Coding theory in Sage

Sage days 66

David Lucas
Daniel Augot, Johan Nielsen, Clément Pernet

Inria Saclay

2015/04/02

Foreword

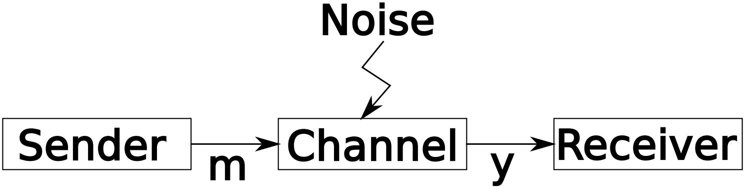
- branch sage version 6.6rc1
- git pull `https://lucasdavid@bitbucket.org/lucasdavid/sage_coding_project.git`
- sage -b

Outline

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

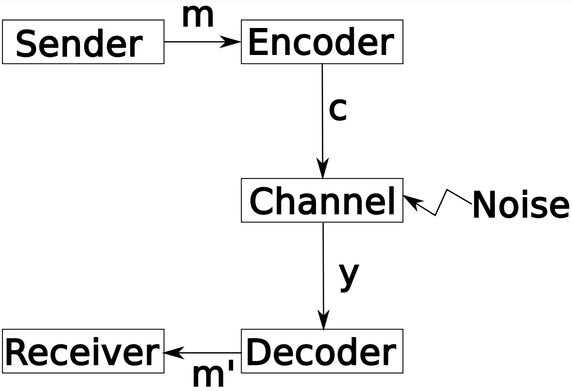
Outline

The communication problem



- $m = (010101)$
- $y = (110111)$

The communication problem



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

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

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

- elements of $\mathcal{C}$ 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

- $\mathcal{C}$ 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

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

- 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

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

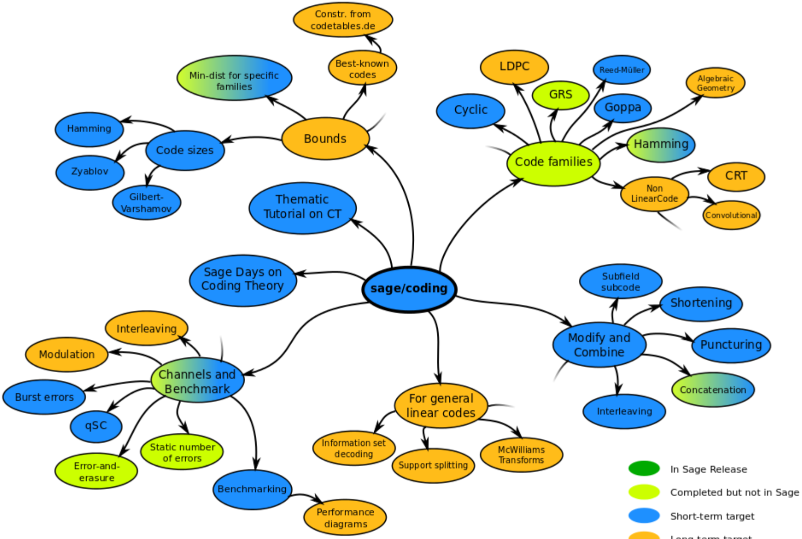
State of CT in Sage

+ 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

Roadmap


Constr. from codetables.de
Best-known codes
Min-dist for specific families
Bounds
Hamming
Code sizes
Zyablov
Gilbert-Varshamov
Thematic Tutorial on CT
LDPC
GRS
Reed-Müller
Algebraic Geometry
Cyclic
Goppa
Hamming
Code families
CRT
Non LinearCode
Convolutional
Sage Days on Coding Theory
sage/coding
Subfield subcode
Shortening
Modify and Combine
Puncturing
Concatenation
Interleaving
Interleaving
Modulation
Channels and Benchmark
Burst errors
qSC
Static number of errors
Error-and-erasure
Benchmarking
Performance diagrams
For general linear codes
Information set decoding
Support splitting
McWilliams Transforms
In Sage Release
Completed but not in Sage
Short-term target
Long-term target

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
  - $\rightarrow$ code families are separate classes
- Multiple points of view supported
  - $\rightarrow$ multiple encoders and decoders
  - $\rightarrow$ but should still be easy
  - $\rightarrow$ 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 links

- https://bitbucket.org/lucasdavid/sage_coding_project/wiki/Home
- https://groups.google.com/forum/#!forum/sage-coding-theory