Sequential Decoding of Convolutional Codes for Synchronization Errors

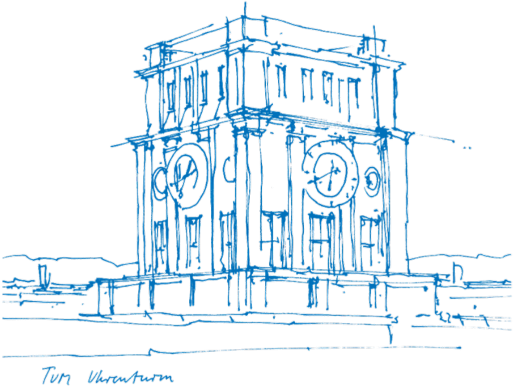
Anisha Banerjee

Joint work with

Andreas Lenz, Antonia Wachter-Zeh

Technical University of Munich

Institute for Communications Engineering



November 8, 2022

Outline

Introduction

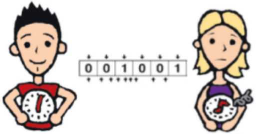
Sequential Decoding

Modifications for IDS Channels

Results & Conclusion

Motivation

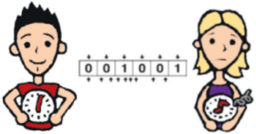
- Insertions and deletions occur
  - due to improper synchronization.



[HMG19] R. Heckel *et al.*, “A Characterization of the DNA Data Storage Channel,” en, *Scientific Reports*, vol. 9, no. 1, p. 9663, Dec. 2019

Motivation

- Insertions and deletions occur
  - due to improper synchronization.
  - in molecular storage paradigms, e.g. DNA storage [HMG19].

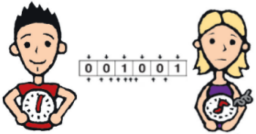


[HMG19] R. Heckel *et al.*, "A Characterization of the DNA Data Storage Channel," en, *Scientific Reports*, vol. 9, no. 1, p. 9663, Dec. 2019

Motivation

- Insertions and deletions occur
  - due to improper synchronization.
  - in molecular storage paradigms, e.g. DNA storage [HMG19].

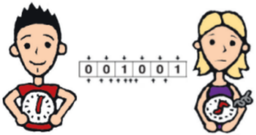
- May use **convolutional codes** for correction.



[HMG19] R. Heckel *et al.*, "A Characterization of the DNA Data Storage Channel," en, *Scientific Reports*, vol. 9, no. 1, p. 9663, Dec. 2019

Motivation

- Insertions and deletions occur
  - due to improper synchronization.
  - in molecular storage paradigms, e.g. DNA storage [HMG19].



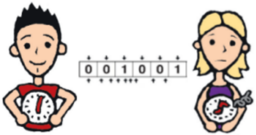
- May use **convolutional codes** for correction.

- Problem: High complexity of Viterbi & MAP decoders.

[HMG19] R. Heckel *et al.*, "A Characterization of the DNA Data Storage Channel," en, *Scientific Reports*, vol. 9, no. 1, p. 9663, Dec. 2019

Motivation

- Insertions and deletions occur
  - due to improper synchronization.
  - in molecular storage paradigms, e.g. DNA storage [HMG19].



- May use **convolutional codes** for correction.
- Problem: High complexity of Viterbi & MAP decoders.
- Solution: Use ***sequential decoders***!
  - Only examines ‘promising’ codewords.
  - Complexity independent of memory.

[HMG19] R. Heckel *et al.*, “A Characterization of the DNA Data Storage Channel,” en, *Scientific Reports*, vol. 9, no. 1, p. 9663, Dec. 2019

Channel Model

- Permits insertions, deletions and substitutions at random positions.

  $\rightarrow$ Denote as '**IDS channel**'.

[DM01] M. C. Davey and D. J. C. MacKay, "Reliable communication over channels with insertions, deletions, and substitutions," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 687–698, Feb. 2001

Channel Model

- Permits insertions, deletions and substitutions at random positions.

  $\rightarrow$ Denote as '**IDS channel**'.

- Specified by insertion, deletion and substitution probabilities, i.e., P_i, P_d and P_s.

[DM01] M. C. Davey and D. J. C. MacKay, "Reliable communication over channels with insertions, deletions, and substitutions," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 687–698, Feb. 2001

Channel Model

- Permits insertions, deletions and substitutions at random positions.

  $\rightarrow$ Denote as '**IDS channel**'.

- Specified by insertion, deletion and substitution probabilities, i.e., P_i, P_d and P_s.
- Modeled as **finite-state machine** [DM01].
  - Let channel input be $\boldsymbol{x} = (x_1, \ldots, x_T)$.
  - When x_i awaits transmission, four events possible.

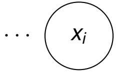
[DM01] M. C. Davey and D. J. C. MacKay, "Reliable communication over channels with insertions, deletions, and substitutions," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 687–698, Feb. 2001

Channel Model

- Permits insertions, deletions and substitutions at random positions.

  $\rightarrow$ Denote as '**IDS channel**'.

- Specified by insertion, deletion and substitution probabilities, i.e., P_i, P_d and P_s.
- Modeled as **finite-state machine** [DM01].
  - Let channel input be $\boldsymbol{x} = (x_1, \ldots, x_T)$.
  - When x_i awaits transmission, four events possible.



[DM01] M. C. Davey and D. J. C. MacKay, "Reliable communication over channels with insertions, deletions, and substitutions," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 687–698, Feb. 2001

Overview

Prior work

- [MT10] & [BF15] adapted Viterbi & MAP decoders to accommodate indels.
- [Gal61] & [MT02] investigated sequential decoding for IDS channels.

Our contributions

- New **decoding metric** for Fano's sequential decoder.
- Determination of '**cutoff rate**'.
  $\rightarrow$ Beyond this code rate, sequential decoder is computationally impractical.

[MT10] M. F. Mansour and A. H. Tewfik, "Convolutional decoding in the presence of synchronization errors," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 2, pp. 218–227, Feb. 2010

[BF15] V. Buttigieg and N. Farrugia, "Improved bit error rate performance of convolutional codes with synchronization errors," in *Proc. Int. Conf. Comm.*, London, Jun. 2015, pp. 4077–4082

[Gal61] R. G. Gallager, "Sequential decoding for binary channel with noise and synchronization errors," Lincoln Lab Group, Arlington, VA, USA, Tech. Rep., Sep. 1961

[MT02] M. F. Mansour and A. H. Tewfik, "Convolutional codes for channels with substitutions, insertions, and deletions," in *Proc. Gobal Commun. Conf.*, vol. 2, Taipei, Taiwan: IEEE, 2002, pp. 1051–1055

Convolutional Codes

- Encoding scheme with memory.

Convolutional Codes

- Encoding scheme with memory.
- For a (c, b, m) convolutional code, its encoder

Convolutional Codes

- Encoding scheme with memory.

- For a (c, b, m) convolutional code, its encoder
  - takes b input bits and

Convolutional Codes

- Encoding scheme with memory.

- For a (c, b, m) convolutional code, its encoder
  - takes b input bits and
  - produces c output bits,

Convolutional Codes

- Encoding scheme with memory.

- For a (c, b, m) convolutional code, its encoder
  - takes b input bits and
  - produces c output bits,
  - which depend on last $b(m+1)$ input bits.

Convolutional Codes

- Encoding scheme with memory.

- For a (c, b, m) convolutional code, its encoder
  - takes b input bits and
  - produces c output bits,
  - which depend on last $b(m+1)$ input bits.

Example

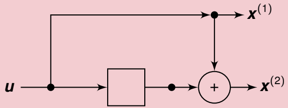


Figure: A $(2, 1, 1)$ binary convolutional encoder

Introduction

Sequential Decoding

Modifications for IDS Channels

Results & Conclusion

Sequential Decoding

- Applicable to tree codes $\supseteq$ convolutional codes.

Sequential Decoding

- Applicable to tree codes $\supseteq$ convolutional codes.
- Uses tree representation of codes.

Example

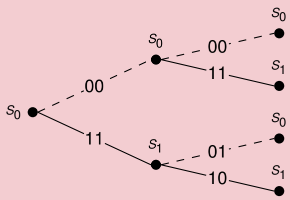


Figure: Code tree for a $(2, 1, 1)$ convolutional code.

Sequential Decoding

- Applicable to tree codes $\supseteq$ convolutional codes.
- Uses tree representation of codes.
- Only explores 'promising' paths in tree, unlike Viterbi.

  $\rightarrow$ Suboptimal, but faster!

Example

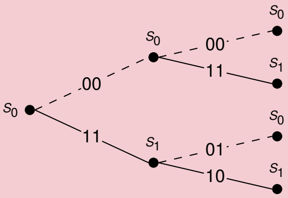


Figure: Code tree for a $(2, 1, 1)$ convolutional code.

Sequential Decoding

- Applicable to tree codes $\supseteq$ convolutional codes.
- Uses tree representation of codes.
- Only explores 'promising' paths in tree, unlike Viterbi.

  $\rightarrow$ Suboptimal, but faster!
- For channel output $\boldsymbol{y}$, assigns '**metrics**' to nodes.

Example

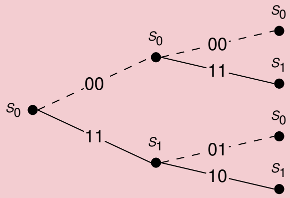


Figure: Code tree for a $(2, 1, 1)$ convolutional code.

Sequential Decoding

- Applicable to tree codes $\supseteq$ convolutional codes.
- Uses tree representation of codes.
- Only explores 'promising' paths in tree, unlike Viterbi.

  $\rightarrow$ Suboptimal, but faster!
- For channel output $\boldsymbol{y}$, assigns '**metrics**' to nodes.

Example

Say we receive $\boldsymbol{y} = 1010$ over a BSC with $P_s = 0.04$.

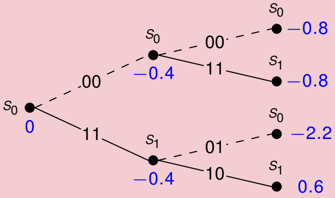
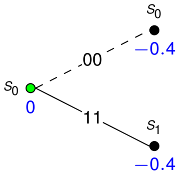


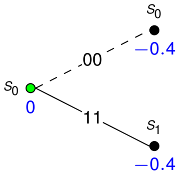
Figure: Code tree for a $(2, 1, 1)$ convolutional code.

Fano's Decoder



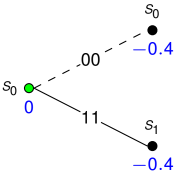
- **Decoder** starts at root; evaluates successors' metrics.

Fano's Decoder



- **Decoder** starts at root; evaluates successors' metrics.
- Chosen step size, $\Delta \leftarrow 0.5$.

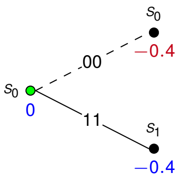
Fano's Decoder



$T = 0$

- **Decoder** starts at root; evaluates successors' metrics.
- Chosen step size, $\Delta \leftarrow 0.5$.
- Initialize threshold.

Fano's Decoder



$\Delta = 0.5$

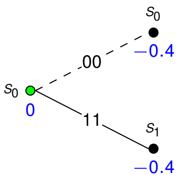
$T = 0$

$\mu_s = -0.4$

Algorithm

- Is best successor metric$\geq T$?

Fano's Decoder



$$T = 0$$

$$\mu_s = -0.4$$

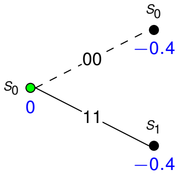
$$\mu_p = -\infty$$

$\Delta = 0.5$

Algorithm

- Is best successor metric$\geq T$?
  - NO: Is predecessor metric $\geq T$?

Fano's Decoder



$\Delta = 0.5$

$T = -0.5$

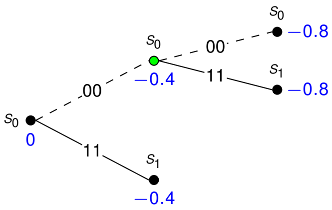
$\mu_s = -0.4$

$\mu_p = -\infty$

Algorithm

- Is best successor metric$\geq$ T?
  - NO: Is predecessor metric $\geq$ T?
    - No: Lowers T by Δ.

Fano's Decoder



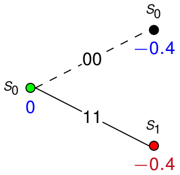
$$\Delta = 0.5$$

$$T = -0.5$$
$$\mu_s = -0.8$$
$$\mu_p = 0$$
$$\mu_c = -0.4$$

Algorithm

- Is best successor metric $\geq T$?
  - NO: Is predecessor metric $\geq T$?
    - No: Lowers T by Δ.
  - YES: Move forward. Evaluate successors.

Fano's Decoder



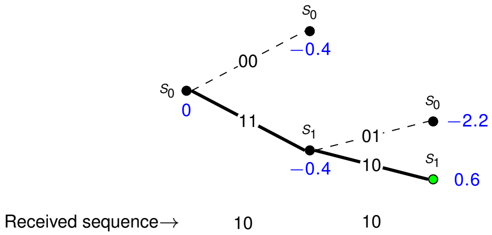
$\Delta = 0.5$

$$T = -0.5$$
$$\mu_s = -0.4$$
$$\mu_p = -\infty$$
$$\mu_c = 0$$

Algorithm

- Is best successor metric$\geq T$?
  - NO: Is predecessor metric $\geq T$?
    - No: Lowers T by Δ.
    - Yes: Move back. Pick next best.
  - YES: Move forward. Evaluate successors.

Fano's Algorithm

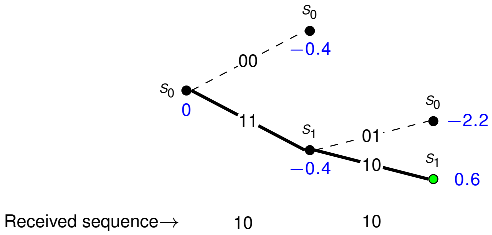


$$T = -0.5$$
$$\mu_p = -0.4$$
$$\mu_c = 0.6$$
$$\mu_s = -$$

Summary

- If $\mu_s \geq T$: decoder moves forward.

Fano's Algorithm

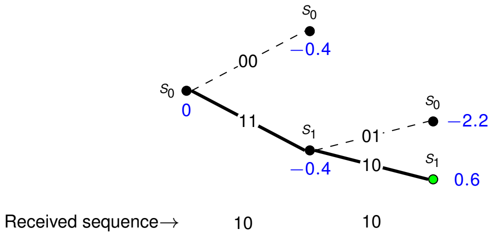


$$T = -0.5$$
$$\mu_p = -0.4$$
$$\mu_c = 0.6$$
$$\mu_s = -$$

Summary

- If $\mu_s \geq T$: decoder moves forward.
- When error occurs $\rightarrow$ metric dip across branch.

Fano's Algorithm

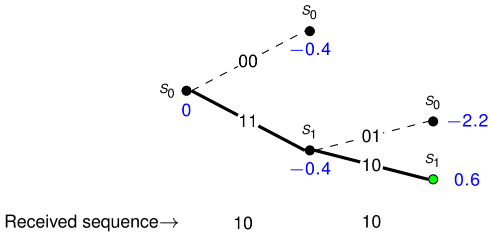


$$T = -0.5$$
$$\mu_p = -0.4$$
$$\mu_c = 0.6$$
$$\mu_s = -$$

Summary

- If $\mu_s \geq T$: decoder moves forward.
- When error occurs $\rightarrow$ metric dip across branch.
- May cause $\mu_s < T$.

Fano's Algorithm



$$T = -0.5$$

$$\mu_p = -0.4$$

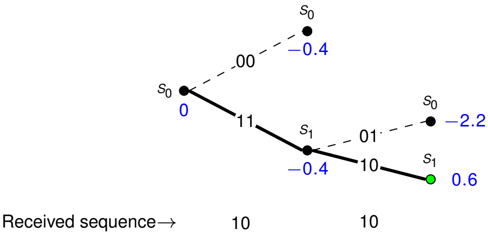
$$\mu_c = 0.6$$

$$\mu_s = -$$

Summary

- If $\mu_s \geq T$: decoder moves forward.
- When error occurs $\rightarrow$ metric dip across branch.
- May cause $\mu_s < T$.
  - Backtracks to find nodes above T.

Fano's Algorithm



$$T = -0.5$$
$$\mu_p = -0.4$$
$$\mu_c = 0.6$$
$$\mu_s = -$$

Summary

- If $\mu_s \geq T$: decoder moves forward.
- When error occurs $\rightarrow$ metric dip across branch.
- May cause $\mu_s < T$.
  - Backtracks to find nodes above T.
  - When none exist, lowers T & repeats.

Introduction

Sequential Decoding

Modifications for IDS Channels

Results & Conclusion

Tree for IDS Channels

- Aim: Modify code tree to account for insertions & deletions.

[BF15] V. Buttigieg and N. Farrugia, “Improved bit error rate performance of convolutional codes with synchronization errors,” in *Proc. Int. Conf. Comm.*, London, Jun. 2015, pp. 4077–4082

[DM01] M. C. Davey and D. J. C. MacKay, “Reliable communication over channels with insertions, deletions, and substitutions,” *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 687–698, Feb. 2001

Tree for IDS Channels

- Aim: Modify code tree to account for insertions & deletions.
- Idea: Use '**drift state**' [DM01, BF15].

  $\rightarrow$ Let d_t = #received bits − #transmitted bits, at time t.

[BF15] V. Buttigieg and N. Farrugia, "Improved bit error rate performance of convolutional codes with synchronization errors," in *Proc. Int. Conf. Comm.*, London, Jun. 2015, pp. 4077–4082

[DM01] M. C. Davey and D. J. C. MacKay, "Reliable communication over channels with insertions, deletions, and substitutions," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 687–698, Feb. 2001

Tree for IDS Channels

- Aim: Modify code tree to account for insertions & deletions.
- Idea: Use '**drift state**' [DM01, BF15].

  $\rightarrow$ Let d_t = #received bits − #transmitted bits, at time t.
- In new code tree, each state must combine encoder & drift states.

$$v_t = (s_t, d_t)$$

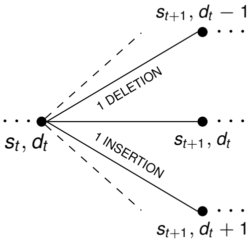


Figure: Hidden Markov Model seen by receiver

Tree for IDS Channels

- Aim: Modify code tree to account for insertions & deletions.
- Idea: Use '**drift state**' [DM01, BF15].

  $\rightarrow$ Let d_t = #received bits − #transmitted bits, at time t.

- In new code tree, each state must combine encoder & drift states.

$$v_t = (s_t, d_t)$$

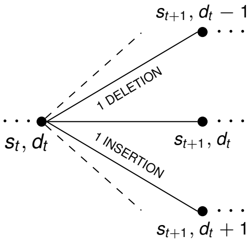


Figure: Hidden Markov Model seen by receiver

Example

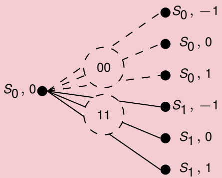


Figure: Code tree for a $(2, 1, 1)$ convolutional code.

Tree for IDS Channels

- Aim: Modify code tree to account for insertions & deletions.
- Idea: Use '**drift state**' [DM01, BF15].

  $\rightarrow$ Let d_t = #received bits − #transmitted bits, at time t.

- In new code tree, each state must combine encoder & drift states.

$$v_t = (s_t, d_t)$$

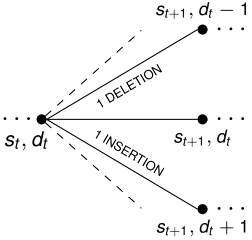


Figure: Hidden Markov Model seen by receiver

Example

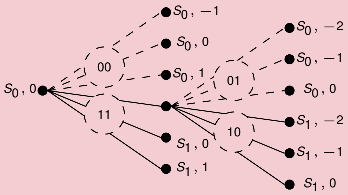


Figure: Code tree for a (2, 1, 1) convolutional code.

Decoding Metric

- Consider received sequence $\boldsymbol{y}$.

Decoding Metric

- Consider received sequence $\boldsymbol{y}$.
- Metric of any node v_t should

Decoding Metric

- Consider received sequence $\boldsymbol{y}$.
- Metric of any node v_t should
  - minimize probability of choosing wrong successor.

Decoding Metric

- Consider received sequence $\boldsymbol{y}$.
- Metric of any node v_t should
  - minimize probability of choosing wrong successor.
  - quantify closeness of $\boldsymbol{y}$ with path to v_t from root.

Decoding Metric

- Consider received sequence $\boldsymbol{y}$.
- Metric of any node v_t should
  - minimize probability of choosing wrong successor.
  - quantify closeness of $\boldsymbol{y}$ with path to v_t from root.
- Say v_t is reached from tree root via
  - encoder states $\boldsymbol{s} = (s_0, \ldots, s_t)$,
  - drift states $\boldsymbol{d} = (d_0 = 0, d_1 \ldots, d_t)$.

Decoding Metric

- Consider received sequence $\boldsymbol{y}$.
- Metric of any node v_t should
  - minimize probability of choosing wrong successor.
  - quantify closeness of $\boldsymbol{y}$ with path to v_t from root.
- Say v_t is reached from tree root via
  - encoder states $\boldsymbol{s} = (s_0, \dots, s_t)$,
  - drift states $\boldsymbol{d} = (d_0 = 0, d_1 \dots, d_t)$.

$$\begin{aligned}\mu(v_t) &= \log P(v_t|\boldsymbol{y}) \\ &= \log P(\boldsymbol{s}, \boldsymbol{d}|\boldsymbol{y})\end{aligned}$$

Decoding Metric

- Consider received sequence $\boldsymbol{y}$.
- Metric of any node v_t should
  - minimize probability of choosing wrong successor.
  - quantify closeness of $\boldsymbol{y}$ with path to v_t from root.
- Say v_t is reached from tree root via
  - encoder states $\boldsymbol{s} = (s_0, \ldots, s_t)$,
  - drift states $\boldsymbol{d} = (d_0 = 0, d_1 \ldots, d_t)$.

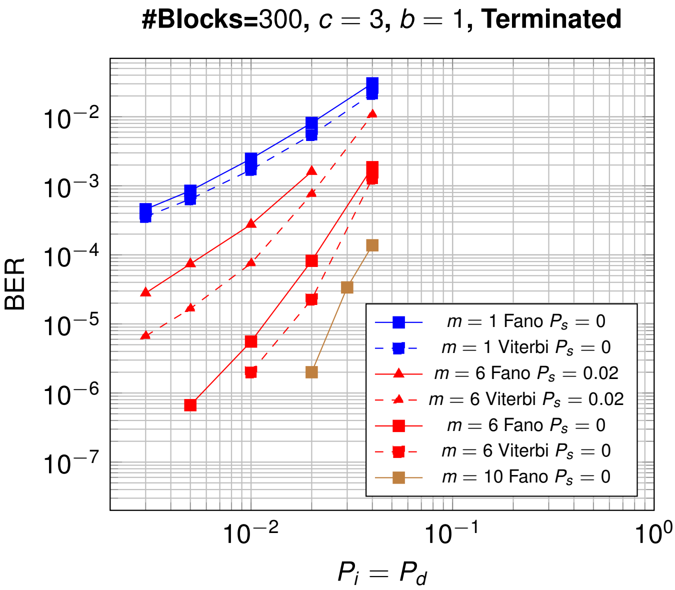
$$\begin{aligned}\mu(v_t) &= \log P(v_t|\boldsymbol{y}) \\ &= \log P(\boldsymbol{s}, \boldsymbol{d}|\boldsymbol{y})\end{aligned}$$

- Special case: When no $P_i = P_d = 0$, ignore drift states.

$$\mu(v_t) = \log P(\boldsymbol{s}|\boldsymbol{y})$$

$\rightarrow$ Original Fano metric for substitution-only channels!

Bit Error Rate



Computational Performance

- Recall:

$$\text{Complexity} \propto \#\text{Branch metric computations}$$

Computational Performance

- Recall:

$$\text{Complexity} \propto \#\text{Branch metric computations}$$

- Decoding complexity for
  - Viterbi:

$$\#\text{Edges in trellis}$$

Computational Performance

- Recall:

$$\text{Complexity} \propto \#\text{Branch metric computations}$$

- Decoding complexity for
  - Viterbi:

$$\#\text{Nodes in trellis} \cdot \#\text{Edges}_{\text{out}} \text{ per node} \qquad \text{[Constant]}$$

Computational Performance

- Recall:

$$\text{Complexity} \propto \#\text{Branch metric computations}$$

- Decoding complexity for
  - Viterbi:

    $$\#\text{Nodes in trellis} \cdot \#\text{Edges}_{\text{out}} \text{ per node} \qquad \text{[Constant]}$$

  - Fano's:

    $$\#\text{Steps forward} \cdot \#\text{Edges}_{\text{out}} \text{ per node} \qquad \text{[Variable]}$$

Computational Performance

- Recall:

$$\text{Complexity} \propto \#\text{Branch metric computations}$$

- Decoding complexity for
  - Viterbi:

    $$\#\text{Nodes in trellis} \cdot \#\text{Edges}_{\text{out}} \text{ per node} \qquad \text{[Constant]}$$

  - Fano's:

    $$\#\text{Steps forward} \cdot \#\text{Edges}_{\text{out}} \text{ per node} \qquad \text{[Variable]}$$

    $\rightarrow$ Note: Independent of memory!

Computational Performance

- Recall:

$$\text{Complexity} \propto \#\text{Branch metric computations}$$

- Decoding complexity for
  - Viterbi:
  $$\#\text{Nodes in trellis} \cdot \#\text{Edges}_{\text{out}} \text{ per node} \qquad \text{[Constant]}$$
  - Fano's:
  $$\#\text{Steps forward} \cdot \#\text{Edges}_{\text{out}} \text{ per node} \qquad \text{[Variable]}$$

  $\rightarrow$ Note: Independent of memory!

- Define

$$\text{Complexity reduction factor} = \frac{\text{Complexity of Viterbi}}{\text{Mean complexity of Fano's}}$$

Computational Performance

- Recall:

$$\text{Complexity} \propto \#\text{Branch metric computations}$$

- Decoding complexity for
  - Viterbi:

$$\#\text{Nodes in trellis} \cdot \#\text{Edges}_{\text{out}} \text{ per node} \qquad \text{[Constant]}$$

  - Fano's:

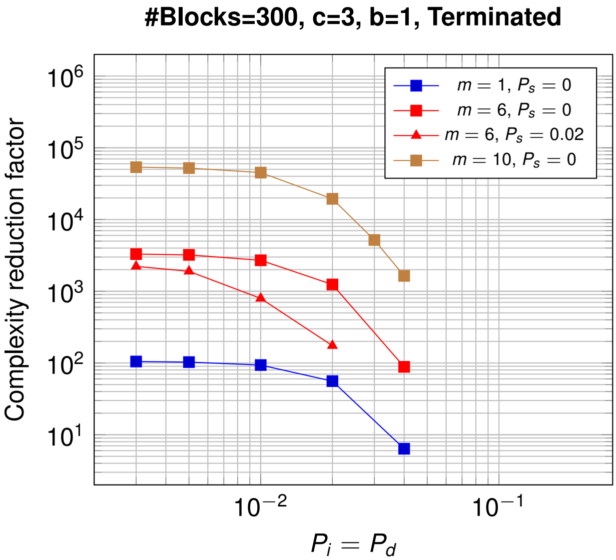
$$\#\text{Steps forward} \cdot \#\text{Edges}_{\text{out}} \text{ per node} \qquad \text{[Variable]}$$

$\rightarrow$ Note: Independent of memory!

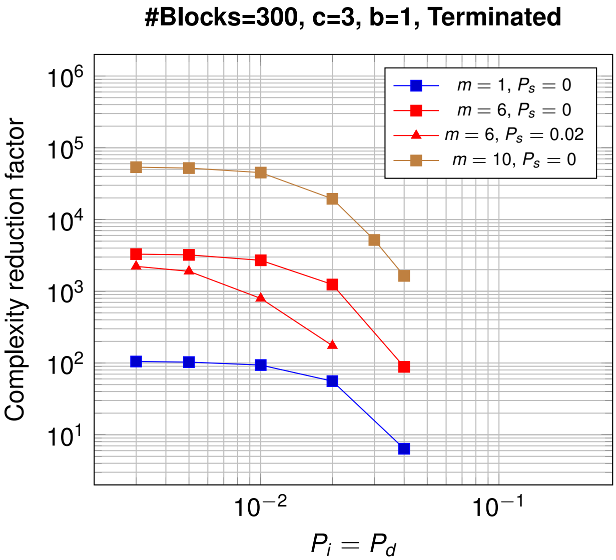
- Define

$$\text{Complexity reduction factor} = \frac{\text{Complexity of Viterbi}}{\text{Mean complexity of Fano's}} = \frac{\#\text{Nodes in trellis}}{\#\text{Mean steps forward}}$$

Simulated Complexity Comparison

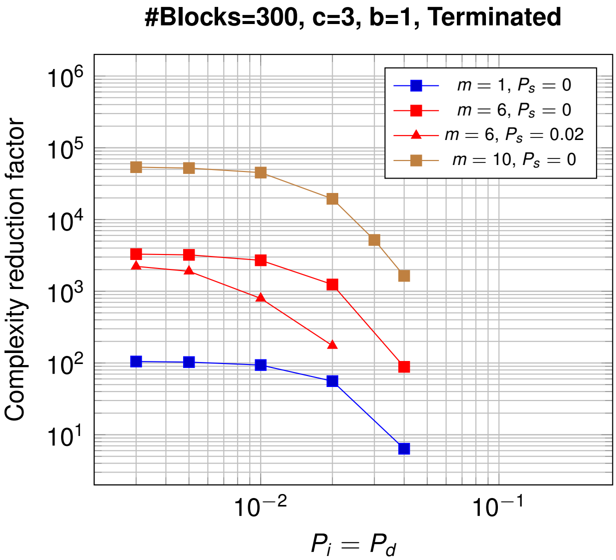


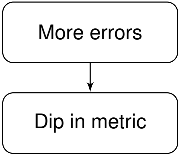
Simulated Complexity Comparison



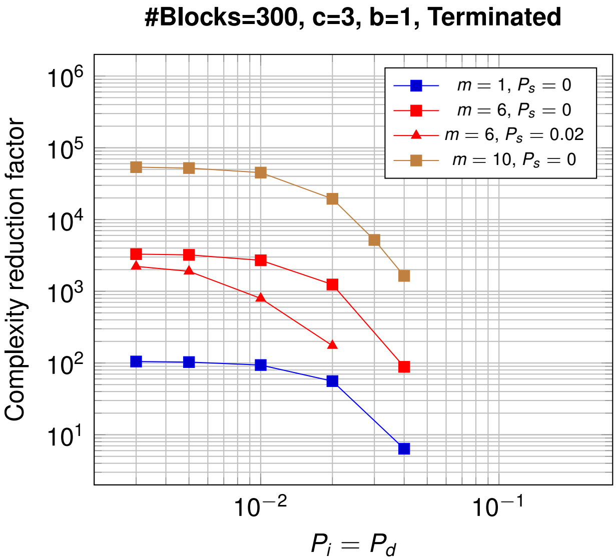
More errors

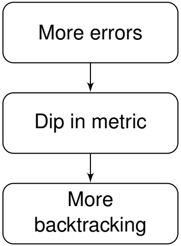
Simulated Complexity Comparison



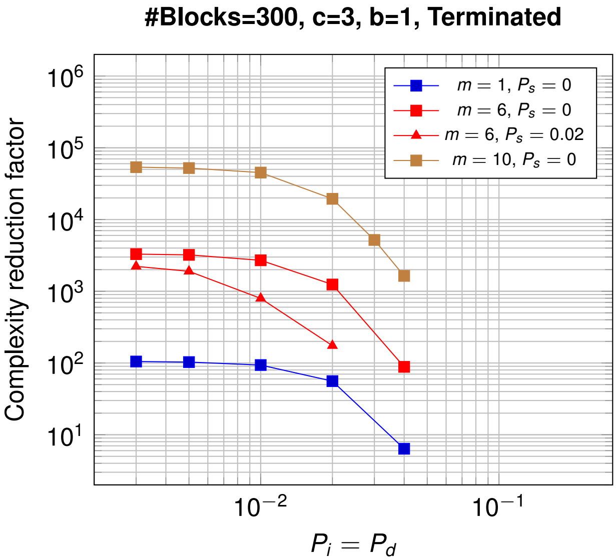


Simulated Complexity Comparison





Simulated Complexity Comparison



More errors

↓

Dip in metric

↓

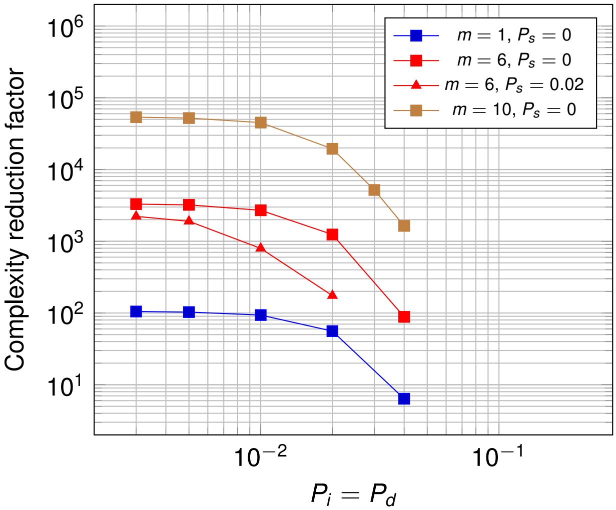
More backtracking

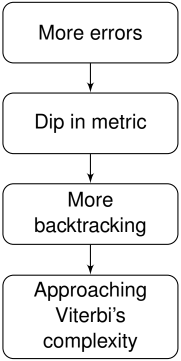
↓

Approaching Viterbi's complexity

Simulated Complexity Comparison

#Blocks=300, c=3, b=1, Terminated





Question: When is Fano's algorithm no longer practical to use?

Computational Cutoff Rate

- Beyond cutoff rate R_0, sequential decoder is computational impractical.

[JZ15] R. Johannesson and K. S. Zigangirov, “Sequential decoding,” in *Fundamentals of Convolutional Coding*. Hoboken, New Jersey: John Wiley & Sons, Inc., 2015, pp. 425–484

Computational Cutoff Rate

- Beyond cutoff rate R_0, sequential decoder is computational impractical.
- For rates $< R_0$,
  - Complexity of decoding one frame grows **linearly** with #blocks.

[JZ15] R. Johannesson and K. S. Zigangirov, "Sequential decoding," in *Fundamentals of Convolutional Coding*. Hoboken, New Jersey: John Wiley & Sons, Inc., 2015, pp. 425–484

Computational Cutoff Rate

- Beyond cutoff rate R_0, sequential decoder is computational impractical.
- For rates $< R_0$,
  - Complexity of decoding one frame grows **linearly** with #blocks.
- For rates $> R_0$,
  - Too much backtracking.
  - Complexity of decoding one frame grows **exponentially** with #blocks..

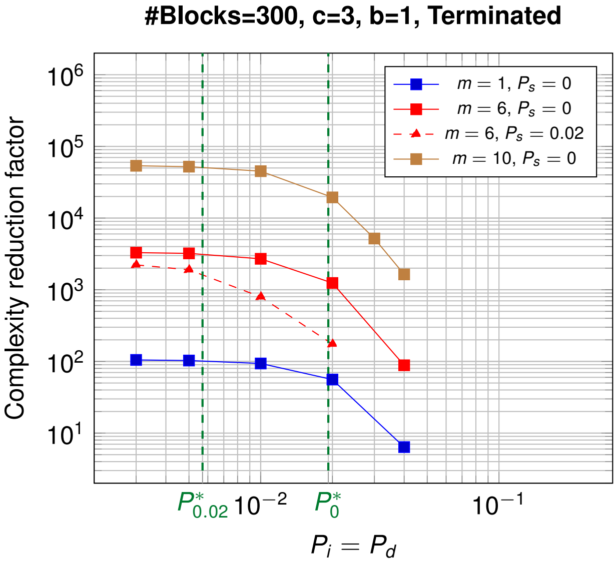
[JZ15] R. Johannesson and K. S. Zigangirov, “Sequential decoding,” in *Fundamentals of Convolutional Coding*. Hoboken, New Jersey: John Wiley & Sons, Inc., 2015, pp. 425–484

Computational Cutoff Rate

- Beyond cutoff rate R_0, sequential decoder is computational impractical.
- For rates $< R_0$,
  - Complexity of decoding one frame grows **linearly** with #blocks.
- For rates $> R_0$,
  - Too much backtracking.
  - Complexity of decoding one frame grows **exponentially** with #blocks..
- By extending methods in [JZ15], we can compute this!

[JZ15] R. Johannesson and K. S. Zigangirov, “Sequential decoding,” in *Fundamentals of Convolutional Coding*. Hoboken, New Jersey: John Wiley & Sons, Inc., 2015, pp. 425–484

Simulated Complexity Comparison



P^*_0 & $P^*_{0.02}$ mark cutoff rate operation for code $(3, 1)$ at $P_s = 0$ and $P_s = 0.02$ respectively.

Conclusion

Summary

- New **decoding metric** for Fano's algorithm in IDS channels.
- Determination of computational **cutoff rate**.

Future work

- Error probability analysis.
- Branching process techniques for complexity analysis.

Conclusion

Summary

- New **decoding metric** for Fano's algorithm in IDS channels.
- Determination of computational **cutoff rate**.

Future work

- Error probability analysis.
- Branching process techniques for complexity analysis.

Thank you!

References

[1] R. Heckel, G. Mikutis, and R. N. Grass, “A Characterization of the DNA Data Storage Channel,” en, *Scientific Reports*, vol. 9, no. 1, p. 9663, Dec. 2019.

[2] M. C. Davey and D. J. C. MacKay, “Reliable communication over channels with insertions, deletions, and substitutions,” *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 687–698, Feb. 2001.

[3] M. F. Mansour and A. H. Tewfik, “Convolutional decoding in the presence of synchronization errors,” *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 2, pp. 218–227, Feb. 2010.

[4] V. Buttigieg and N. Farrugia, “Improved bit error rate performance of convolutional codes with synchronization errors,” in *Proc. Int. Conf. Comm.*, London, Jun. 2015, pp. 4077–4082.

[5] R. G. Gallager, “Sequential decoding for binary channel with noise and synchronization errors,” Lincoln Lab Group, Arlington, VA, USA, Tech. Rep., Sep. 1961.

[6] M. F. Mansour and A. H. Tewfik, “Convolutional codes for channels with substitutions, insertions, and deletions,” in *Proc. Gobal Commun. Conf.*, vol. 2, Taipei, Taiwan: IEEE, 2002, pp. 1051–1055.

[7] R. Johannesson and K. S. Zigangirov, “Sequential decoding,” in *Fundamentals of Convolutional Coding*. Hoboken, New Jersey: John Wiley & Sons, Inc., 2015, pp. 425–484.