Systematic Single-Deletion Multiple-Substitution Correcting Codes

Wentu Song,¹ Nikita Polyanskiiy,² Kui Cai,¹ and Xuan He¹

¹Science, Mathematics and Technology Cluster, Singapore University of Technology and Design, Singapore
²Technical University of Munich, Germany, and Skolkovo Institute of Science and Technology, Russia

12TH ANNUAL NON-VOLATILE MEMORIES WORKSHOP, UCSD, MARCH 2021
Applications of deletion, insertion and substitution correcting codes: DNA data storage, file synchronization.
### 1. Introduction

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

- **Systematic code:**
  
  \[\begin{align*}
  x & \mapsto (x, p) \\
  \end{align*}\]

- **Bounds on the optimal redundancy** \( r_{opt} \) of \( t \)-deletion correcting codes [1]:
  \[
t \log n + o(\log n) \leq r_{opt} \leq 2t \log n + o(\log n).
  \]

- **Varshamov-Tenengolts (VT) codes,** a family of asymptotically optimal **single**-deletion correcting codes:
  
  For each \( a \in \{0, 1, 2, \ldots, n\} \),
  
  \[ VT_a(n) = \left\{ (x_1, x_2, \ldots, x_n) \in \{0, 1\}^n : x_1 + 2x_2 + \ldots + nx_n \equiv a \mod (n + 1) \right\}. \]

---

1. Introduction

- t-deletion correcting codes for \( t > 1 \):

<table>
<thead>
<tr>
<th>Construction in</th>
<th>Redundancy</th>
<th>Encoding/Decoding Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Levenshtein [1], [2]</td>
<td>( r \leq 2t \log n )</td>
<td>( O(n^{2t}2^n) )</td>
</tr>
<tr>
<td>Brakensiek – Guruswami - Zbarsky [2]</td>
<td>( r \leq O(t^2 \log t \log n) )</td>
<td>( O(n \log^4 n) )</td>
</tr>
<tr>
<td>Sima - Bruck[3]</td>
<td>( r \leq 8t \log n + o(\log n) )</td>
<td>( O(n^{2t+1}) )</td>
</tr>
<tr>
<td>Sima – Gabrys – Bruck [4]</td>
<td>( r \leq 4t \log n + o(\log n) )</td>
<td>( O(n^{2t+1}) )</td>
</tr>
</tbody>
</table>

**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 \( r + o + s \leq t \).


1. Introduction

- Codes correcting deletion and insertion errors [1]:

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

- Codes correcting deletion, insertion and substitution errors:

<table>
<thead>
<tr>
<th>Input</th>
<th>A substitution</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>101001</td>
<td>101001</td>
<td>101101</td>
</tr>
</tbody>
</table>

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

However, it is not necessarily optimal in redundancy.

1. Introduction

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

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

<table>
<thead>
<tr>
<th>Construction</th>
<th>Redundancy</th>
<th>Encoding/Decoding Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sima – Gabrys – Bruck [4]</td>
<td>$r \leq (4s + 4) \log n + o(\log n)$</td>
<td>$O(n^{2s+3})$</td>
</tr>
<tr>
<td><strong>Our work</strong></td>
<td>$r \leq (3s + 4) \log n + o(\log n)$</td>
<td>$O(n^{s+3})$</td>
</tr>
</tbody>
</table>


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 \( f : \{0, 1\}^L \rightarrow \{0, 1\}^R \) is said to satisfy the Confusability property if

\[
f(c) \neq f(c') \\
\forall c \neq c' : B_{1,s}(c) \cap B_{1,s}(c') \neq \emptyset.
\]

➢ 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 \( f(c) \).

\[
\left( y, f(c) \right) \xrightarrow{\text{Decoding}} c
\]
2. Construction Method

➢ Structure of the encoding function:

\[ x \rightarrow c = h(x) \rightarrow (c, g(c), \text{Rep}_{2s+2}(f(g(c)))) \]

where

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

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

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

\[
\begin{align*}
\text{length of } g(c) &= (2s + 4) \log n_0 + o(\log n_0); \\
\text{length of } \text{Rep}_{2s+2}(f(g(c))) &= \text{length of } f(g(c)) = o(\log n_0).
\end{align*}
\]

➢ 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

\[
x \leftrightarrow c = h(x) \leftrightarrow g(c) \leftrightarrow f(g(c))
\]
2. Construction Method

➢ Description of $f$:

\[
f : \{0, 1\}^L \rightarrow \{0, 1\}^R
\]

where

\[
R = (s + 1)(2s + 1) \log L + (2s + 1) \log (2s + 1).
\]

➢ Construction of $f$: Denote $c = (c_1, c_2, \ldots, c_L)$. Then $f(c) = (f(c)_1, f(c)_2, \ldots, f(c)_{2s+1})$ 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

\[
f(c) \neq f(c'),
\]

\[
\forall c \text{ and } c' : c' \neq c \text{ and } B_{1, s}(c') \cap B_{1, s}(c) \neq \emptyset.
\]

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

\[
\text{length of Rep}_{2s+2}(f(g(c))) = \text{length of } f(g(c)) = o(\log n_0).
\]
2. Construction Method

Let \( L = \text{length of } h(x) = n_0 \).

- Syndrome Compression [8]: There exists a function \( P : \{0, 1\}^{n_0} \rightarrow [N 2^{O(\log n_0)}] \) such that
  \[ f(c) \not\equiv f(c') \mod P(c), \]
  \[ \forall c = h(x) \text{ and } c' = h(x'): c' \neq c \text{ and } B_{1,s}(c') \cap B_{1,s}(c) \neq \emptyset, \]

where

\[ N = \# \{ c' = h(x') : c' \neq c \text{ and } B_{1,s}(c') \cap B_{1,s}(c) \neq \emptyset \} \leq n_0^{s+2}. \]

2. Construction Method

Let \( L = \text{length of } h(x) = n_0. \)

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

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

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

where

\[ N = \# \{ c' = h(x') : c' \neq c \text{ and } B_{1,s}(c') \cap B_{1,s}(c) \neq \emptyset \} \leq n_0^{s+2}. \]

Construction of \( g : \) Let \( g(c) = (f(c) \mod P(c), \ P(c)) \)

Then we have

\[ \text{length of } g(c) = (2s + 4) \log n_0 + o(\log n_0). \]
2. Construction Method

➢ Syndrome Compression without precoding: There exists a function

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

such that

\[ f(c) \not\equiv f(c') \mod P(c), \]

\[ \forall c': c' \neq c \text{ and } B_{1,s}(c') \cap B_{1,s}(c) \neq \emptyset, \]

where

\[ N = \# \{ c' : c' \neq c \text{ and } B_{1,s}(c') \cap B_{1,s}(c) \neq \emptyset \} \leq n_0^{2s+2}. \]

➢ Construction of \( g \): Let \( g(c) = (f(c) \mod P(c), P(c)) \)

Then we have

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

➢ 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

$$r \leq (4t + 3s) \log n + o(\log n).$$
Thank you

Q & A