Coding theory in Sage
Groupe utilisateurs parisiens de Sage

David Lucas
Daniel Augot, Johan Nielsen, Clément Pernet

Inria Saclay

2015/07/01
Foreword

- branch sage version 6.8beta6
- git pull https://lucasdavid@bitbucket.org/lucasdavid/sage_coding_project.git
- sage -b
Outline

1. A quick overview of coding theory
2. Coding theory in computer algebra systems
3. ACTIS project
Outline

1. A quick overview of coding theory
2. Coding theory in computer algebra systems
3. ACTIS project
The communication problem

Sender \[ m = (010101) \]
Channel \[ y = (110111) \]
Receiver

Noise

Other applications:
- data storage
- public key cryptography
- private key cryptography
- combinatorics
- theoretical computer science
- distributed systems
The communication problem

Sender \( m \) \rightarrow \text{Encoder} \rightarrow \text{Channel} \rightarrow \text{Noise} \rightarrow \text{Decoder} \rightarrow \text{Receiver} \( m' \)

- \( m = (010101) \)
- \( c = (000111000111000111) \)
- \( y = (100111000111100111) \)
- \( m = (010101) \)

Other applications:
- data storage
- public key cryptography
- private key cryptography
- combinatorics
- theoretical computer science
- distributed systems
The communication problem

Other applications:

- data storage
- public key cryptography
- private key cryptography
- combinatorics
- theoretical computer science
- distributed systems
Linear Codes: definition

$(n, k)$-linear code

$C$ is a linear subspace of $\mathbb{F}_q^n$ of dimension $k$

- elements of $C$ can be far apart: minimum distance ($d$)
- get closer to an element (codeword): decoding problem
- example: minimum distance decoding $\lfloor \frac{d}{2} \rfloor$
- Problem: these are NP-hard
- (demonstration)
Beyond linear Codes

- $C$ can be studied as:
  - "random" linear vector spaces
  - specific families (algebraic point of view)
A family of linear codes: Reed-Solomon codes

\((n, k)\)-Reed-Solomon code

\[ C = \{ (f(\alpha_1), \ldots, f(\alpha_n)) \mid f \in \mathbb{F}[X]_{<k}, (\alpha_1, \ldots, \alpha_n) \in \mathbb{F}^n \} \]

- Minimum distance computation is trivial: \( d = n - k + 1 \)
  - \( f \) has at most \( k - 1 \) roots
- Decoding is quasi-linear in code length
- (demonstration)
Decoding algorithms

- RS codes have many decoding algorithms:
  - Peterson (1960)
  - Berlekamp-Massey (1967)
  - Berlekamp-Welch (1986)
  - Guruswami-Sudan (1999)
  - Gao (2002)
  - Power decoding (2006)
  - Wu (2008)
  - And multiple speed improvements
Outline

1. A quick overview of coding theory

2. Coding theory in computer algebra systems

3. ACTIS project
What a user would expect

- A system that provides tools for his own field:
  - information theory
  - combinatorics
  - cryptography
  - decoding
- Asymptotically fast
- Easy to use: intuitive
- Why do we want coding theory into Sage?
  - Teaching
  - Experimenting
State of CT in Sage

- A lot of methods related to combinatorics
- A lot of methods to manipulate linear codes
- Structure of code families is not kept
- Exhaustive search: only generic algorithms
- Very few methods related to decoding
- Nothing for performing simulation and experiments
  → Channels
Outline

1. A quick overview of coding theory
2. Coding theory in computer algebra systems
3. ACTIS project
A quick overview of coding theory
Coding theory in computer algebra systems
ACTIS project

Roadmap
A word on the development

- We needed a sandbox to experiment
- So we built a fork of Sage
- Short-term integration into Sage
- Long-term: might kill this fork
Design

- We want to remember families of code
  - code families are separate classes
- Multiple points of view supported
  - multiple encoders and decoders
  - but should still be easy
  - heavy use of default implementation
Encoders and Decoders

- Objects associated with a code class
- Managed by a registration structure
  - Each code has a dictionary of encoders and decoders
- You just want any encoder/decoder?
  - There is a method for that!
- (demonstration)
Communication channels

- Idea: emulate a real communication channel
- Facilitate experimentation and simulation
- So far, we have:
  - Static error rate
  - Error-erasure
  - Lot more to come!
- (demonstration)
A few other features

- Concatenated codes
  - two encoders
  - one decoder

- Cyclic codes
  - including three different ways to build a cyclic code
  - two encoders
Interested in the project?

- trac tickets #18376 and #18813 need you!
- "structural" tickets, more on the engineering side
Useful links

- https://bitbucket.org/lucasdavid/sage_coding_project/wiki/Home
- https://groups.google.com/forum/#!forum/sage-coding-theory
- http://trac.sagemath.org/ticket/18376
- http://trac.sagemath.org/ticket/18813