Stochastic Integration via Error-Correcting Codes

Dimitris Achlioptas Pei Jiang

UC Santa Cruz

The Setting

- A (large) domain $\Omega = D_1 \times \cdots \times D_n$, where $\{D_i\}_{i=1}^n$ are finite.
- A non-negative function $f : \Omega \to \mathbb{R}$.

The Goal ("Stochastic Approximate Integration")

Probabilistically, approximately estimate $Z = \sum_{\sigma \in \Omega} f(\sigma)$.

Non-negativity of $f \implies$ No Cancellations

Appplications

- Probabilistic Inference via graphical models (partition function)
- Automatic test-input generation in verification (model counting)
- Generic alternative to MCMC

The Setting

- A (large) domain $\Omega = D_1 \times \cdots \times D_n$, where $\{D_i\}_{i=1}^n$ are finite.
- A non-negative function $f : \Omega \to \mathbb{R}$.

The Goal ("Stochastic Approximate Integration")

Probabilistically, approximately estimate $Z = \displaystyle\sum_{\sigma \in \Omega} f(\sigma)$.

Non-negativity of $f \implies$ No Cancellations

Quality Guarantee

For any accuracy $\epsilon > 0$, with effort proportional to $s\, n/\epsilon^2$,

$$\Pr_{\mathcal{A}}\left[1 - \epsilon < \frac{\widehat{Z}}{Z} < 1 + \epsilon\right] = 1 - \exp(-\Theta(s)) \ .$$

The Setting

- A (large) domain $\Omega = D_1 \times \cdots \times D_n$, where $\{D_i\}_{i=1}^n$ are finite.
- A non-negative function $f : \Omega \to \mathbb{R}$.

The Goal ("Stochastic Approximate Integration")

Probabilistically, approximately estimate $Z = \sum_{\sigma \in \Omega} f(\sigma)$.

Non-negativity of $f \implies$ No Cancellations

Rest of the Talk

- $\Omega = \{0,1\}^n$ — $D_i = \{0,1\}$ for all $i \in [n]$
- 32-approximation. — Typically $Z = \exp(n)$

The Setting

- A (large) domain $\Omega = D_1 \times \cdots \times D_n$, where $\{D_i\}_{i=1}^{n}$ are finite.
- A non-negative function $f : \Omega \to \mathbb{R}$.

The Goal ("Stochastic Approximate Integration")

Probabilistically, approximately estimate $Z = \sum_{\sigma \in \Omega} f(\sigma)$.

Non-negativity of $f \implies$ No Cancellations

General Idea

- For i from 0 to n

  Repeat $\Theta(\epsilon^{-2})$ times

  - Generate random $R_i \subseteq \Omega$ of size $\sim 2^{n-i}$
  - Find $y_i = \max_{\sigma \in R_i} f(\sigma)$
- Combine $\{y_i\}$ in a straightforward way to get $\widehat{Z}$.

The Setting

- A (large) domain $\Omega = D_1 \times \cdots \times D_n$, where $\{D_i\}_{i=1}^n$ are finite.
- A non-negative function $f : \Omega \to \mathbb{R}$.

The Goal ("Stochastic Approximate Integration")

Probabilistically, approximately estimate $Z = \sum_{\sigma \in \Omega} f(\sigma)$.

Non-negativity of $f \implies$ No Cancellations

General Idea

- For i from 0 to n

  Repeat $\Theta(\epsilon^{-2})$ times

  – Generate random $R_i \subseteq \Omega$ of size $\sim 2^{n-i}$ as an ECC

  – Find $y_i = \max_{\sigma \in R_i} f(\sigma)$
- Combine $\{y_i\}$ in a straightforward way to get $\widehat{Z}$.

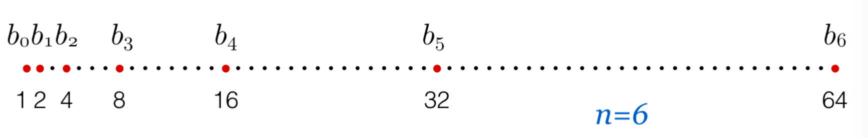
Estimation by Stratification

Thought Experiment

Sort Ω by decreasing f-value. W.l.o.g.

$$f(\sigma_1) \geq f(\sigma_2) \geq f(\sigma_3) \cdots f(\sigma_{2^i}) \cdots \geq f(\sigma_{2^n})$$

Imagine we could get our hands on the $n+1$ numbers $b_i = f(\sigma_{2^i})$.



Estimation by Stratification

Thought Experiment

Sort Ω by decreasing f-value. W.l.o.g.

$$f(\sigma_1) \geq f(\sigma_2) \geq f(\sigma_3) \cdots f(\sigma_{2^i}) \cdots \geq f(\sigma_{2^n})$$

Imagine we could get our hands on the $n+1$ numbers $b_i = f(\sigma_{2^i})$.

If we let

$$U := b_0 + \sum_{i=0}^{n-1} b_i 2^i \qquad \text{and} \qquad L := b_0 + \sum_{i=0}^{n-1} b_{i+1} 2^i$$

then

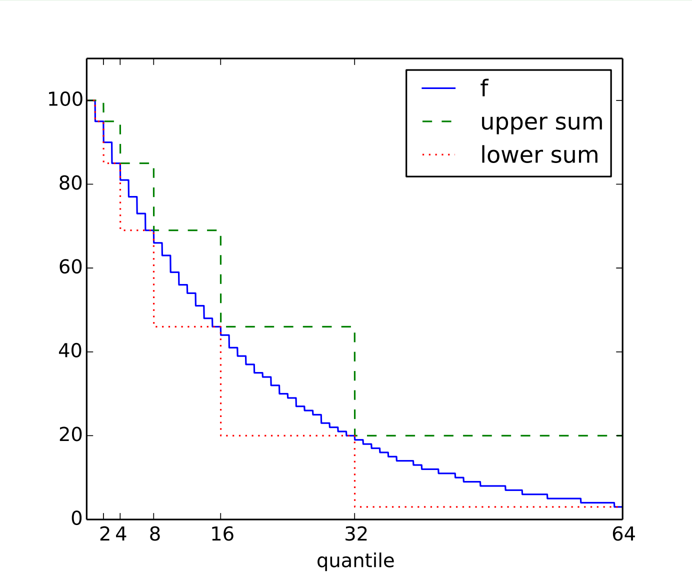
$$L \leq Z \leq U \leq 2L$$

Estimation by Stratification

Thought

Sort Ω by

Imagine



).

Estimation by Stratification

Thought Experiment

Sort Ω by decreasing f-value. W.l.o.g.

$$f(\sigma_1) \geq f(\sigma_2) \geq f(\sigma_3) \cdots f(\sigma_{2^i}) \cdots \geq f(\sigma_{2^n})$$

Imagine we could get our hands on the $n+1$ numbers $b_i = f(\sigma_{2^i})$.

Theorem (EGSS)

To get a 2^{2c+1}-approximation it suffices to find for each $0 \leq i \leq n$,

$$b_{i+c} \leq \widehat{b_i} \leq b_{i-c} \ .$$

Estimation by Stratification

Thought Experiment

Sort Ω by decreasing f-value. W.l.o.g.

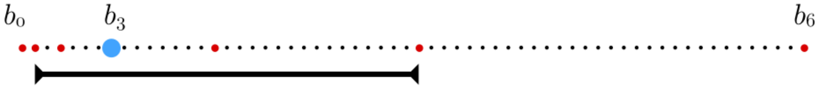
$$f(\sigma_1) \geq f(\sigma_2) \geq f(\sigma_3) \cdots f(\sigma_{2^i}) \cdots \geq f(\sigma_{2^n})$$

Imagine we could get our hands on the $n+1$ numbers $b_i = f(\sigma_{2^i})$.

Corollary (when $c = 2$)

To get a 32-approximation it suffices to find for each $0 \leq i \leq n$,

$$b_{i+2} \leq \widehat{b_i} \leq b_{i-2} \quad .$$



Refinement by Repetition

Lemma (Concentration of measure)

Let X be any r.v. such that:

$$\begin{aligned} \Pr[X \leq \text{Upper}] &\geq 1/2 + \delta \\ &\text{and} \\ \Pr[X \geq \text{Lower}] &\geq 1/2 + \delta \ . \end{aligned}$$

If $\{X_1, X_2, \ldots, X_t\}$ are independent samples of X, then

$$\Pr\left[\text{Lower} \leq \text{Median}(X_1, X_2, \ldots, X_t) \leq \text{Upper}\right] \geq 1 - 2\exp\left(-\delta^2 t\right)$$

The Basic Plan

Thinning Sets

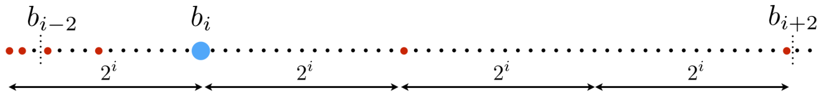
We will consider random sets R_i such that for every $\sigma \in \Omega$,

$$\Pr[\sigma \in R_i] = 2^{-i} \quad .$$

Our estimator for $b_i = f(\sigma_{2^i})$ will be

$$m_i = \max_{\sigma \in R_i} f(\sigma) \quad .$$

Recall that $f(\sigma_1) \geq f(\sigma_2) \geq f(\sigma_3) \cdots \geq f(\sigma_{2^i}) \geq f(\sigma_{2^i} + 1) \cdots \geq f(\sigma_{2^n})$



The Basic Plan

Thinning Sets

We will consider random sets R_i such that for every $\sigma \in \Omega$,

$$\Pr[\sigma \in R_i] = 2^{-i} \ .$$

Our estimator for $b_i = f(\sigma_{2^i})$ will be

$$m_i = \max_{\sigma \in R_i} f(\sigma) \ .$$

Lemma (Avoiding Overestimation is Easy)

$$\begin{aligned} \Pr[m_i > b_{i-2}] &\leq \Pr[R_i \cap \{\sigma_1, \sigma_2, \sigma_3, \ldots, \sigma_{2^{i-2}}\} \neq \emptyset] \\ &\leq 2^{i-2}\, 2^{-i} \qquad \text{Union Bound} \\ &= 1/4 \ . \end{aligned}$$

Getting Down to Business: Avoiding Underestimation

To avoid underestimation, i.e., to achieve $m_i \geq b_{i+2}$, we need

$$X_i = |R_i \cap \{\sigma_1, \sigma_2, \sigma_3, \ldots, \sigma_{2^{i+2}}\}| > 0 \ .$$

Observe that

$$\mathbb{E}X_i = 2^{i+2}2^{-i} = 4 \ .$$

So, we have:

- Two exponential-sized sets
  - $\{\sigma_1, \sigma_2, \sigma_3, \ldots, \sigma_{2^{i+2}}\}$
  - $|R_i| \sim 2^{n-i}$
- Which must intersect with probability $1/2 + \delta$
- While having expected intersection size 4

It Boils Down to This

We need to design a random set R such that:

- $\Pr[\sigma \in R] = 2^{-i}$ for every $\sigma \in \{0,1\}^n$ — e.g., a random subcube of dimension $n - i$
- Describing R can be done in $\mathrm{poly}(n)$ time — ditto
- For fixed $S \subseteq \{0,1\}^n$, the variance of $X = |R \cap S|$ is minimized

It Boils Down to This

We need to design a random set R such that:

- $\Pr[\sigma \in R] = 2^{-i}$ for every $\sigma \in \{0,1\}^n$ — e.g., a random subcube of dimension $n - i$
- Describing R can be done in $\mathrm{poly}(n)$ time — ditto
- For fixed $S \subseteq \{0,1\}^n$, the variance of $X = |R \cap S|$ is minimized

---

Minimizing variance amounts to minimizing

$$\sum_{\sigma \neq \sigma' \in S} \Pr[\sigma' \in R \mid \sigma \in R]$$

It Boils Down to This

We need to design a random set R such that:

- $\Pr[\sigma \in R] = 2^{-i}$ for every $\sigma \in \{0,1\}^n$ e.g., a random subcube of dimension $n-i$
- Describing R can be done in $\mathrm{poly}(n)$ time ditto
- For fixed $S \subseteq \{0,1\}^n$, the variance of $X = |R \cap S|$ is minimized

---

Minimizing variance amounts to minimizing

$$\sum_{\sigma \neq \sigma' \in S} \Pr[\sigma' \in R \mid \sigma \in R]$$

Since we know nothing about the geometry of S, a sensible goal is

$$\Pr[\sigma' \in R \mid \sigma \in R] = \Pr[\sigma' \in R]$$ Pairwise Independence

It Boils Down to This

We need to design a random set R such that:

- $\Pr[\sigma \in R] = 2^{-i}$ for every $\sigma \in \{0,1\}^n$ — e.g., a random subcube of dimension $n - i$
- Describing R can be done in $\text{poly}(n)$ time — ditto
- For fixed $S \subseteq \{0,1\}^n$, the variance of $X = |R \cap S|$ is minimized

---

Minimizing variance amounts to minimizing

$$\sum_{\sigma \neq \sigma' \in S} \Pr[\sigma' \in R \mid \sigma \in R]$$

Since we know nothing about the geometry of S, a sensible goal is

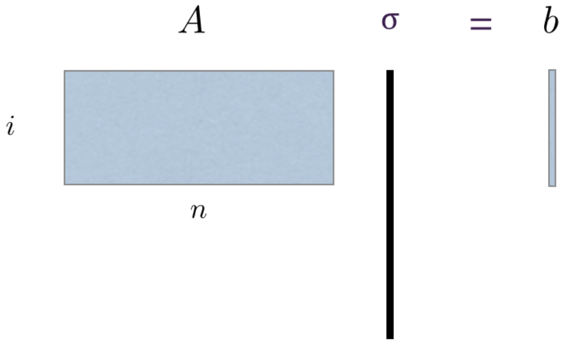
$$\Pr[\sigma' \in R \mid \sigma \in R] = \Pr[\sigma' \in R]$$ Pairwise Independence

How can this be reconciled with R being “simple to describe”?

Uncle Claude to the Rescue

Linear Error-Correcting Codes

Let
$$R = \{\sigma \in \{0,1\}^n : A\sigma = b\}$$
where both $A \in \{0,1\}^{i \times n}$ and $b \in \{0,1\}^i$ are uniformly random.



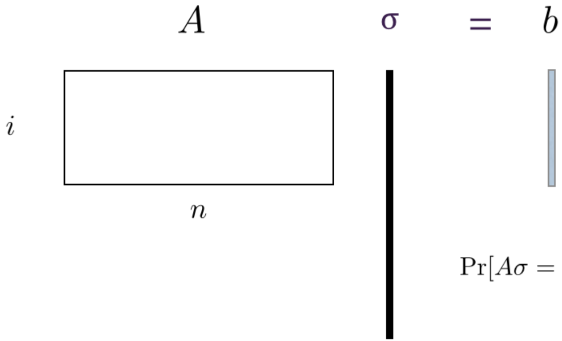
Uncle Claude to the Rescue

Linear Error-Correcting Codes

Let

$$R = \{\sigma \in \{0,1\}^n : A\sigma = b\}$$

where both $A \in \{0,1\}^{i \times n}$ and $b \in \{0,1\}^i$ are uniformly random.

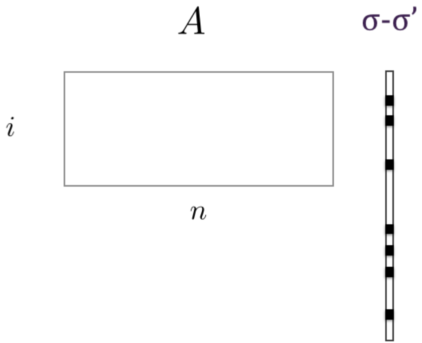


$$\Pr[A\sigma = b] = \left(\frac{1}{2}\right)^i$$

Uncle Claude to the Rescue

The probability that both σ, σ' are codewords is

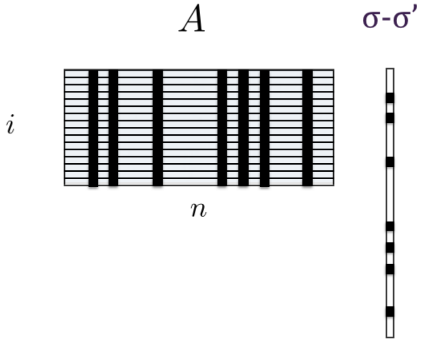
$$\Pr[A(\sigma' - \sigma) = 0 \wedge A\sigma = b] = \Pr[A(\sigma' - \sigma) = 0] \cdot \Pr[A\sigma = b] \ .$$



Uncle Claude to the Rescue

The probability that both σ, σ' are codewords is

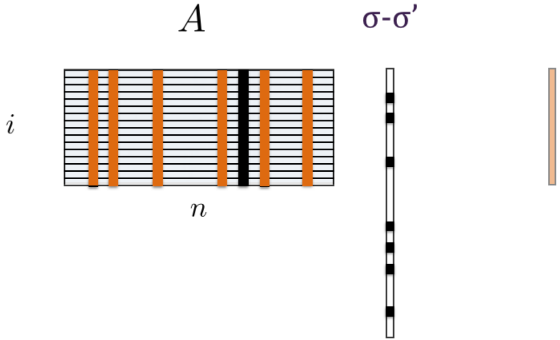
$$\Pr[A(\sigma' - \sigma) = 0 \wedge A\sigma = b] = \Pr[A(\sigma' - \sigma) = 0] \cdot \Pr[A\sigma = b] \ .$$



Uncle Claude to the Rescue

The probability that both σ, σ' are codewords is

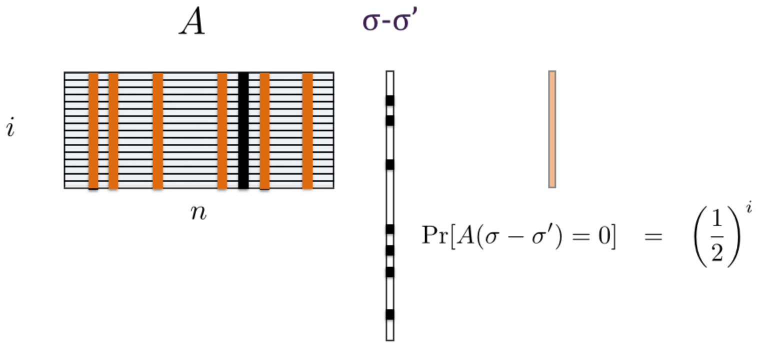
$$\Pr[A(\sigma' - \sigma) = 0 \wedge A\sigma = b] = \Pr[A(\sigma' - \sigma) = 0] \cdot \Pr[A\sigma = b] \ .$$



Uncle Claude to the Rescue

The probability that both σ, σ' are codewords is

$$\Pr[A(\sigma' - \sigma) = 0 \wedge A\sigma = b] = \Pr[A(\sigma' - \sigma) = 0] \cdot \Pr[A\sigma = b] \ .$$



Are We Done Yet?

Recapping

- Define R_i via i random parity constraints with $\sim n/2$ variables each
- Estimate b_i by maximizing f subject to the constraints

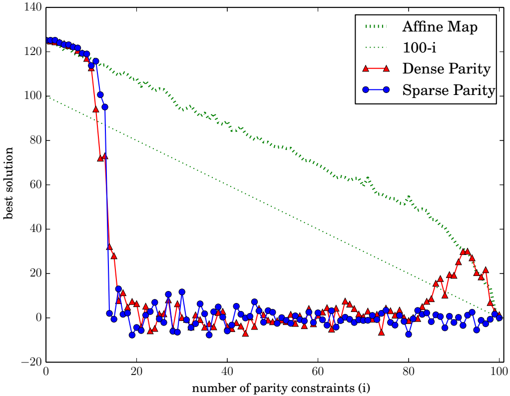
Are We Done Yet?

Recapping

- Define R_i via i random parity constraints with $\sim n/2$ variables each
- Estimate b_i by maximizing f subject to the constraints

$n = 10 \times 10$
Ferromagnetic
Ising Grid

Coupling Strengths
& External Fields
Near criticality



First Contribution: Random Affine Maps (Exploiting Linearity)

Let $G \in \{0,1\}^{(n-i)\times n}$ be the generator matrix of R, i.e.,

$$R = \left\{\sigma \in \{0,1\}^n : \sigma = xG \text{ and } x \in \{0,1\}^{n-i}\right\} .$$

First Contribution: Random Affine Maps (Exploiting Linearity)

Let $G \in \{0,1\}^{(n-i)\times n}$ be the generator matrix of R, i.e.,

$$R = \left\{\sigma \in \{0,1\}^n : \sigma = xG \text{ and } x \in \{0,1\}^{n-i}\right\} \ .$$

Instead of solving the constrained optimization problem

$$\max_{\substack{\sigma\in\{0,1\}^n \\ A\sigma=b}} f(\sigma) \ ,$$

First Contribution: Random Affine Maps (Exploiting Linearity)

Let $G \in \{0,1\}^{(n-i)\times n}$ be the generator matrix of R, i.e.,

$$R = \left\{\sigma \in \{0,1\}^n : \sigma = xG \text{ and } x \in \{0,1\}^{n-i}\right\} \; .$$

Instead of solving the constrained optimization problem

$$\max_{\substack{\sigma \in \{0,1\}^n \\ A\sigma = b}} f(\sigma) \; ,$$

solve the *unconstrained* optimization problem

$$\max_{x \in \{0,1\}^{n-i}} f(xG) \; ,$$

over the *exponentially* smaller set $\{0,1\}^{n-i}$.

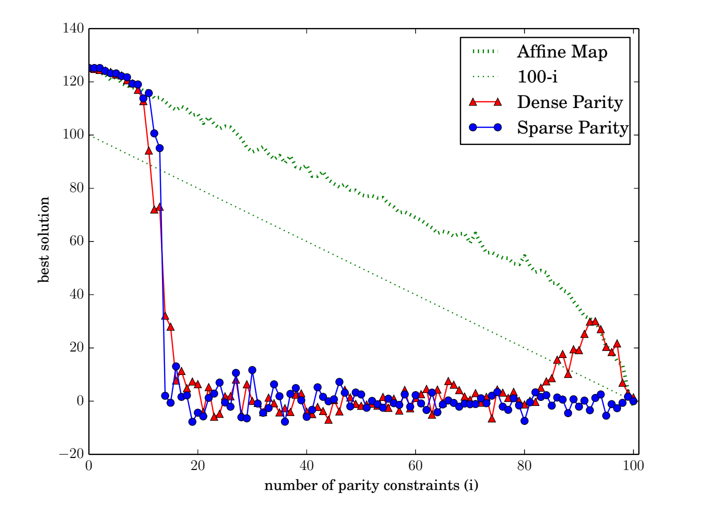
First Contribution: Random Affine Maps (Exploiting Linearity)

Let $G \in$

Instead

solve the

over the

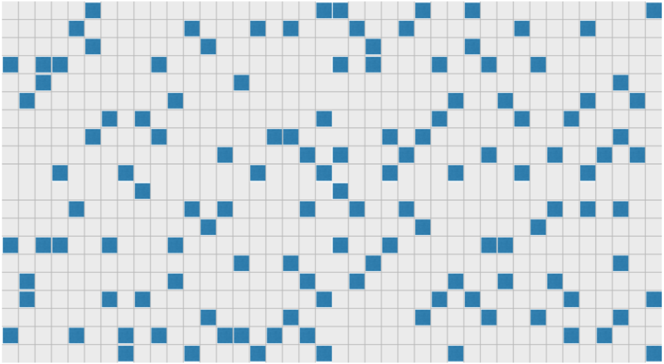


Fact

Working with an explicit representation of f is often crucial for efficient maximization

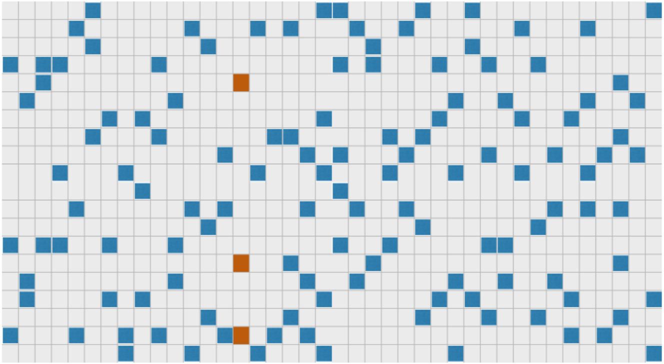
Second Contribution: Use Low Density Parity Check Codes

Extremely sparse equations but with variable regularity



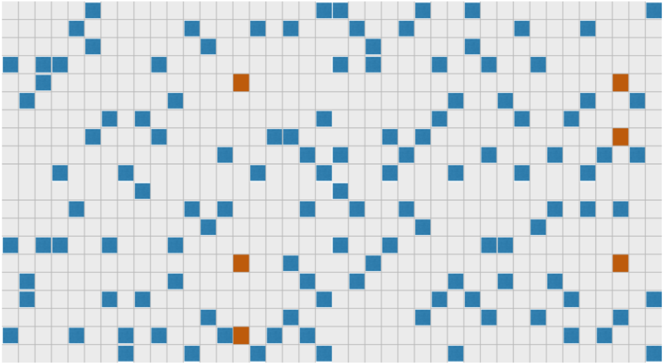
Second Contribution: Use Low Density Parity Check Codes

Extremely sparse equations but with variable regularity



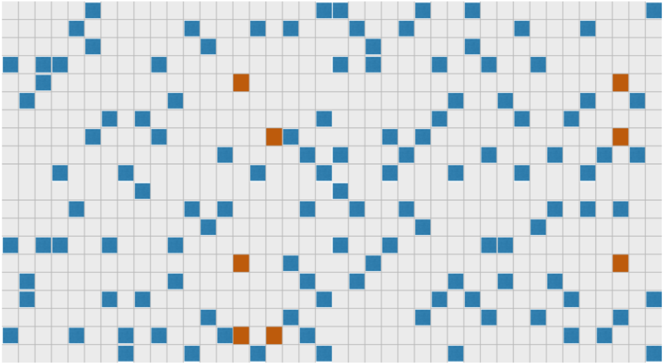
Second Contribution: Use Low Density Parity Check Codes

Extremely sparse equations but with variable regularity



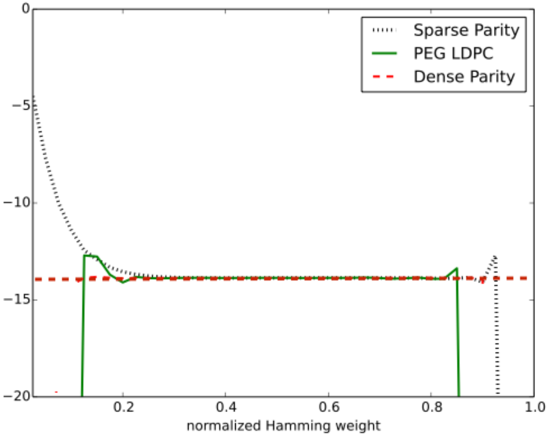
Second Contribution: Use Low Density Parity Check Codes

Extremely sparse equations but with variable regularity



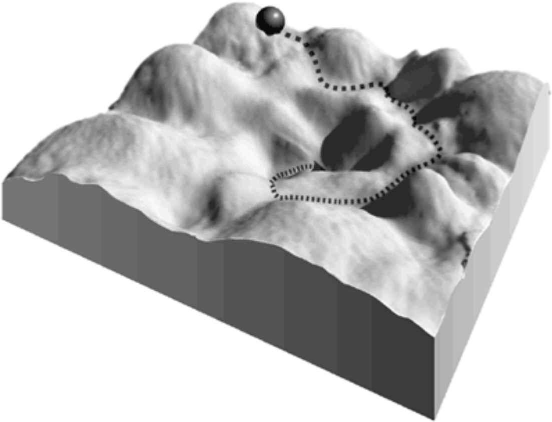
Second Contribution: Use Low Density Parity Check Codes

Extremely sparse equations but with variable regularity



Second Contribution: Use Low Density Parity Check Codes

Extremely sparse equations but with variable regularity



Second Contribution: Use Low Density Parity Check Codes

Extremely sparse equations but with variable regularity

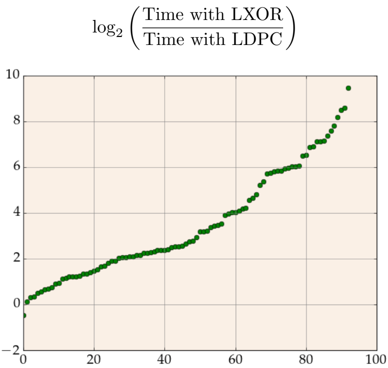
Second Contribution: Use Low Density Parity Check Codes

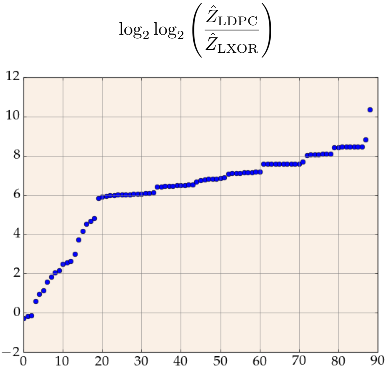
Extremely sparse equations but with variable regularity

- Scales to problems with several thousand variables
- Running-time when proving satisfiability comparable to original instance
- In all problems where ground truth is known:
  - Equally accurate as long XORs
  - 2-1000x faster

Second Contribution: Use Low Density Parity Check Codes

Extremely sparse equations but with variable regularity





Each point represents one CNF formula

Thanks!