Analysis of Noisy Gradient-Descent Bit Flipping (NGDBF

Using MATLAB and the PRISM Model Checking Tool

Eric Reiss
Utah State University

LDPC Codes and Trapping Sets

- Low-Density Parity Check (LDPC) codes were introduced by Gallager in 1963

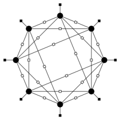


Figure 1: (8,8) Absorbing set that is dominant in the 802.3an 10GBASE-T LDPC Code [2].

LDPC Codes and Trapping Sets

- Low-Density Parity Check (LDPC) codes were introduced by Gallager in 1963
- LDPC codes are commonly represented as sparse Tanner graph

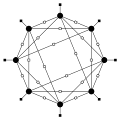


Figure 1: (8,8) Absorbing set that is dominant in the 802.3an 10GBASE-T LDPC Code [2].

LDPC Codes and Trapping Sets

- Low-Density Parity Check (LDPC) codes were introduced by Gallager in 1963
- LDPC codes are commonly represented as sparse Tanner graph
- Trapping sets are a sub-set of the graph that limit the performance of decoding algorithms, creating an error-floor

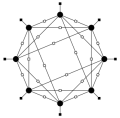


Figure 1: (8,8) Absorbing set that is dominant in the 802.3an 10GBASE-T LDPC Code [2].

LDPC Codes and Trapping Sets

- Low-Density Parity Check (LDPC) codes were introduced by Gallager in 1963
- LDPC codes are commonly represented as sparse Tanner graph
- Trapping sets are a sub-set of the graph that limit the performance of decoding algorithms, creating an error-floor
- Absorbing sets are a special case of a trapping sets that are stable in a bit flipping decoder [1]

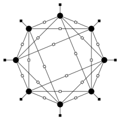


Figure 1: (8,8) Absorbing set that is dominant in the 802.3an 10GBASE-T LDPC Code [2].

Algorithm Overview

- NGDBF is part of a family of bit flipping decoding algorithms

Algorithm Overview

- NGDBF is part of a family of bit flipping decoding algorithms
- Improves upon the Gradient-Descent Bit Flipping (GDBF) Decoding Algorithm proposed by Wadayama et al. [3]

Algorithm Overview

- NGDBF is part of a family of bit flipping decoding algorithms
- Improves upon the Gradient-Descent Bit Flipping (GDBF) Decoding Algorithm proposed by Wadayama et al. [3]

  - GDBF gets “stuck” in local minima while decoding

Algorithm Overview

- NGDBF is part of a family of bit flipping decoding algorithms
- Improves upon the Gradient-Descent Bit Flipping (GDBF) Decoding Algorithm proposed by Wadayama et al. [3]

  - GDBF gets “stuck” in local minima while decoding

- NGDBF introduces psuedo-random noise to escape local minima

Algorithm Overview

- NGDBF is part of a family of bit flipping decoding algorithms
- Improves upon the Gradient-Descent Bit Flipping (GDBF) Decoding Algorithm proposed by Wadayama et al. [3]

  - GDBF gets “stuck” in local minima while decoding

- NGDBF introduces psuedo-random noise to escape local minima
- Algorithm steps [4]:

Algorithm Overview

- NGDBF is part of a family of bit flipping decoding algorithms
- Improves upon the Gradient-Descent Bit Flipping (GDBF) Decoding Algorithm proposed by Wadayama et al. [3]

    - GDBF gets “stuck” in local minima while decoding

- NGDBF introduces psuedo-random noise to escape local minima
- Algorithm steps [4]:

    - Let H be an $n \times m$ parity check matrix, N_i be the adjacency for bit i, M_j be the adjacency for parity check j

Algorithm Overview

- NGDBF is part of a family of bit flipping decoding algorithms
- Improves upon the Gradient-Descent Bit Flipping (GDBF) Decoding Algorithm proposed by Wadayama et al. [3]

  - GDBF gets "stuck" in local minima while decoding

- NGDBF introduces psuedo-random noise to escape local minima
- Algorithm steps [4]:

  - Let H be an $n \times m$ parity check matrix, N_i be the adjacency for bit i, M_j be the adjacency for parity check j
  - Given channel samples, $\vec{y}$, initialize $\vec{x}$ to be the $sign(\vec{y})$

Algorithm Overview

- NGDBF is part of a family of bit flipping decoding algorithms
- Improves upon the Gradient-Descent Bit Flipping (GDBF) Decoding Algorithm proposed by Wadayama et al. [3]

  - GDBF gets “stuck” in local minima while decoding

- NGDBF introduces psuedo-random noise to escape local minima
- Algorithm steps [4]:

  - Let H be an $n \times m$ parity check matrix, N_i be the adjacency for bit i, M_j be the adjacency for parity check j
  - Given channel samples, $\vec{y}$, initialize $\vec{x}$ to be the $sign(\vec{y})$
  - Compute the syndrome, $s_j = \prod_{i \in M_j} x_i$

Algorithm Overview

- NGDBF is part of a family of bit flipping decoding algorithms
- Improves upon the Gradient-Descent Bit Flipping (GDBF) Decoding Algorithm proposed by Wadayama et al. [3]

  - GDBF gets “stuck” in local minima while decoding

- NGDBF introduces psuedo-random noise to escape local minima
- Algorithm steps [4]:

  - Let H be an $n \times m$ parity check matrix, N_i be the adjacency for bit i, M_j be the adjacency for parity check j
  - Given channel samples, $\vec{y}$, initialize $\vec{x}$ to be the $sign(\vec{y})$
  - Compute the syndrome, $s_j = \prod_{i \in M_j} x_i$
  - Calculate the energy for bit i, $E_i = y_i x_i + w \sum_{j \in N_i} s_j + z_i$, where w is i.i.d white noise and z_i is a zero mean noise pertubation

Algorithm Overview

- NGDBF is part of a family of bit flipping decoding algorithms
- Improves upon the Gradient-Descent Bit Flipping (GDBF) Decoding Algorithm proposed by Wadayama et al. [3]

  - GDBF gets "stuck" in local minima while decoding

- NGDBF introduces psuedo-random noise to escape local minima
- Algorithm steps [4]:

  - Let H be an $n \times m$ parity check matrix, N_i be the adjacency for bit i, M_j be the adjacency for parity check j
  - Given channel samples, $\vec{y}$, initialize $\vec{x}$ to be the $sign(\vec{y})$
  - Compute the syndrome, $s_j = \prod_{i \in M_j} x_i$
  - Calculate the energy for bit i, $E_i = y_i x_i + w \sum_{j \in N_i} s_j + z_i$, where w is i.i.d white noise and z_i is a zero mean noise pertubation
  - Given a threshold θ flip bit i if $E_i < \theta$

Model Construction

- The Markov Chain structure used in the tool was proposed in [1]

Model Construction

- The Markov Chain structure used in the tool was proposed in [1]
- For a given (a,b) trapping set, the state space is described by the a-bit binary representation of the numbers from 0 to $2^a - 1$

Model Construction

- The Markov Chain structure used in the tool was proposed in [1]
- For a given (a,b) trapping set, the state space is described by the a-bit binary representation of the numbers from 0 to $2^a - 1$

    - For example, the (3,3) trapping set would have 8 states, 000 - 111

Model Construction

- The Markov Chain structure used in the tool was proposed in [1]
- For a given (a,b) trapping set, the state space is described by the a-bit binary representation of the numbers from 0 to $2^a - 1$
  - For example, the (3,3) trapping set would have 8 states, 000 - 111
  - A transition from 010 to 000 implies that the middle bit flipped

Model Construction

- The Markov Chain structure used in the tool was proposed in [1]
- For a given (a,b) trapping set, the state space is described by the a-bit binary representation of the numbers from 0 to $2^a - 1$
  - For example, the (3,3) trapping set would have 8 states, 000 - 111
  - A transition from 010 to 000 implies that the middle bit flipped
- Transistion

MATLAB Tool

Sources

- [1] Tasnuva Dissertation

Sources

- [1] Tasnuva Dissertation
- [2] T. Tithi, C. Winstead, and G. Sundararajan, Gopalakrishnan, “Decoding LDPC codes via Noisy Gradient Descent Bit-Flipping with Re-Decoding”, 2015.

Sources

- [1] Tasnuva Dissertation
- [2] T. Tithi, C. Winstead, and G. Sundararajan, Gopalakrishnan, “Decoding LDPC codes via Noisy Gradient Descent Bit-Flipping with Re-Decoding”, 2015.
- [3] T. wadayama, K. Nakamura, M. Yagita, Y. Funahashi, S. Usami, and I. Takumi, “Gradient descent bit flipping algorithms for decoding LDPC codes”, *Communications, IEEE Transactions on*, vol. 58, no. 6, pp. 1610-1614, 2010.

Sources

- [1] Tasnuva Dissertation
- [2] T. Tithi, C. Winstead, and G. Sundararajan, Gopalakrishnan, "Decoding LDPC codes via Noisy Gradient Descent Bit-Flipping with Re-Decoding", 2015.
- [3] T. wadayama, K. Nakamura, M. Yagita, Y. Funahashi, S. Usami, and I. Takumi, "Gradient descent bit flipping algorithms for decoding LDPC codes", *Communications, IEEE Transactions on*, vol. 58, no. 6, pp. 1610-1614, 2010.
- [4] NGDBF demo

Making slides

- I like HTML presentations based on Slidy, but many prefer PDF slides produced by LaTeX/Beamer.

Making slides

- I like HTML presentations based on Slidy, but many prefer PDF slides produced by LaTeX/Beamer.
- It can be a challenge to write code in either HTML or LaTeX.

Making slides

- I like HTML presentations based on Slidy, but many prefer PDF slides produced by LaTeX/Beamer.
- It can be a challenge to write code in either HTML or LaTeX.
- Markdown and Pandoc provide an easy text-based syntax for writing papers and presentations.

Making slides

- I like HTML presentations based on Slidy, but many prefer PDF slides produced by LaTeX/Beamer.
- It can be a challenge to write code in either HTML or LaTeX.
- Markdown and Pandoc provide an easy text-based syntax for writing papers and presentations.
- This template is designed to work with Pandoc to simultaneously:

Making slides

- I like HTML presentations based on Slidy, but many prefer PDF slides produced by LaTeX/Beamer.
- It can be a challenge to write code in either HTML or LaTeX.
- Markdown and Pandoc provide an easy text-based syntax for writing papers and presentations.
- This template is designed to work with Pandoc to simultaneously:
  - Produce html output based on slidy

Making slides

- I like HTML presentations based on Slidy, but many prefer PDF slides produced by LaTeX/Beamer.
- It can be a challenge to write code in either HTML or LaTeX.
- Markdown and Pandoc provide an easy text-based syntax for writing papers and presentations.
- This template is designed to work with Pandoc to simultaneously:
    - Produce html output based on slidy
    - Produce matching PDF output based on LaTeX/Beamer

Procedure

1. Prepare slides in a text document using Markdown syntax.

Procedure

1. Prepare slides in a text document using Markdown syntax.
2. Place any images in the `figures/` subdirectory.

Procedure

1. Prepare slides in a text document using Markdown syntax.
2. Place any images in the `figures/` subdirectory.
3. Edit the included `Makefile` to specify the presentation name and other details.

Procedure

1. Prepare slides in a text document using Markdown syntax.
2. Place any images in the `figures/` subdirectory.
3. Edit the included `Makefile` to specify the presentation name and other details.
4. Build the presentation by running `make`

Including Figures

To include a figure using Markdown, use this syntax:

```
![Optional figure caption.](figures/example.png){width=60%}
```

Result:

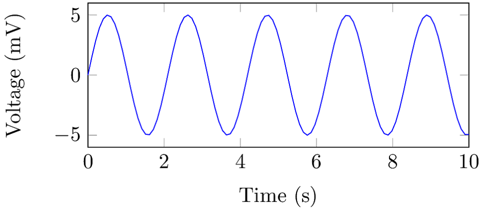


Figure 2: Optional figure caption.

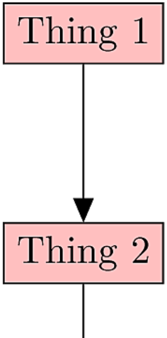
Two-Column Slides

Starting two-column mode:

Here is a column.

- Text.

And another column.



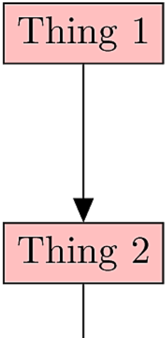
Two-Column Slides

Starting two-column mode:

Here is a column.

- Text.
- More text.

And another column.



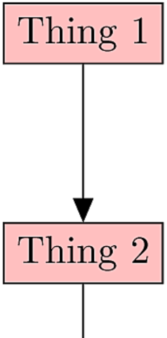
Two-Column Slides

Starting two-column mode:

Here is a column.

- Text.
- More text.
- Description.

And another column.



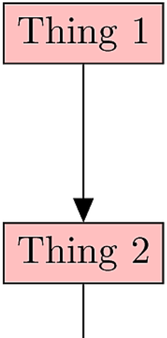
Two-Column Slides

Starting two-column mode:

Here is a column.

- Text.
- More text.
- Description.
- Discussion.

And another column.



Two-Column Slides

Starting two-column mode:

Here is a column.

- Text.

And another column.

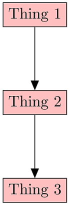


Figure 4: A tall figure.

Columns are now over.

Two-Column Slides

Starting two-column mode:

Here is a column.

- Text.
- More text.

And another column.

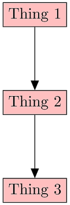


Figure 4: A tall figure.

Columns are now over.

Two-Column Slides

Starting two-column mode:

Here is a column.

- Text.
- More text.
- Description.

And another column.

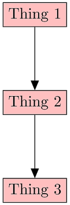


Figure 4: A tall figure.

Columns are now over.

Two-Column Slides

Starting two-column mode:

Here is a column.

- Text.
- More text.
- Description.
- Discussion.

And another column.

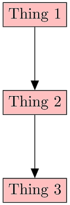


Figure 4: A tall figure.

Columns are now over.