Codes for Correcting Tandem Repeats

Yeow Meng Chee, Johan Chrisnata, Han Mao Kiah, and Tuan Thanh Nguyen
School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore
email: {ymchee, jchrisnata, hmkiah, nguyentu001}@ntu.edu.sg

Abstract—Tandem duplication in DNA is the process of inserting a copy of a segment of DNA adjacent to the original position. Motivated by applications that store data in living organisms, Jain et al. (2016) proposed the study of codes that correct tandem duplications to improve the reliability of data storage. We investigate algorithms associated with the study of these codes.

Two words are said to be $\leq k$-confusable if there exists two sequences of tandem duplications of lengths at most $k$ such that the resulting words are equal. We demonstrate that the problem of deciding whether two words is $\leq k$-confusable is linear-time solvable through a characterisation that can be checked efficiently for $k = 3$.

Using insights gained from the algorithm, we study the size of tandem-duplication codes. We improve the previous known upper bound and then construct codes with larger sizes as compared to the previous constructions. We determine the sizes of optimal tandem-duplication codes for lengths up to twenty, and develop recursive methods to construct tandem-duplication codes for allword lengths.

I. INTRODUCTION

Lander et al. [1] published a draft sequence of the human genome and reported that more than 50% of the genome consists of repeated substrings. There are two types of common repeats: interspersed and tandem repeats. Interspersed repeats are caused by transposons when a segment of DNA is copied and pasted into new positions of the genome. In contrast, tandem repeats are caused by slipped-strand mispairings [2], and they occur when a pattern of one or more nucleotides is repeated and the repetitions are adjacent to each other. For example, consider the word AGTAGTCTGC. The substring AGTAGT is a tandem repeat, and we say that AGTAGTCTGC is generated from AGTCTGC by a tandem duplication of length three. Tandem repeats are believed to be the cause of several genetic disorders [3].

Recently, motivated by applications that store data in living organisms [4]–[6], Jain et al. [7] proposed the study of codes that correct tandem duplications to improve the reliability of data storage. They investigated various types of tandem duplications and provided optimal code construction in the case where duplication length is at most two.

In this work, we investigate algorithms associated with these codes. In particular, given two words $x$ and $y$, we look for efficient algorithms that answer the following question: When are the words $x$ and $y$ confusable under tandem repeats? In the full version of this paper [8], we derive sufficient and necessary conditions for two strings to be confusable and propose a linear-time algorithm to determine the confusability of any two strings assuming that the length of tandem repeats is at most three.

Due to space constraints, we omit the description of the algorithm here. Instead, we summarise the results on the code sizes that were obtained via insights from the algorithm.

II. NOTATIONS AND TERMINOLOGY

Let $\Sigma_q = \{0, 1, \ldots, q - 1\}$. Let $\Sigma_q^n$ denote the set of all sequences of length $n$ over $\Sigma_q$, and let $\Sigma_q^*$ denote the set of all finite sequences over $\Sigma_q$. Given two words $x, y \in \Sigma_q^*$, we denote their concatenation by $xy$.

We state the tandem duplication rules. For nonnegative integers $k \leq n$ and $i \leq n - k$, we define $T_{i,k} : \Sigma_q^n \rightarrow \Sigma_q^{n+k}$ such that $T_{i,k}(x) = uwvw$, where $x = uwv$, $|u| = i$, $|v| = k$.

If a finite sequence of tandem duplications of length at most $k$ is performed to obtain $y$ from $x$, then we say that $y$ is a $\leq k$-descendant of $x$, or $x$ is a $\leq k$-ancestor of $y$, and denote this relation by $x \rightarrow_{\leq k} y$. We define the $\leq k$-descendant cone of $x$ to be the set of all $\leq k$-descendants of $x$ and denote this cone by $D_{\leq k}(x)$.

Motivated by applications that store data on living organisms, Jain et al. [7] looked at the $\leq k$-descendant cones of a pair of words and asked whether the two cones have a nonempty intersection. Specifically, we introduce the notion of confusability.

Definition 1 (Confusability). Two words $x$ and $y$, are said to be $\leq k$-confusable if $D_{\leq k}^*(x) \cap D_{\leq k}^*(y) \neq \emptyset$.

To design error-correcting codes that store information in the DNA of living organisms, Jain et al. then proposed the use of codewords that are not pairwise confusable.

Definition 2 ($\leq k$-Tandem-Duplication Codes). A subset $C \subseteq \Sigma_q^n$ is a $\leq k$-tandem-duplication code if for all $x, y \in C$ and $x \neq y$, we have that $x$ and $y$ are not $\leq k$-confusable. We say that $C$ is an $(n, \leq k; q)$-tandem-duplication code or $(n, \leq k; q)-$TD code.

We are interested in determining the maximum possible size of an $(n, \leq k; q)$-TD code, and we denote the quantity by $T(n, k; q)$.

A. Previous Work

We state known lower and upper bounds on the quantity $T(n, k; q)$. Crucial to these bounds is the concept of irreducible words and roots.

Definition 3. A word $x$ is said to be $\leq k$-irreducible if $x$ cannot be deduplicated into shorter words with deduplication of length at most $k$. In other words, if $y \rightarrow_{\leq k} x$, then $y = x$. The set of $\leq k$-irreducible $q$-ary words is denoted by $\text{Irr}_{\leq k}(q)$ and those of length $n$ is denoted by $\text{Irr}_{\leq k}(n, q)$. The $\leq k$-ancestors of $x \in \Sigma_q^*$ that are $\leq k$-descendants are called the $\leq k$-roots of $x$, denoted by $R_{\leq k}(x)$.

Jain et al. used irreducible words to construct tandem-duplication codes and demonstrate that the construction is optimal for the case $k = 2$.

Construction 1.

(i) $T(n, 2; 3) = \sum_{i=1}^{n} |\text{Irr}_{\leq 2}(i, q)|$.
(ii) $T(n, 3; 3) \geq \sum_{i=1}^{n} |\text{Irr}_{\leq 3}(i, q)|$.

We next look at upper bounds on the size of an optimal $(n, \leq 3; q)$-TD code. By definition, an $(n, \leq 3; q)$-TD code is also an $(n, \leq 2; q)$-TD code. Since an optimal $(n, \leq 2; q)$-TD code is provided by Construction 1, we have the following upper bound on the size of an optimal $(n, \leq 3; q)$-TD code.

Proposition 1. $T(n, 3; q) \leq \sum_{i=1}^{n} |\text{Irr}_{\leq 2}(i, q)|$. 
Proposition 1 implies that Construction 1 is tight for $k = 3$ and $n \leq 5$. Using a combinatorial characterization implied by our algorithm, we improve this upper bound for longer lengths.

### III. Tandem Duplication Codes

Motivated by the concept of roots, we consider a $\leq 3$-irreducible word $r$ and we say that a $(n, \leq 3; q)$-TD code $C$ is an $(n, \leq 3; r)$-TD code if all words in $C$ belong to $D_{\leq 3}(r)$. Since $\bigcup_{r \in \text{Irr}_{\leq 3}(q)} C(n, r)$ is an $(n, \leq 3; q)$-TD code, we provide estimates on the size of an optimal $(n, \leq 3; r)$-TD code for a fixed $r$. To simplify our discussion, we focus on the case $q = 3$ and let $T(n)$ and $T(n, r)$ to denote the sizes of an optimal $(n, \leq 3; 3)$-TD code and an optimal $(n, \leq 3; r)$-TD code, respectively.

Via our combinatorial characterization, we determined the exact values of $T(n, r)$ in certain cases.

**Proposition 2 (Exact Values).**

(i) $T(n, r) = 1$ for $|r| \leq n \leq |r| + 2$.

(ii) $T(|r| + 3, r) = 2$.

(iii) Let $r \in R ≜ \{012, 0120, 01201, 0112, 01120, 011201, 0112012, 01120120, 011201201, 0112012010, 01120120101, 011201201012, 01120120101201, 011201201012010\}$. Set

$$n_2(r) ≜ 10, \quad \text{if} \ r = 012, \quad 11, \quad \text{if} \ r \in \{0121, 0112\}, \quad 12, \quad \text{if} \ r \in \{01201, 01021\}, \quad 13, \quad \text{if} \ r \in \{01202, 01201\}, \quad 14, \quad \text{if} \ r \in \{01201, 01021\}, \quad 15, \quad \text{if} \ r \in \{012012, 010210\}, \quad 16, \quad \text{if} \ r = 01201010.$$

Then we have

$$T(r, n) = \begin{cases} \frac{n-n_2(r)}{6} + 3 & \text{if} \ n \geq n_2(r), \\ 2, & \text{if} \ |r| + 3 \leq n < n_2(r). \end{cases}$$

Our combinatorial characterisation also improves the upper bound on $T(n)$.

**Theorem 3 (Upper Bound).** Let $R$ be as defined in Proposition 2. $I(i, m)$ denote the number of irreducible words in $\text{Irr}_{\leq 3}(i, 3)$ with exactly $m$ regions, and $U(n, i, m)$ be defined as follows:

$$U(n, i, m) ≜ \begin{cases} (n-i)/3+m \quad & \text{if} \ 3 | (n-i), \\ (n-i)/3+m-1 \quad & \text{if} \ 3 \nmid (n-i), \ m | (n-i)/3, \\ \frac{i-1}{3} \quad & \text{if} \ m \nmid (n-i)/3. \end{cases}$$

Then

$$T(n) \leq \sum_{r \in R} T(n, r) + \sum_{i=5}^{n} I(i, m)U(n, i, m). \quad (1)$$

In addition to the above constructions, we consider tandem-duplication codes for small lengths by searching for them exhaustively. We do so by using our confusability algorithm to construct a graph $\mathcal{G}(n, r)$, whose vertices correspond to the set of all descendants of $r$ of length $n$. Then $T(n, r)$ is given by the maximum size of a clique in $\mathcal{G}(n, r)$ and we use the exact algorithm MaxCliqueDyn [9] to compute $T(n, r)$ for $n \leq 6$.

We tabulate the results $T(n)$ in Table I.

Finally, we develop several recursive constructions.

**Proposition 4.** Let $r = r_1r_2 \cdots r_i \in \text{Irr}_{\leq 3}(i, 3)$. Then the following holds.

$$T(n-1, r \setminus r_1), \quad \text{if} \ r_1 = r_3, \quad \max\{T(n-4, r \setminus r_1), T(n-8, r \setminus r_1)\}, \quad \text{if} \ r_1 \neq r_3, \ r_1 \neq r_4,$$

$$T(n, r) \geq \begin{cases} \\
\max\{T(n-5, r \setminus r_1r_2), T(n-10, r \setminus r_1r_2)\}, \quad & \text{if} \ r_1 \neq r_3, \ r_1 \neq r_4, r_2 \neq r_3, \\
\max\{T(n-6, r \setminus r_1r_2r_3), T(n-12, r \setminus r_1r_2r_3)\}, & \text{if} \ r_1 \neq r_3, \ r_1 \neq r_4, r_2 = r_3. 
\end{cases}$$

Furthermore, $T(n, r) \geq T(n-1, r)$ and $T(n, r) = T(n, r^{\text{rev}})$, where $z^{\text{rev}}$ denotes the reverse of word $z$.

Using Proposition 4 with Proposition 2 and the values computed by MaxCliqueDyn, we derive lower bounds for $T(n)$ for $21 \leq n \leq 30$. The results are summarized in Table I. In addition to the lower bounds for the code size $T(n)$, we also compare the upper bounds in Proposition 1 and (1). Observe that (1) is tight up to lengths at most ten and the constructions in this paper improve the rates 2 for Construction 1 by as much as 6.74%.

**REFERENCES**


1\footnote{The number of vertices can be dramatically reduced using a finer analysis. We refer the reader to [8] for more details.}

2\footnote{The rate of a code $C$ of length $n$ is given by $\log_2 |C| / n$.}