Systematic Single-Deletion Multiple-Substitution Correcting Codes

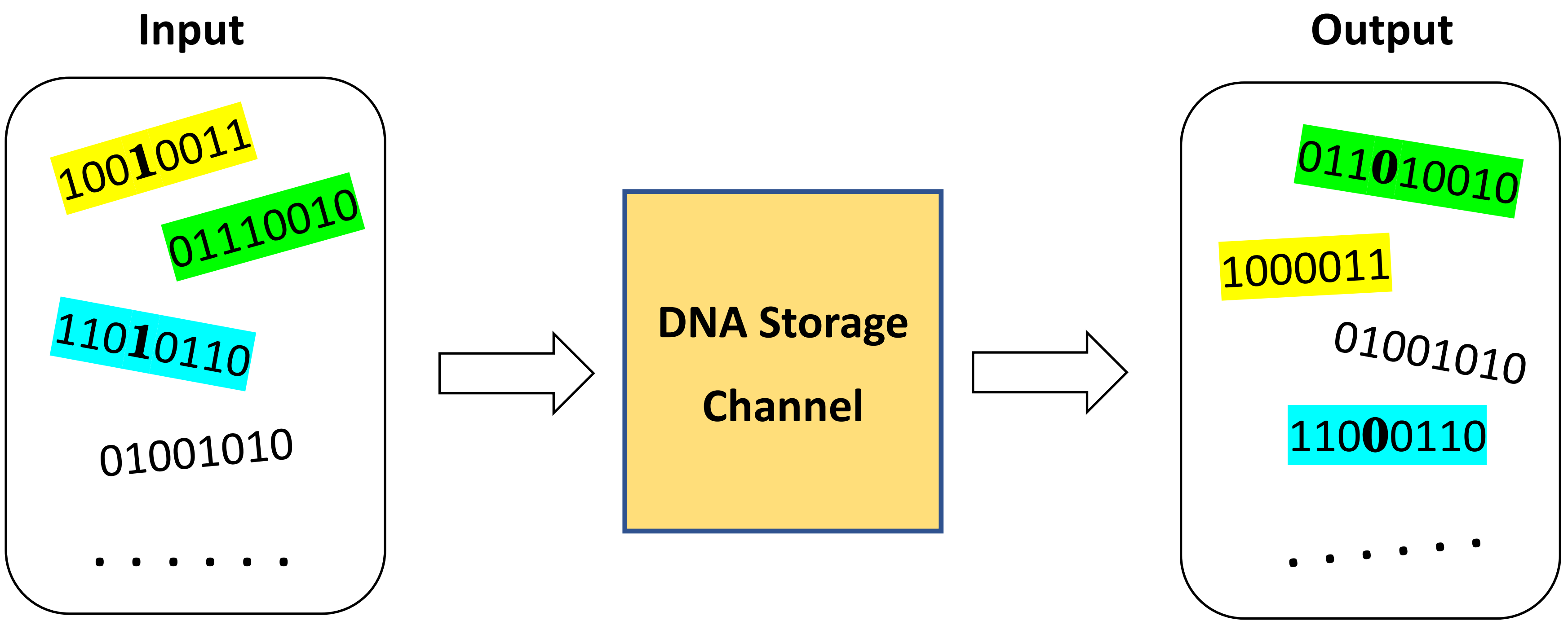
Wentu Song,[1] Nikita Polyanskiiy,[2] Kui Cai,[1] and Xuan He[1]

[1] Science, Mathematics and Technology Cluster, Singapore University of Technology and Design, Singapore

[2] Technical University of Munich, Germany, and Skolkovo Institute of Science and Technology, Russia

12TH ANNUAL NON-VOLATILE MEMORIES WORKSHOP, UCSD, MARCH 2021

1. Introduction



➢ Applications of deletion, insertion and substitution correcting codes : DNA data storage, file synchronization.

1. Introduction

- Redundancy of a code : $\boldsymbol{r(C) = n - \log|C|}$, where $\boldsymbol{n}$ is the length of $\boldsymbol{C}$.

- Systematic code : message $\longrightarrow$ codeword

$$\boldsymbol{x} \longrightarrow (\boldsymbol{x},\ \boldsymbol{p})$$

- Bounds on the optimal redundancy $\boldsymbol{r_{\text{opt}}}$ of $\boldsymbol{t}$-deletion correcting codes [1] :

$$\boldsymbol{t \log n + o(\log n) \le r_{\text{opt}} \le 2t \log n + o(\log n)}.$$

- Varshamov-Tenengolts (VT) codes, a family of asymptotically optimal **single**-deletion correcting codes :

  For each $\boldsymbol{a} \in \boldsymbol{\{0, 1, 2, \dots, n\}}$,

$$\mathbf{VT}_{\boldsymbol{a}}(\boldsymbol{n}) = \left\{ (\boldsymbol{x_1, x_2, \dots, x_n}) \in \{0,1\}^{\boldsymbol{n}} : \boldsymbol{x_1 + 2x_2 + \dots + n\,x_n \equiv a} \ \mathbf{mod}\ (\boldsymbol{n}+1) \right\}.$$

[1] V. I. Levenshtein, “Binary codes capable of correcting deletions, insertions and reversals (in Russian),” Doklady Akademii Nauk SSR, vol. 163, no. 4, pp. 845-848, 1965.

∈

1. Introduction

- ***t***-deletion correcting codes for ***t* > 1 :**

Construction in	**Redundancy**	**Encoding/Decoding Complexity**
Levenshtein [1], [2]	$r \leq 2t \log n$	$O(n^{2t}2^n)$
Brakensiek – Guruswami - Zbarsky [2]	$r \leq O(t^2 \log t \log n)$	$O(n \log^4 n)$
Sima - Bruck[3]	$r \leq 8\, t \log n + o(\log n)$	$O(n^{2t+1})$
Sima – Gabrys – Bruck [4]	$r \leq 4\, t \log n + o(\log n)$	$O(n^{2t+1})$

Remark : The codes constructed in [4] are systematic are capable of correcting r deletions, o insertions, and s substitutions for any r, o, and s satisfying $r + o + s \leq t$.

[2] J. Brakensiek, V. Guruswami, and S. Zbarsky, "Efficient low-redundancy codes for correcting multiple deletions," IEEE Trans. on Inf. Th., vol. 64, no. 5, pp. 3403-3410, 2018.

[3] J. Sima and J. Bruck, "Optimal k-deletion correcting codes," ISIT 2019.

[4] J. Sima, R. Gabrys, and J. Bruck, "Optimal Systematic **t**-Deletion Correcting Codes," ISIT 2020.

1. Introduction

- Codes correcting deletion and insertion errors [1] :

A code C can correct t deletions	⟺	C can correct t insertions	⟺	C can correct the combination of t_1 insertions and t_2 deletions for any non-negative integers t_1 and t_2 such that $t_1 + t_2 \le t$.

- Codes correcting deletion, insertion and substitution errors:

Input		**A substitution**		**Output**
101001	→	101001	→	101101

Input		**A deletion & insertion**		**Output**
101001	→	101~~0~~_01	→	101101

A $(t + o + 2s)$-deletion correcting code	⟹	An $(t + s)$-deletion $(o + s)$-insertion correcting code	⟹	An t-deletion o-insertion s-substitution correcting code

However, it is not necessarily optimal in redundancy.

[5] R. Heckel, G. Mikutis, and R. N. Grass, “A Characterization of the DNA Data Storage Channel,” Scientific Reports, vol. 9, no. 1, pp. 9663, 2019. Available online at: https://doi.org/10.1038/s41598-019-45832-6.

1. Introduction

- **Single**-deletion **s**-substitution correcting codes [6] :
  1) Bounds of optimal redundancy r_{opt} : $(s + 1) \log n + o(\log n) \le r_{opt} \le 2(s + 1) \log n + o(\log n)$;
  2) Construction of **single**-deletion **single**-substitution correcting codes with redundancy **:** $r \le 6 \log n + 8$.

[6] I. Smagloy, L. Welter, A. Wachter-Zeh, and E. Yaakobi, "Single-Deletion Single-Substitution Correcting Codes," ISIT 2020.

- **Our contributions in this work** : Construction of **single**-deletion ***s***-substitution correcting codes ($s \ge 2$) with redundancy $r \le (3s + 4) \log n + o(\log n)$ and encoding/decoding complexity $O(n^{s+3})$.

Construction	Redundancy	Encoding/Decoding Complexity
Sima – Gabrys – Bruck [4]	$r \le (4s + 4) \log n + o(\log n)$	$O(n^{2s+3})$
Our work	$r \le (3s + 4) \log n + o(\log n)$	$O(n^{s+3})$

[7] W. Song, N. Polyanskii, K. Cai, and X. He, "Systematic Single-Deletion Multiple-Substitution Correcting Codes,'' 2020, Available online at: https://arxiv.org/abs/2006.11516

2. Construction Method

- **Notation :**

$B_{1,s}(c)$ —— **the set of all sequences that can be obtained from c by at most one deletion and at most s substitutions (error ball of c).**

- **Confusability Property :** **A function** $f : \{0, 1\}^L \longrightarrow \{0, 1\}^R$ **is said to satisfy the Confusability property if**

$$f(c) \neq f(c')$$

$$\forall\, c \neq c' : B_{1,s}(c) \cap B_{1,s}(c') \neq \phi.$$

- **If f satisfies the Confusability Property, then given any y that is obtained from c by one deletion and at most s substitutions, c can be recovered from y and $f(c)$.**

$$\big(y,\ f(c)\big) \xrightarrow{\text{Decoding}} c$$

2. Construction Method

- **Structure of the encoding function :**

$$x \longrightarrow c = h(x) \longrightarrow \Big(c,\ g(c),\ \mathrm{Rep}_{2s+2}\big(f(g(c))\big)\Big)$$

where

1) $h:\ \{0,1\}^k \longrightarrow \{0,1\}^{n_0}$ **is a systematic encoding function of the binary narrow-sense primitive BCH code (or its shortened code if necessary), hence we have** $n_0 - k \le s(\log n_0 + 2)$.

2) $\mathrm{Rep}_{2s+2}(\cdot)$ **is the encoding function of the $(2s+2)$-fold repetition code.**

3) Both f and g satisfy the Confusability Property, and

$$\text{length of } g(c) = (2s+4)\log n_0 + o(\log n_0);$$

$$\text{length of } \mathrm{Rep}_{2s+2}\big(f(g(c))\big) = \text{length of } f(g(c)) = o(\log n_0).$$

- **Redundancy :** $s(\log n_0 + 2) + (2s+4)\log n_0 + o(\log n_0) + o(\log n_0) = (3s+4)\log n_0 + o(\log n_0)$.

- **Decoding :** **Received sequence** $y \xRightarrow{\text{Splitting}} y^{(1)}\quad y^{(2)}\quad y^{(3)}$

$$x \Longleftarrow c = h(x) \Longleftarrow g(c) \Longleftarrow f(g(c))$$

2. Construction Method

- **Description of f:** $f : \{0, 1\}^{L} \longrightarrow \{0, 1\}^{R}$

  where $R = (s+1)(2s+1)\log L + (2s+1)\log(2s+1)$.

- **Construction of f: Denote** $c = \left(c_1,\ c_2,\ \ldots,\ c_L\right)$**. Then** $f(c) = \left(f(c)_1,\ f(c)_2,\ \ldots,\ f(c)_{2s+1}\right)$ **such that**

$$f(c)_j = \sum_{i=1}^{L}\left(\sum_{e=1}^{i} e^{\,j-1}\right)c_i \mod (2s+1)L^{j}$$

We can prove that f satisfies the Confusability Property, that is

$$f(c) \neq f(c'),$$

$$\forall\, c \text{ and } c' : c' \neq c \text{ and } B_{1,s}(c') \cap B_{1,s}(c) \neq \phi.$$

Let L **= length of** $g(c) = (2s+4)\log n_0 + o(\log n_0)$ **, then we have**

$$\text{length of } \mathrm{Rep}_{2s+2}\left(f\left(g(c)\right)\right) = \text{length of } f\left(g(c)\right) = o(\log n_0)\ .$$

2. Construction Method

Let L = length of $h(x) = n_0$.

- **Syndrome Compression [8] :** There exists a function $P : \{0, 1\}^{n_0} \longrightarrow \left[N\, 2^{o(\log n_0)}\right]$

such that $$f(c) \not\equiv f(c') \mod P(c)\,,$$

$$\forall\, c = h(x) \text{ and } c' = h(x') :\ c' \neq c \text{ and } B_{1,s}(c') \cap B_{1,s}(c) \neq \phi,$$

where

$$\boxed{N = \#\left\{ c' = h(x') :\ c' \neq c \text{ and } B_{1,s}(c') \cap B_{1,s}(c) \neq \phi \right\} \leq n_0^{s+2}.}$$

[8] J. Sima, R. Gabrys, and J. Bruck, "Syndrome Compression for Optimal Redundancy Codes," ISIT, 2020.

2. Construction Method

Let L = length of $h(x) = n_0$.

- **Syndrome Compression [8] :** There exists a function $P : \{0, 1\}^{n_0} \longrightarrow \left[N\, 2^{o(\log n_0)}\right]$

such that

$$f(c) \not\equiv f(c') \mod P(c),$$

$$\forall\, c = h(x) \text{ and } c' = h(x') :\ c' \neq c \text{ and } B_{1,s}(c') \cap B_{1,s}(c) \neq \phi,$$

where

$$\boxed{N = \#\{c' = h(x') :\ c' \neq c \text{ and } B_{1,s}(c') \cap B_{1,s}(c) \neq \phi\} \le n_0^{s+2}.}$$

- **Construction of g :** Let $$g(c) = \big(f(c) \bmod P(c),\ P(c)\big)$$

Then we have

$$\boxed{\text{length of } g(c) = (2s+4)\log n_0 + o(\log n_0).}$$

2. Construction Method

- **Syndrome Compression without precoding : There exists a function**

$$P : \{ 0, 1 \}^{n_0} \longrightarrow \left[N\, 2^{o(\log n_0)} \right]$$

such that

$$f(c) \not\equiv f(c') \mod P(c),$$

$$\forall\, c' : c' \neq c \text{ and } B_{1,s}(c') \cap B_{1,s}(c) \neq \phi,$$

where

$$\boxed{N = \#\left\{ c' : c' \neq c \text{ and } B_{1,s}(c') \cap B_{1,s}(c) \neq \phi \right\} \leq n_0^{2s+2}.}$$

- **Construction of g : Let**

$$g(c) = \big(f(c) \mod P(c),\ P(c) \big)$$

Then we have

$$\boxed{\text{length of } g(c) = 2(2s+2) \log n_0 + o(\log n_0) = (4s+4) \log n_0 + o(\log n_0).}$$

- **Redundancy of the resulted code: $(4s + 4) \log n_0 + o(\log n_0) + o(\log n_0) = (4s + 4) \log n_0 + o(\log n_0)$.**

3. Further Discussions

- **Generalization to construct t-deletion s-substitution correcting codes for t > 1 with redundancy**

$$\boxed{r \leq (4t + 3s) \log n + o(\log n)\ .}$$

Thank you

Q & A