**: Use cellular automation with belief propagation and row-local updates to resolve uncertain bits.
3. **Verification**: Validate resolved rows using SHA-256 hash checks against the LHASH matrix.

Convergence occurs when all rows are verified or the system reaches a stable fixed point.

## Deterministic Elimination (DE)

DE aims to resolve all bits that can be deterministically inferred. This includes:

* Rows, columns, diagonals, or anti-diagonals with sums equal to 0 or s.
* Lines with exactly one unknown bit, inferable by subtraction.
* Early elimination of known-pattern rows via LHASH lookup (e.g., alternating bit patterns).

The output of DE is a partially filled Cross-Sum Matrix (CSM), with many bits fixed and entropy localized in more ambiguous rows. Each element of the CSM includes a lock-bit. Once a bit is solved—using DE or any other mechanism—it is locked via this lock-bit to prevent further modification by downstream inference phases.

## Game of Beliefs Protocol (GOBP)

### Belief Field Initialization

After DE, each unresolved CSM bit is assigned a probability , estimated through Loopy Belief Propagation (LBP). The factor graph consists of variable nodes for each unknown bit and factor nodes representing cross-sum constraints.

### Row-Master Thread Model

Each unsolved row spawns a row-master thread that:

* Receives the fixed bit budget from LSM.
* Initializes the row with exactly that number of set (1) values, based on belief strengths.
* Spawns cellular automaton (CA) threads to manage slices of the row.

Budgeted set (1) values budgeted for a row cannot be transferred to another row, as this would violate the requirements of

### Cellular Automaton (CA) Phase

Each CA thread attempts to reduce local constraint violations by migrating set (1) values within the row:

* Prioritize cells with high belief and minimal residual pressure from VSM, DSM, and XSM.
* Preserve the row's bit budget at all times.
* Synchronize via GPU shared memory within row-local threadblocks.

### Row Hash Validation

Each row-master computes the SHA-256 hash of the current row. If it matches the LHASH entry and cross-sum consistency is preserved, the row is locked and marked as solved. Otherwise, it continues updating for a fixed number of steps or until convergence.

### Convergence Criteria

The protocol terminates when:

* All rows pass SHA-256 verification.
* Or no further changes occur in the belief field after a maximum iteration threshold.

Optionally, rows that fail to converge may be flagged for fallback methods, though brute-force phases are omitted in the current design.

# Computational Complexity Analysis

## Deterministic Elimination (DE)

Let be the dimension of the Cross-Sum Matrix (CSM), fixed at throughout our implementation.

DE iteratively resolves matrix bits using cross-sum rules and hash-based pattern resolution. Each of the four cross-sum vectors (LSM, contains entries. Each entry potentially involves scanning up to ss bits, and updates propagate as elements are solved and locked. This gives:

DE time complexity:  per pass, up to  passes⇒.

In practice, the number of effective passes is reduced by the locking mechanism, which prevents reprocessing of resolved rows/columns.

The DE phase also attempts hash-based resolution for known patterns, requiring. SHA-256 hashes (one per row), each of cost , thus also .

## Game of Beliefs Protocol (GOBP)

Post-DE, remaining bits are handled using probabilistic inference and local constraint optimization. GOBP consists of:

1. ****Factor Graph Initialization****:

* Build a bipartite graph with up to variable nodes and factor nodes.

1. ****Loopy Belief Propagation (LBP)****:

* For each edge (up to 4), messages are exchanged for iterations:

, assuming  is constant-bound (e.g., 10–20).

1. ****Cellular Automata (CA)****:

* Each row contains ss bits. Local adjustments (bit flips, shifts) are attempted under strict LSM budgets.
* Each row thread executes R rounds: , with R constant (e.g., 100).

1. ****Row Validation****:

* Each row is hashed and compared to LHASH:  per iteration.

Total GOBP complexity per block is:

With fixed constants T, R, this is asymptotically .

## Combined Complexity

Combining both stages, we get a total decompression complexity per block of . This upper bound is consistent with empirical convergence behavior, particularly for high-entropy inputs.

# Memory Complexity Analysis

Each decompression block requires:

* CSM matrix: bits = 32KB
* Cross-sum vectors: bits, = 9
* LHASH matrix: bits = 16KB
* Belief matrix: 4 bytes = 1MB (32-bit floats)
* Temporary GPU buffers and row automata state: bytes

Total approximate memory per block:

≈1.05 MB

This assumes a fixed block size s.

# Performance and GPU Acceleration

The CRSCE decompression process is explicitly designed for high-parallel throughput on modern GPUs:

* **DE Phase**: Amenable to thread-parallelism, particularly for parallel reductions along rows, columns, and diagonals.
* **GOBP LBP Phase**: Maps naturally to CUDA warp/block architectures using message-passing semantics.
* **CA Threads**: Executed as row-local thread blocks using shared memory for synchronization and bit-shifting operations.
* **Hash Validation**: Can be batched and overlapped using warp-shuffle and crypto intrinsics.

### Entropy-Independent Convergence Behavior

A unique feature of CRSCE is its predictable performance, independent of input entropy. Unlike traditional compressors (e.g., LZ77 or Huffman) whose performance degrades on random data, CRSCE’s decompression cost is tied to bit structure, not frequency. Highly structured rows (e.g., 0-filled, alternating) are resolved in DE. More complex rows defer to GOBP.

Therefore, performance is **entropy-agnostic** and instead tracks the spatial regularity of the CSM and its cross-sum budget constraints.

# Comparison to Traditional Compression

Comparative Summary:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Algorithm | Time Complexity (Worst Case) | Input-Entropy Sensitivity | Predictable Runtime | GPU Suitability |
| DEFLATE (gzip) | O(n2)\mathcal{O}(n^2)O(n2) | High (degrades on random) | ✗ | Moderate |
| CRSCE | O(s3)\mathcal{O}(s^3)O(s3) | Low (entropy-agnostic) | ✓ | Excellent |

# **References**

Dworkin, M. (2015). SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. NIST FIPS PUB 202.

Koller, D., & Friedman, N. (2009). Probabilistic Graphical Models: Principles and Techniques. MIT Press.

Salomon, D., & Motta, G. (2010). Handbook of Data Compression. Springer.

Sayood, K. (2018). Introduction to Data Compression (5th ed.). Morgan Kaufmann.

Stevens, M., et al. (2017). The first collision for full SHA-1. In Advances in Cryptology – CRYPTO 2017. Springer.