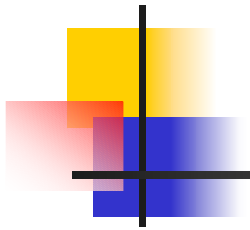


Deletion Decoding Codes from GRS Codes

L McAven, R Safavi-Naini, Y Wang
CIS- UoW
AUSTRALIA

Motivation

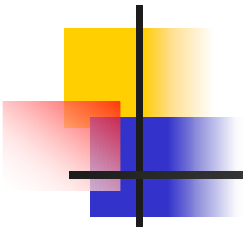
0	1	2	3	4	5	6
0	1	2	3	4	5	6
0	1	2	3	4	5	6
0	1	2	3	4	5	6
0	1	2		4		
0	1	2				

Tracing shortened fingerprints

Deletion Correction

- Codewords, of length *n*
- A received word has *n-r* elements, order preserved.
- Problem: Recover the original word
  - Transmit **x**=(1 6 2 5); receive **y**=(1 6 5).

- Applications: Synchronisation, traitor tracing.
- A code can correct *r* deletions if words of length *n-r* are subwords of *at most* one codeword.



Example

0000	1111	2222	3333	4444	5555	6666
1304	2415	3526	4630	5041	6152	0263
2601	3012	4123	5234	6345	0456	1560
3205	4316	5420	6531	0642	1053	2164
4502	5613	6024	0135	1246	2350	3461
5106	6210	0321	1432	2543	3654	4065
6403	0514	1625	2036	3140	4251	5362

Deletion Correcting Codes

- Constructions
  - Perfect code
    - Combinatorial structures
  - No efficient decoding
- Decoding:
- Brute force:

  For a substring x of length n-r, find codewords that contain x

  Repeat for all x

Generalised Reed-Solomon

- Generalised Reed-Solomon codes are known for their error correcting properties.
- Let Γ be a GRS code GRS*(k,q,n,α,**v**).*
  - A codeword **c** is obtained from a polynomial **fc** over *GF(q)*, of degree $\leq k$,
    - Evaluate polynomial at a subset of points of GF(q)
    - There are *q^(k+1)* codewords.

Theorem: *There exist GRS(k,q,n,α,**v**) codes capable of correcting deletions.*

Decoding shortened words using list decoding

- Efficient list decoding algorithm for GRS codes:
  - Guruswami and Sudan (1999).
- Applicable to deletion decoding.
  - Safavi-Naini and Wang (ACM DRM 2002).
- Let *t=(n-r)* length of the received word. Then for *n≥log q* and *n>k(r+1)+r* the decoding algorithm has running time;

$$O(t^6 \max\{\frac{(k^3 n^6)}{(t^2 - kn)^6}, \frac{1}{k^3}\})$$

Deletion capacity of GRS codes

- Exhaustive searches over small fields ($q<\sim 7$).
- Partial searches over fields (~7<q<~149).
- Correct over half the code length.
- Tabulate length of unique substrings for codelength *n*

		4	5	6	7	8	9	10	11	12	13
k=1, q=13		3	3	4	4	5	5	6	6	7	8
k=2, q=13		4	5	5	6	6	7	7	8	9	9
k=3,q=13		4	5	6	7	7	8	8	9		
k=1,q=31		3	3	3	4	4	4	5	5	6	