New Approaches to Uniformity Testing

Winston Li

Rutgers University

Mentor: Periklis Papakonstantinou

July 17, 2025

Randomness

Problem

Given a string of bits X, can we determine if X is **random**?

Randomness

Problem

Given a string of bits X, can we determine if X is **random**?

Hypothesis Testing

Write $X := X_1 X_2 \dots X_n$. If X is random, then for all $i \in [n]$, $X_i \sim \text{Bern}(\frac{1}{2})$

Randomness

Problem

Given a string of bits X, can we determine if X is **random**?

Hypothesis Testing

Write $X := X_1X_2 \dots X_n$. If X is random, then for all $i \in [n]$, $X_i \sim \text{Bern}(\frac{1}{2})$

Frequency Test (NIST)

Define $S_n = X_1 + X_2 + \dots + X_n$. By Central Limit Theorem, for large enough n (too big), $S_n \to \mathcal{N}(\frac{n}{2}, \frac{n}{4})$. Performing a z-test, we can compute the probability for the value of S_n occurring.

Randomness

Problem

Given a string of bits X, can we determine if X is **random**?

Hypothesis Testing

Write $X := X_1 X_2 \dots X_n$. If X is random, then for all $i \in [n]$, $X_i \sim \text{Bern}(\frac{1}{2})$

Frequency Test (NIST)

Define $S_n = X_1 + X_2 + \dots + X_n$. By Central Limit Theorem, for large enough n (too big), $S_n \to \mathcal{N}(\frac{n}{2}, \frac{n}{4})$. Performing a z-test, we can compute the probability for the value of S_n occurring.

Question

Can we do better?

Concentration Bounds

Chernoff-Hoeffding Bounds

Let $X = \sum_{i=1}^{n} X_i$, where X_i are independent random variables distributed in $[0, 1]$. For all $t > 0$,

$$\Pr[X > \mathbb{E}[X] + t] \leq e^{-2t^2/n}$$

$$\Pr[X < \mathbb{E}[X] - t] \leq e^{-2t^2/n}$$

Concentration Bounds

Chernoff-Hoeffding Bounds

Let $X = \sum_{i=1}^{n} X_i$, where X_i are independent random variables distributed in $[0, 1]$. For all $t > 0$,

$$\Pr[X > \mathbb{E}[X] + t] \leq e^{-2t^2/n}$$

$$\Pr[X < \mathbb{E}[X] - t] \leq e^{-2t^2/n}$$

Frequency Test

Recall $S_n = \sum_{i \in [n]} X_i$ where $X_i \sim \text{Bern}(\frac{1}{2})$. Then $\mathbb{E}[X] = \frac{n}{2}$, so $\Pr[|S_n - \frac{n}{2}| > t] \leq 2e^{-2t^2/n}$. For $\alpha \in (0, 1)$, take $t = \sqrt{\frac{n}{2}(\log 2 - \log \alpha)}$. The probability S_n is farther than t away from $\frac{n}{2}$ is less than α.

Concentration Bounds

Block Frequency Test

Let X be a bit string of length $n \times m$. Write $X = X_1 \dots X_n$, where X_i is a substring of length m. Let $\pi_i = \frac{1}{m} \sum_{j=1}^{m} X_{ij}$ be the proportion of bits in X_i that are 1. Then $S = 4m \sum_{i=1}^{n} (\pi_i - \frac{1}{2})^2$ is a χ^2 distribution.

Concentration Bounds

Block Frequency Test

Let X be a bit string of length $n \times m$. Write $X = X_1 \dots X_n$, where X_i is a substring of length m. Let $\pi_i = \frac{1}{m} \sum_{j=1}^{m} X_{ij}$ be the proportion of bits in X_i that are 1. Then $S = 4m \sum_{i=1}^{n} (\pi_i - \frac{1}{2})^2$ is a χ^2 distribution.

Chernoff Bounds (χ^2)

Let $Y_i = \frac{2\pi_i - m}{\sqrt{m}}$, so $S = \sum_{i=1}^{n} Y_i^2$. Applying Chernoff bounds,

$$M_{Y_i}(\lambda) \leq (1-2\lambda)^{-1/2} \qquad M_S(\lambda) \leq (1-2\lambda)^{-n/2}$$
$$\Pr[S \geq t] \leq \min_\lambda e^{-\lambda t}(1-2\lambda)^{-n/2} = \left(\tfrac{t}{n}\right)^{n/2} e^{(n-t)/2}$$

Concentration Bounds

Block Frequency Test

Let X be a bit string of length $n \times m$. Write $X = X_1 \dots X_n$, where X_i is a substring of length m. Let $\pi_i = \frac{1}{m} \sum_{j=1}^{m} X_{ij}$ be the proportion of bits in X_i that are 1. Then $S = 4m \sum_{i=1}^{n} (\pi_i - \frac{1}{2})^2$ is a χ^2 distribution.

Chernoff Bounds (χ^2)

Let $Y_i = \frac{2\pi_i - m}{\sqrt{m}}$, so $S = \sum_{i=1}^{n} Y_i^2$. Applying Chernoff bounds,

$$M_{Y_i}(\lambda) \leq (1-2\lambda)^{-1/2} \qquad M_S(\lambda) \leq (1-2\lambda)^{-n/2}$$
$$\Pr[S \geq t] \leq \min_\lambda e^{-\lambda t}(1-2\lambda)^{-n/2} = \left(\tfrac{t}{n}\right)^{n/2} e^{(n-t)/2}$$

Problem

Notice m is not included in the bound. For χ^2 tests, this scales with class count, but not string length.

Extractors

Definition

The **statistical difference** between two random variables X, Y is $\Delta(X, Y) := \frac{1}{2}|X - Y|_1$. X and Y are ε-close if $\Delta(X, Y) \leq \varepsilon$.

Extractors

Definition

The **statistical difference** between two random variables X, Y is $\Delta(X, Y) := \frac{1}{2}|X - Y|_1$. X and Y are ε-close if $\Delta(X, Y) \leq \varepsilon$.

Definition

Let X be a random variable.
The **Shannon Entropy** is $H_{Sh}(X) = \mathbb{E}_{x \sim X}[-\log \Pr[X = x]]$.
The **min-entropy** is $H_\infty(X) = \min_x(-\log \Pr[X = x])$.

Extractors

Definition

The **statistical difference** between two random variables X, Y is $\Delta(X, Y) := \frac{1}{2}|X - Y|_1$. X and Y are ε-close if $\Delta(X, Y) \leq \varepsilon$.

Definition

Let X be a random variable.
The **Shannon Entropy** is $H_{Sh}(X) = \mathbb{E}_{x \sim X}[-\log \Pr[X = x]]$.
The **min-entropy** is $H_\infty(X) = \min_x(-\log \Pr[X = x])$.

Definition

A random variable X is a k-source if $H_\infty(X) \geq k$, or $\Pr[X = x] \leq 2^{-k}$.

Extractors

Definition

The **statistical difference** between two random variables X, Y is $\Delta(X, Y) := \frac{1}{2}|X - Y|_1$. X and Y are ε-close if $\Delta(X, Y) \leq \varepsilon$.

Definition

Let X be a random variable.
The **Shannon Entropy** is $H_{Sh}(X) = \mathbb{E}_{x \sim X}[-\log \Pr[X = x]]$.
The **min-entropy** is $H_\infty(X) = \min_x(-\log \Pr[X = x])$.

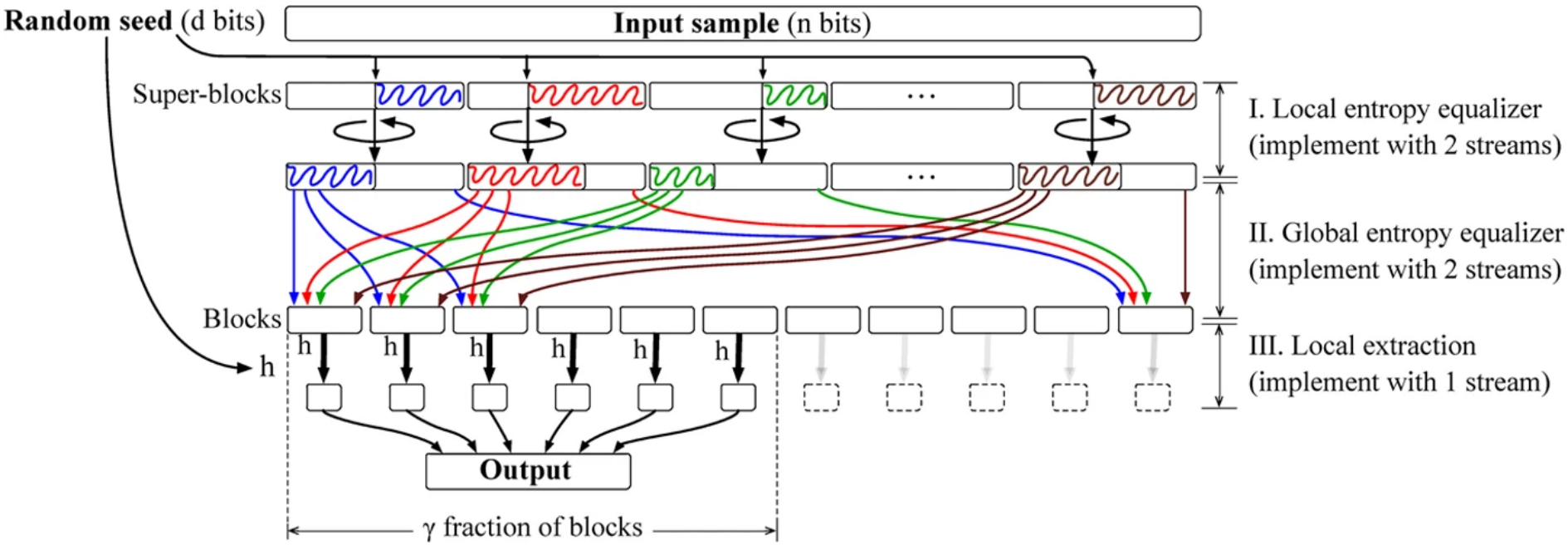
Definition

A random variable X is a k-source if $H_\infty(X) \geq k$, or $\Pr[X = x] \leq 2^{-k}$.

Definition

$\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a (k, ε)-**extractor** if for every k-source X, $\mathrm{Ext}(X, U_d)$ is ε close to U_m.

Randomized Re-Bucketing Extractor



Implementation

Definition

A **Toeplitz Matrix** is a matrix with constant diagonals.

$$A = \begin{bmatrix} a_0 & a_1 & a_2 & a_3 \\ a_{-1} & a_0 & a_1 & a_2 \\ a_{-2} & a_{-1} & a_0 & a_1 \end{bmatrix} \qquad A_{i,j} = A_{i+1,j+1}$$

Implementation

Definition

A **Toeplitz Matrix** is a matrix with constant diagonals.

$$A = \begin{bmatrix} a_0 & a_1 & a_2 & a_3 \\ a_{-1} & a_0 & a_1 & a_2 \\ a_{-2} & a_{-1} & a_0 & a_1 \end{bmatrix} \qquad A_{i,j} = A_{i+1,j+1}$$

Property 1

Toeplitz matrices over $\mathbb{F}_2$ form a family of **universal hash functions**.

Implementation

Definition

A **Toeplitz Matrix** is a matrix with constant diagonals.

$$A = \begin{bmatrix} a_0 & a_1 & a_2 & a_3 \\ a_{-1} & a_0 & a_1 & a_2 \\ a_{-2} & a_{-1} & a_0 & a_1 \end{bmatrix} \qquad A_{i,j} = A_{i+1,j+1}$$

Property 1

Toeplitz matrices over $\mathbb{F}_2$ form a family of **universal hash functions**.

Property 2

An $m \times n$ Toeplitz matrix is generated by a $m + n - 1$ dimension vector.

$$A \in \mathbb{F}_2^{m \times n} \qquad w = (a_{1-m}, \ldots, a_{m+n-1})$$
$$Av = \bigotimes_{i=1}^{n} v_i w_{i:i+m}$$

Compression Testing

Source Coding Theorem (Shannon)

For a constant stream of bits, the maximum compression ratio asymptotically approaches the Shannon entropy of the source.

Compression Testing

Source Coding Theorem (Shannon)

For a constant stream of bits, the maximum compression ratio asymptotically approaches the Shannon entropy of the source.

Experiment

Fix some computable sequence $f : \mathbb{N} \to \{0, 1\}$ and proportion $p \in [0, 1]$. Use f to poison a bit string $X := X_1 \dots X_n$, generating $Y := Y_1 \dots Y_n$,

$$Y_i = \begin{cases} X_i & \text{with probability } p \\ f(i) & \text{with probability } 1 - p \end{cases}$$

We expect $H_{Sh}(X) = n$ and $H_{Sh}(Y) \approx nH_b(\frac{1-p}{2})$. Expect $|\mathrm{compress}(Y)|$ to be similar to $H_{Sh}(Y)$.

Compression Testing

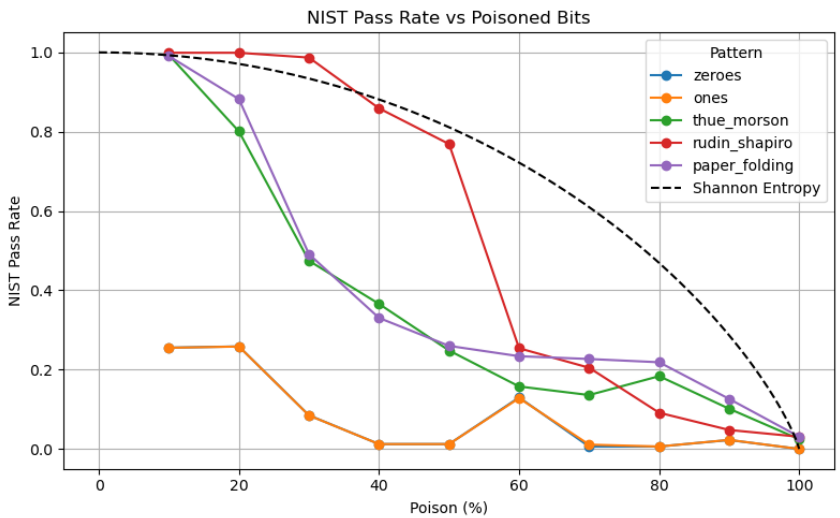
Setup

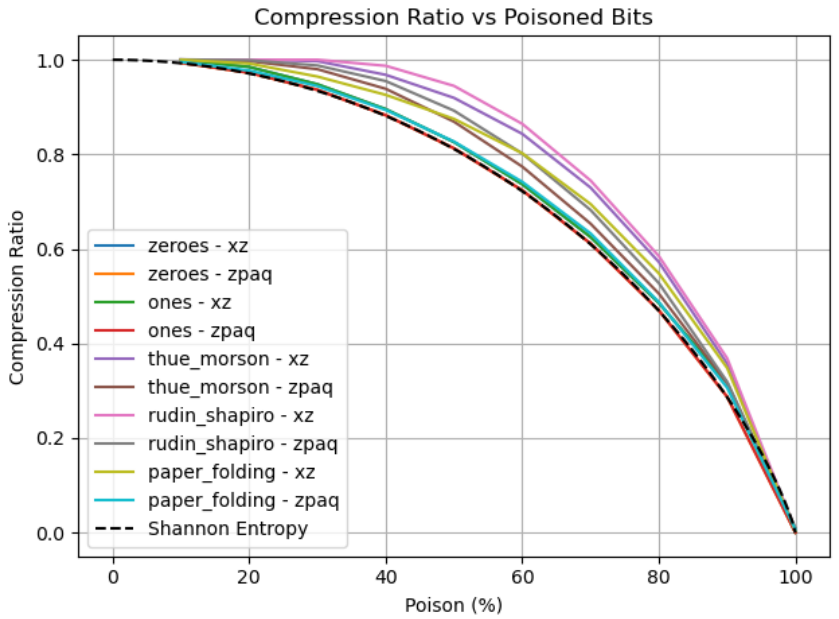
- Extract seed from 2.6GB of audio files using BIWZ extractor.
- Extract data from 64GB movie file with RRB extractor and seed.
- Poison extracted data using a fixed sequence.
- Compress poisoned file with xz and zpaq at max setting.
- Run NIST tests on poisoned file.

Compression Testing

Setup

- Extract seed from 2.6GB of audio files using BIWZ extractor.
- Extract data from 64GB movie file with RRB extractor and seed.
- Poison extracted data using a fixed sequence.
- Compress poisoned file with xz and zpaq at max setting.
- Run NIST tests on poisoned file.





Conspiracy

RRB Parameterization

Given a bit string of length 67,108,864 (8 MiB), split it into 50 substrings. Run the NIST test suite on each substring. The program takes an unusually long time computing a DFT (Discrete Fourier Transform).

Truncating the string to $50 \times 1{,}000{,}000$ bits, the tests run instantly.

Conspiracy

RRB Parameterization

Given a bit string of length 67,108,864 (8 MiB), split it into 50 substrings. Run the NIST test suite on each substring. The program takes an unusually long time computing a DFT (Discrete Fourier Transform).

Truncating the string to $50 \times 1{,}000{,}000$ bits, the tests run instantly.

Lightning Strikes!

It turns out $\lfloor \frac{67108864}{50} \rfloor = 1342177$ is a prime number. For a bit string of length $n = p_1 p_2 \dots p_k$, NIST's DFT runs in $O(n \sum_k p_i)$. This is around $O(n \log n)$ if n has many factors, but $O(n^2)$ if n is prime.

Conspiracy

RRB Parameterization

Given a bit string of length 67,108,864 (8 MiB), split it into 50 substrings. Run the NIST test suite on each substring. The program takes an unusually long time computing a DFT (Discrete Fourier Transform).

Truncating the string to $50 \times 1{,}000{,}000$ bits, the tests run instantly.

Lightning Strikes!

It turns out $\lfloor \frac{67108864}{50} \rfloor = 1342177$ is a prime number. For a bit string of length $n = p_1 p_2 \dots p_k$, NIST's DFT runs in $O(n \sum_k p_i)$. This is around $O(n \log n)$ if n has many factors, but $O(n^2)$ if n is prime.

Solution

Splitting the bit string into 64 substrings, or dropping a bit from each substring resolves this problem.

Acknowledgements

- I would like to thank my mentor Professor Periklis Papakonstantinou for guiding me through this project.
- I would like to thank Larry for being Larry.
- This work was carried out as part of the 2025 DIMACS REU program at Rutgers University, supported by NSF grant CCF-2247342

References

[1] Boaz Barak, Russell Impagliazzo, and Avi Wigderson. "Extracting Randomness Using Few Independent Sources". In: *SIAM Journal on Computing* 36.4 (2006), pp. 1095–1118. DOI: 10.1137/S0097539705447141.

[2] Lawrence Bassham et al. *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*. en. 2010.

[3] Devdatt Dubhashi and Alessandro Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. 1st. USA: Cambridge University Press, 2009. ISBN: 0521884276.

[4] Periklis A. Papakonstantinou, David P. Woodruff, and Guang Yang. "True Randomness from Big Data". en. In: *Scientific Reports* 6.1 (2016). DOI: 10.1038/srep33740.

[5] C. E. Shannon. "A mathematical theory of communication". In: *The Bell System Technical Journal* 27.3 (1948), pp. 379–423. DOI: 10.1002/j.1538-7305.1948.tb01338.x.

Bonus!

Jul 16 11:55 PM

tmux attach

Main.cpp × Extractor.cpp × Config.cpp × meson.build × Extractor.h ×

Extractor › extract

```
// cache toeplitz row vectors
std::cout << "Generating Toeplitz row vector bit offsets" << std::endl;
using block_type = boost::dynamic_bitset<>::block_type;
uint64_t vecWidth = (opt.imageSize + 63) / 64;
std::vector<std::vector<block_type>> templates(64);
for (int i = 0; i < 64; i++) {
  auto window = opt.toeplitzSeed;
  window >>= i;
  boost::to_block_range(window, std::back_inserter(templates[i]));
}

std::cout << "Beginning Extraction" << std::endl;
std::vector<block_type> vec(vecWidth);
for (auto _ : tqdm(opt.outputBlockCount, opt.progressFreq)) {
  std::fill(vec.begin(), vec.end(), 0ULL);
  // Let A = toeplitz matrix, b_j = sequences[j][i]
  for (uint64_t j = 0; j < opt.sequences.size(); j++) {
    // if b_j == 1, then vec += A * e_j
    if (opt.sequences[j].next()) {
      // memory optimization
      uint64_t bitInx = j % 64;
      uint64_t byteInx = j / 64;
      auto &tmpl = templates[bitInx];
      for (uint64_t b = 0; b < vecWidth; b++) {
        vec[b] ^= tmpl[b + byteInx];
      }
    }
  }

  // vec = A * b
  boost::dynamic_bitset<> bitVector(vec.begin(), vec.end());
  bitVector.resize(opt.imageSize);
  opt.file.write(bitVector);
```

main cpp clangd TS

```
Block Count: 2048
Block Size: 8192
Super Block Size: 131072
Image Block Size: 1024
Loading Seed
Toeplitz Row Vector (9215): 0000000100001111100001100001101100001011110101010110001111111001111...
Rotations (8192): 102634 599 51066 58892 93693 25518 106601 83338 60232 79445 ...
Loading Sequences
==1585958== brk segment overflow in thread #1: can't grow to 0x4894000
==1585958== (see section Limitations in user manual)
==1585958== NOTE: further instances of this message will not be shown
Generating Toeplitz row vector bit offsets
Beginning Extraction
[==================================================] 100.00% 86.49 iter/s
==1585958==
==1585958== Events    : Ir
==1585958== Collected : 8063945379
==1585958==
==1585958== I   refs:      8,063,945,379
caa84fbf1a54710002b6f1793831f9b402ff3d4e  out.bin
lita@icosahedron ~/..../Extractor main
```

$ perf.sh ×

```
sha1sum out.bin

output_file=$(ls -r ./traces/callgrind.out.* | head -n1)
echo "Callgrind output written to: ${output_file}"
```

main $ sh 3 6 TS

lita@icosahedron:~/Desktop/Projects/UniformityTesting/Extractor

callgrind.out.1585958 [./build-perf/rbbextractor -s data/seed.bin -i ../Tester/data/movie.y4m -o out.bin -b 8192 -c 2048 -d 1024 -l 1073741824 -D 8] — KCachegrind

File View Go Settings Help

Open... Back Forward Up Relative Cycle Detection Relative to Parent Shorten Templates Instruction Fetch

Flat Profile

Search: Search Query (No Grouping)

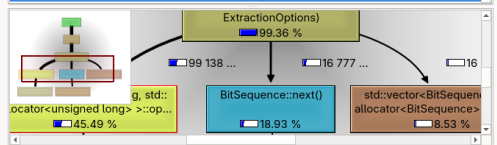
Incl.	Self	Called	Function	Location
100.00	0.00	(0)	0x000000000001d5c0	ld-linux-x86-64.so.2
99.96	0.00	1	(below main)	rbbextractor
99.96	0.00	1	__libc_start_main@@G...	libc.so.6: libc-start.c
99.96	0.00	1	(below main)	libc.so.6: libc_start_call_main.h
99.96	0.00	1	main	rbbextractor: Main.cpp
99.36	16.91	1	Extractor::extract(Extra...	rbbextractor: Extractor.cpp, stl_iterator.h, ne...
45.50	27.05	99 164 424	std::vector<unsigned l...	rbbextractor: stl_vector.h
19.10	11.96	16 925 695	BitSequence::next()	rbbextractor: File.cpp
18.84	18.84	101 284 768	std::vector<unsigned l...	rbbextractor: stl_vector.h
8.53	5.20	16 777 216	std::vector<BitSequen...	rbbextractor: stl_vector.h
7.14	4.41	16 925 695	std::vector<unsigned c...	rbbextractor: stl_vector.h
6.66	6.66	33 556 483	std::vector<BitSequen...	rbbextractor: stl_vector.h
4.35	0.42	2 048	OutputFile::write(boost...	rbbextractor: File.cpp
2.99	0.31	2 097 152	boost::dynamic_bitset<...	rbbextractor: dynamic_bitset.hpp
2.73	2.73	16 942 079	std::vector<unsigned c...	rbbextractor: stl_vector.h
2.68	0.42	2 097 152	boost::dynamic_bitset<...	rbbextractor: dynamic_bitset.hpp
2.26	0.60	2 097 152	boost::dynamic_bitset<...	rbbextractor: dynamic_bitset.hpp
1.58	0.96	3 098 148	std::vector<std::vector...	rbbextractor: stl_vector.h
0.96	0.57	2 097 152	std::vector<unsigned l...	rbbextractor: stl_vector.h
0.95	0.77	2 097 152	OutputFile::write(bool)	rbbextractor: File.cpp
0.61	0.61	3 098 148	std::vector<std::vector...	rbbextractor: stl_vector.h
0.52	0.34	2 097 152	boost::dynamic_bitset<...	rbbextractor: dynamic_bitset.hpp
0.22	0.00	1	Extractor::seed(Extract...	rbbextractor: Extractor.cpp
0.22	0.01	1	Extractor::generateSeq...	rbbextractor: Extractor.cpp

callgrind.out.1585958 [1] - Total Instruction Fetch Cost: 8 063 945 379

Extractor::extract(Extractor::ExtractionOptions)

Types Callers All Callers Callee Map Source Code Control Flow Graph

#	Ir	Source
	3.33	16779264 calls to 'std::vector<BitSequence, std::allocator<BitSequ...
61		// if b_j == 1, then vec += A * e_j
62	2.08	if (opt.sequences[j].next()) {
	8.53	16777216 calls to 'std::vector<BitSequence, std::allocator<BitSeque...
	18.93	16777216 calls to 'BitSequence::next()' (rbbextractor: File.cpp)
63		// memory optimization
64	0.12	uint64_t bitInx = j % 64;
65	0.12	uint64_t byteInx = j / 64;
66	0.23	auto &tmpl = templates[bitInx];
	1.58	3098084 calls to 'std::vector<std::vector<unsigned long, std::alloc...
67	2.65	for (uint64_t b = 0; b < vecWidth; b++) {
68	9.84	vec[b] ^= tmpl[b + byteInx];
	45.49	99138688 calls to 'std::vector<unsigned long, std::allocator<unsign...



Parts Callees Call Graph All Callees Caller Map Machine Code