Title	Deterministic Polynomial-Time Algorithms for Designing Short DNA Words
Author(s)	Kao, MY; Leung, HCM; Sun, H; Zhang, Y
Citation	Theoretical Computer Science, 2013, v. 494, p. 144-160
Issued Date	2013
URL	http://hdl.handle.net/10722/185124
Rights	Creative Commons: Attribution 3.0 Hong Kong License

Deterministic Polynomial-Time Algorithms for Designing Short DNA Words

Ming-Yang Kao* Henry C. M. Leung† He Sun‡ Yong Zhang§

(first draft July 29, 2010; last revised January 30, 2012)

Abstract

Designing short DNA words is a problem of constructing a set (i.e., code) of n DNA strings (i.e., words) with the minimum length such that the Hamming distance between each pair of words is at least k and the n words satisfy a set of additional constraints. This problem has applications in, e.g., DNA self-assembly and DNA arrays. Previous works include those that extended results from coding theory to obtain bounds on code and word sizes for biologically motivated constraints and those that applied heuristic local searches, genetic algorithms, and randomized algorithms. In particular, Kao, Sanghi, and Schweller [16] developed polynomial-time randomized algorithms to construct n DNA words of length within a multiplicative constant of the smallest possible word length (e.g., $9 \cdot \max\{\log n, k\}$) that satisfy various sets of constraints with high probability. In this paper, we give deterministic polynomial-time algorithms to construct DNA words based on derandomization techniques. Our algorithms can construct n DNA words of shorter length (e.g., $2.1 \log n + 6.28k$) and satisfy the same sets of constraints as the words constructed by the algorithms of Kao et al. Furthermore, we extend these new algorithms to construct words that satisfy a larger set of constraints for which the algorithms of Kao et al. do not work.

Keywords: DNA word design, deterministic algorithms, derandomization.

1 Introduction

Building on the work of Kao, Sanghi, and Schweller [16], this paper considers the problem of designing sets (codes) of DNA strings (words) satisfying certain combinatorial constraints with the length as short as possible. Many applications depend on the scalable design of such words. For instance, DNA words can be used to store information at the molecular level [6], to act as molecular bar codes for identifying molecules in complex libraries [6, 7, 20], or to implement DNA arrays [3]. For DNA computing, inputs to computational problems are encoded into DNA strands to perform computation via complementary binding [1, 25]. For DNA self-assembly, Wang tile self-assembly systems are implemented by encoding glues of Wang tiles into DNA strands [2, 24–26].

*Department of Electrical Engineering and Computer Science, Northwestern University, USA. Email: kao@northwestern.edu. Supported in part by NSF Grant CCF-1049899.

†Department of Computer Science, The University of Hong Kong, Hong Kong. Email: cmleung2@cs.hku.hk.

‡Max Planck Institute for Informatics, Germany. Institute of Modern Mathematics and Physics, Fudan University, China. Email: hsun@mpi-inf.mpg.de.

§Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, China. Department of Computer Science, The University of Hong Kong, Hong Kong. Email: yzhang@cs.hku.hk. Supported in part by NSFC Grant 11171086.

A set of DNA words chosen for such applications typically need to meet certain combinatorial constraints. For instance, hybridization should not occur between distinct words in the set, or even between a word and the reverse of another word in the set. For such requirements, Marathe et al. [18] proposed the *basic Hamming constraint* (C_1), the *reverse complement Hamming constraint* (C_2), and the *self-complementary constraint* (C_3). In addition to C_1, C_2, and C_3, Kao et al. [16] further considered certain more restricting *shifting* versions (C_4, C_5, C_6) of these constraints which require C_1, C_2, and C_3 to hold between alignments of pairs of words [5].

Kao et al. [16] also considered three constraints unrelated to Hamming distance. The *GC content constraint* (C_7) requires that a specified fraction of the bases in a word are G or C. This constraint gives the words similar thermodynamic properties [21–23]. The *consecutive base constraint* (C_8) limits the length of any run of identical bases in a word. Long runs of identical bases can cause hybridization errors [4, 5, 21]. The *free energy constraint* (C_9) requires that the difference in the free energies of two words is bounded by a small constant. This constraint helps ensure that the words in the set have similar melting temperatures [5, 18].

Furthermore, it is desirable for the length ℓ of the words to be as small as possible. The motivation for minimizing ℓ is in part because it is more difficult to synthesize longer DNA strands. Also, longer DNA strands require more DNAs to be used for the respective application.

There have been a considerable number of previous works in the design of DNA words [5, 6, 9–13, 15, 17–20, 22, 23]. Most of the existing works are based on heuristics, genetic algorithms, or stochastic local searches and do not provide analytical performance guarantees. Notable exceptions include the work of Marathe et al. [18] that extends results from coding theory to obtain bounds on code size for biologically motivated constraints. Also, Kao et al. [16] formulated an optimization problem that takes as input a desired cardinality n and produces n words of length ℓ that satisfy a specified set of constraints, while minimizing the length ℓ. Kao et al. introduced randomized algorithms that run in polynomial time to construct words whose length ℓ is within a constant multiplicative factor of the optimal word length. However, with a non-negligible probability, the constructed words do not satisfy the given constraints. The results of Kao et al. are summarized in Table 1 for comparison with ours.

This paper presents deterministic polynomial-time algorithms for constructing n desired words of length within a constant multiplicative factor of the optimal word length. As shown in Table 1, our algorithms can construct words shorter than those constructed by the randomized algorithms of Kao et al. [16]. Also, our algorithms can construct desired words that satisfy more constraints than the work of Kao et al. has done. Our algorithms derandomize a randomized algorithm of Kao et al. Depending on the values of k and n, different parameters of derandomization can be applied to minimize the length ℓ of words. Our results are summarized in Table 1.

An Erratum The conference version of this work [14] has claimed a set of results based on expander codes. As we announced at our conference presentation of this work, those results are false. Those results have been removed from this full version.

Organization of the Remainder of This Paper Section 2 gives some basic notations and the nine constraints C_1 through C_9 for DNA words. Section 3 discusses how to design a set of short DNA words satisfying the constraints C_1 and C_4. Section 4 discusses how to construct short DNA words under additional sets of constraints. Section 5 concludes the paper with some directions for further research.

Codes	Randomized Algorithms (Kao et al. [16])	Deterministic Algorithms (this paper)
$\mathcal{W}_{1,4}$	see $\mathcal{W}_{1\sim 6}$	$\ell^\star = \lceil c_1 \log n + c_2 k \rceil$
$\mathcal{W}_{1\sim 6}$	$\ell = 9\max\{\log n, k\}$	$\ell = \ell^\star + k$
$\mathcal{W}_{1\sim 7}$	$\ell = 10\max\{\log n, k\}$	$\ell = \ell^\star + 2k$
$\mathcal{W}_{1\sim 3,7,8}$	$\ell = \frac{d}{d-1}10\max\{\log n, k\}$	$\ell = \frac{d}{d-1}(\ell^\star + 2k) + O(1)$
$\mathcal{W}_{1\sim 8}$	no result	$\ell = \ell^\star + 2k$ when $\frac{1}{d+1} \le \gamma \le \frac{d}{d+1}$ $\ell = \frac{d}{d-1}\ell^\star + \frac{d}{d-2}2k + O(d)$ when $d \ge 3$
$\mathcal{W}_{1\sim 6,9}$	$\ell = 27\max\{\log n, k\}$ when $\sigma \ge 4D + \Gamma_{\max}$	$\ell = 3\ell^\star + 2k$ when $\sigma \ge 4D + \Gamma_{\max}$

Table 1: Comparison of word lengths. The constraints C_1 through C_9 are defined in Section 2. $\mathcal{W}_{1,4}$ is a code of n words that satisfies C_1 and C_4. Code $\mathcal{W}_{1\sim 6}$ satisfies C_1 through C_6. Codes $\mathcal{W}_{1\sim 7}$, $\mathcal{W}_{1\sim 3,7,8}$, $\mathcal{W}_{1\sim 8}$, and $\mathcal{W}_{1\sim 6,9}$ are similarly defined. The output parameters ℓ and $\ell^\star$ are the lengths of the constructed words. The constraint parameter k is the maximum of the dissimilarity parameters for the associated subset of C_1 through C_6; the constraint parameter d is the run-length parameter for C_8; the constraint parameter σ, D and $\Gamma_{\max}$ are free-energy parameters for C_9, where D and $\Gamma_{\max}$ are defined in Section 4.5. The design parameters c_1 and c_2 can be used to control the lengths of the constructed words, where c_1 is any real number greater than 2, and $c_2 = \frac{c_1}{2}\left\{\log\left(\frac{c_1}{(c_1-2)\ln 2}\right) + 2.5 - \frac{1}{\ln 2}\right\}$. As examples, for $c_1 = 2.1$, $\ell^\star = \lceil 2.1\log n + 6.28k \rceil$, and for $c_1 = 3$, $\ell^\star = \lceil 3\log n + 4.76k \rceil$. For simplicity, we omit the ceiling notation from the right-hand sides of expressions for ℓ in the table. The results of this work summarized in this table are corollaries of Theorems 9, 13, 15, 17, 19, 21, and 23. The lengths $\ell^\star$ and k used in these theorems are typically slightly smaller than those used in this table.

Technical Remarks Throughout this paper, all logarithms log have base 2 unless explicitly specified otherwise.

2 Preliminaries

This paper considers words on two alphabets, namely, the binary alphabet $\Pi_B = \{0, 1\}$ and the DNA alphabet $\Pi_D = \{\mathrm{A}, \mathrm{C}, \mathrm{G}, \mathrm{T}\}$.

Let $X = x_1 \cdots x_\ell$ be a word where x_i belongs to an alphabet Π. The *reverse* of X, denoted by X^R, is the word $x_\ell x_{\ell-1} \cdots x_1$. The *complement* of X, denoted by X^c, is $x_1^c \cdots x_\ell^c$, where if Π is the binary alphabet $\Pi_B = \{0, 1\}$, then $0^c = 1$ and $1^c = 0$, and if Π is the DNA alphabet $\Pi_D = \{\mathrm{A}, \mathrm{C}, \mathrm{G}, \mathrm{T}\}$, then $\mathrm{A}^c = \mathrm{T}, \mathrm{C}^c = \mathrm{G}, \mathrm{G}^c = \mathrm{C}$, and $\mathrm{T}^c = \mathrm{A}$. For integer i and j with $1 \le i \le j \le \ell$, $X[i \cdots j]$ denotes the substring $x_i \cdots x_j$ of X. The *Hamming distance* between two words X and Y of equal length, denoted by $H(X, Y)$, is the number of positions where X and Y differ.

Next we review the nine constraints C_1 through C_9 as defined in [16]. Let $\mathcal{W}$ be a set of words of equal length ℓ. The constraints are defined for $\mathcal{W}$. For naming consistency, we rename the Self-Complementary Constraint of [16] to the Self Reverse Complementary Constraint in this paper; similarly, we rename the Shifting Self-Complementary Constraint of [16] to the Shifting Self Reverse Complementary Constraint in this paper.

1. **Basic Hamming Constraint** $C_1(k_1)$: Given an integer k_1 with $\ell \ge k_1 \ge 0$, for any distinct

words $Y, X \in \mathcal{W}$,

$$H(Y, X) \quad \geq \quad k_1. \tag{1}$$

This constraint limits non-specific hybridization between a word Y and the Watson-Crick complement of a distinct word X (and by symmetry between the Watson-Crick complement of a word Y with a distinct word X).

2. **Reverse Complementary Constraint** $C_2(k_2)$: Given an integer k_2 with $\ell \geq k_2 \geq 0$, for any distinct words $Y, X \in \mathcal{W}$,

$$H(Y, X^{RC}) \geq k_2.$$

This constraint limits hybridization between a word Y and the reverse of a distinct word X.

3. **Self Reverse Complementary Constraint** $C_3(k_3)$: Given an integer k_3 with $\ell \geq k_3 \geq 0$, for any word $Y \in \mathcal{W}$,

$$H(Y, Y^{RC}) \geq k_3.$$

This constraint prevents a word Y from hybridizing with the reverse of itself.

4. **Shifting Hamming Constraint** $C_4(k_4)$: Given an integer k_4 with $\ell \geq k_4 \geq 0$, for any distinct words $Y, X \in \mathcal{W}$,

$$H(Y[1 \cdots i], X[(\ell - i + 1) \cdots \ell]) \geq k_4 - (\ell - i) \text{ for all } \ell \geq i \geq \ell - k_4. \tag{2}$$

This constraint is a stronger version of the constraint C_1 applied to every pair of a prefix of Y and a suffix of X of equal length i with $\ell \geq i \geq \ell - k_4$ and a length-adjusted lower bound $k_4 - (\ell - i)$ for the Hamming distance.

5. **Shifting Reverse Complementary Constraint** $C_5(k_5)$: Given an integer k_5 with $\ell \geq k_5 \geq 0$, for any distinct words $Y, X \in \mathcal{W}$,

$$\begin{aligned} H(Y[1 \cdots i], X[1 \cdots i]^{RC}) \quad &\geq \quad k_5 - (\ell - i); \text{ and} \\ H(Y[(\ell - i + 1) \cdots \ell], X[(\ell - i + 1) \cdots \ell]^{RC}) \quad &\geq \quad k_5 - (\ell - i) \text{ for all } \ell \geq i \geq \ell - k_5. \end{aligned}$$

This constraint is a stronger version of the constraint C_2 applied to every pair of a prefix of Y and a prefix of X of equal length i and also every pair of a suffix of Y and a suffix of X of equal length i with $\ell \geq i \geq \ell - k_5$ and a length-adjusted lower bound $k_5 - (\ell - i)$ for the Hamming distance.

6. **Shifting Self Reverse Complementary Constraint** $C_6(k_6)$: Given an integer k_6 with $\ell \geq k_6 \geq 0$, for any word $Y \in \mathcal{W}$,

$$\begin{aligned} H(Y[1 \cdots i], Y[1 \cdots i]^{RC}) \quad &\geq \quad k_6 - (\ell - i); \text{ and} \\ H(Y[(\ell - i + 1) \cdots \ell], Y[(\ell - i + 1) \cdots \ell]^{RC}) \quad &\geq \quad k_6 - (\ell - i) \text{ for all } \ell \geq i \geq \ell - k_6. \end{aligned}$$

This constraint is a stronger version of the constraint C_3 applied to every prefix of Y and every suffix of Y of length i with $\ell \geq i \geq \ell - k_6$ and a length-adjusted lower bound $k_6 - (\ell - i)$ for the Hamming distance.

7. **GC Content Constraint** $C_7(\gamma)$: Given a real number γ with $1 \geq \gamma \geq 0$, γ fraction of the characters (e.g., $\lceil \gamma\ell \rceil$ characters, $\lfloor \gamma\ell \rfloor$ characters, or $\gamma\ell + O(1)$ characters) in each word $Y \in \mathcal{W}$ are G or C.

   The GC content affects thermodynamic properties of a word [21, 23]. Therefore, having the same ratio of GC content for all the words helps ensure similar thermodynamic characteristics.

8. **Consecutive Base Constraint** $C_8(d)$: Given an integer $d \geq 2$, no word in $\mathcal{W}$ has more than d consecutive bases.

   In some applications, consecutive occurrences (also known as runs) of the same base increase annealing errors.

   Note that if $d = 1$ and $\mathcal{W}$ is a set of binary words, then $\mathcal{W}$ consists of at most two words, of which one word starts with 0 and alternates between 0 and 1, and the other word is the complement of the former word. The requirement that $d \geq 2$ rules out this trivial case.

9. **Free Energy Constraint** $C_9(\sigma)$: Given a real number $\sigma \geq 0$, for any two distinct words $Y, X \in \mathcal{W}$,

$$|\mathrm{FE}(Y) - \mathrm{FE}(X)| \leq \sigma,$$

   where $\mathrm{FE}(Z)$ denotes the free energy of a word Z. See Section 4.5 for the definition of a particular free energy function FE considered in [16] and this paper.

   This constraint helps ensure that the words in the set $\mathcal{W}$ have similar melting temperatures, which allows multiple DNA strands to hybridize simultaneously at a temperature [20].

The lemma below summarizes some simple properties of constraints $C_1(k_1)$ through $C_6(k_6)$ and $C_8(d)$.

Lemma 1 (see, e.g., [16])**.**

1. *If $C_4(k)$ holds, then $C_1(k)$ also holds.*
2. *For each C_p of the first six constraints, if $k \geq k_p$ and $C_p(k)$ holds, then $C_p(k_p)$ also holds.*
3. *For two integers $d \geq d' \geq 2$, if $C_8(d')$ holds, then $C_8(d)$ also holds.*
4. *For each C_p of the first six constraints, if $\mathcal{W}$ is set of n distinct binary words (respectively, DNA words) of equal length ℓ and satisfies $C_p(k_p)$, then $\ell \geq \max\{\log n, k_p\}$ (respectively, $\ell \geq \max\{\log_4 n, k_p\}$).*

Proof. Statement 1 follows from the fact that $C_1(k)$ is the same as the case $i = \ell$ in Inequality (2) for $C_4(k)$. Statements 2 through 4 are also straightforward. □

Technical Remarks In this work, we interpret the terms $X[1 \cdots i]^{RC}$, $X[(\ell - i + 1) \cdots \ell]^{RC}$, $Y[1 \cdots i]^{RC}$, and $Y[(\ell - i + 1) \cdots \ell]^{RC}$ in the definitions of $C_5(k_5)$ and $C_6(k_6)$ as $(X[1 \cdots i])^{RC}$, $(X[(\ell-i+1) \cdots \ell])^{RC}$, $(Y[1 \cdots i])^{RC}$, and $(Y[(\ell-i+1) \cdots \ell])^{RC}$, respectively. However, it would also be reasonable to interpret these terms in a subtly different manner as $(X^{RC})[1 \cdots i]$, $(X^{RC})[(\ell - i + 1) \cdots \ell]$, $(Y^{RC})[1 \cdots i]$, and $(Y^{RC})[(\ell - i + 1) \cdots \ell]$.

3 Designing Words for Constraints $C_1(k_1)$ and $C_4(k_4)$

In this section, we give a deterministic polynomial-time algorithm, namely, DetWords (Algorithm 1), which can be used to construct a code $\mathcal{W}_{1,4}$ of n DNA words of length $\ell^\star = \lceil c_1 \log n + c_2 k \rceil$ for a range of positive constants c_1 and c_2 to satisfy constraints $C_1(k_1)$ and $C_4(k_4)$, where $k = \max\{k_1, k_4\}$.

Algorithm DetWords takes n, ℓ, k_1, and k_4 as input and then outputs an $n \times \ell$ binary matrix. We can view the rows of this binary matrix as a code of n binary words of length ℓ. In turn, we can convert these binary words into DNA words by replacing 0 and 1 with two distinct DNA characters. The remainder of this section will focus on constructing binary words. Also, for convenience, we will refer to binary row vectors, binary words, and DNA words interchangeably when there is no risk of ambiguity.

We design Algorithm DetWords by derandomizing a randomized algorithm in [16]. The basic idea for Algorithm DetWords is to implicitly generate a random $n \times \ell$ binary matrix M by assigning 0 or 1 with equal probability $1/2$ to each of the $n\ell$ positions in M independently. We then derandomize the assignment at each position to choose 0 or 1 one position at a time based on conditional expectations of the number of pairs of distinct rows and their shifted prefixes and suffixes that satisfy $C_1(k_1)$ and $C_4(k_4)$.

More specifically, Algorithm DetWords works as follows. It first creates an empty $n \times \ell$ binary matrix. It then fills the empty entries one at a time with 0 or 1. Before the algorithm chooses 0 or 1 to fill an empty entry, it computes two expectations. The first expectation is the term E_0 at Line 7 in Algorithm 1. Informally, this expectation is the expected number of times Inequalities (1) and (2) are satisfied if the current empty entry is filled with 0. The second expectation is the term E_1 at Line 8 in Algorithm 1. Informally, this expectation is the expected number of times Inequalities (1) and (2) are satisfied if the current empty entry is filled with 1. These expectations are formally defined in Equation (3) below. According to the manner in which Equation (3) counts how many times Inequalities (1) and (2) are satisfied, a set of n words of length ℓ can satisfy or fail these inequalities exactly $\binom{n}{2} \cdot (1 + 2(k_4 - 1))$ times in total. In particular, a set of n words of length ℓ satisfies Constraints $C_1(k_1)$ and $C_4(k_4)$ if and only if it satisfies Inequalities (1) and (2) exactly $\binom{n}{2} \cdot (1 + 2(k_4 - 1))$ times and fails 0 time. Furthermore, when ℓ is sufficiently large, an empty $n \times \ell$ binary matrix is expected to satisfy Inequalities (1) and (2) strictly greater than $\binom{n}{2} \cdot (1 + 2(k_4 - 1)) - 1$ times. With this lower bound and the linearity of expectations, Algorithm DetWords can choose to fill each empty entry with 0 or 1 one at a time to arrive at a set of n words of length ℓ which satisfies Inequalities (1) and (2) exactly $\binom{n}{2} \cdot (1 + 2(k_4 - 1))$ times and thus satisfies Constraints $C_1(k_1)$ and $C_4(k_4)$. That is, Algorithm DetWords chooses to fill an empty entry with 0 or 1 whichever yields a larger expected number of times Inequalities (1) and (2) are satisfied.

To choose a sufficiently large ℓ for Algorithm DetWords, let δ be any positive real number. Let $c_1 = 2 + \delta$. Let $c_2 = \frac{c_1}{2}\left\{\log\left(\frac{c_1}{(c_1-2)\ln 2}\right) + 2.5 - \frac{1}{\ln 2}\right\}$. Let $k = \max\{k_1, k_4\}$. Let $\ell^\star = \lceil c_1 \log n + c_2 k \rceil$. Theorem 9 below shows that, by setting $\ell = \ell^\star$, Algorithm DetWords deterministically constructs a code $\mathcal{W}_{1,4}$ of n DNA words of length $\ell^\star$ that satisfies constraints $C_1(k_1)$ and $C_4(k_4)$. Theorem 9 also shows that this construction takes $O(n^2(\ell^\star)^3)$ time.

The remainder of this section provides details to elaborate on the above overview. In Section 3.1, we define a polynomial-time computable expectation that will be used by Algorithm DetWords for the purpose of derandomization. In Section 3.2, we give Algorithm DetWords in Algorithm 1. The word length $\ell^\star$ above is determined analytically and for the binary alphabet; in Section 3.3, we discuss how to improve this word length computationally and with a larger alphabet, i.e., the DNA

alphabet.

3.1 A Polynomial-Time Computable Expectation for Derandomization

To describe Algorithm DetWords in Algorithm 1, we first give some definitions and lemmas.

Definition 1. Given n, ℓ, k_1, and k_4, an $n \times \ell$ binary matrix M is called a (k_1, k_4)-*distance matrix* if the set of the n rows of M satisfies constraints $C_1(k_1)$ and $C_4(k_4)$.

Lemma 2. *An (k_1, k_4)-distance matrix M of dimension $n \times \ell$ can be converted into a code $\mathcal{W}_{1,4}$ of n DNA words of length ℓ that satisfies $C_1(k_1)$ and $C_4(k_4)$.*

Proof. As discussed in the overview at the start of this section, we first view the rows of M as a code of n binary words of length ℓ. Then, we convert these binary words into DNA words by replacing 0 and 1 with two distinct DNA characters. □

Definition 2. Let M be an $n \times \ell$ matrix, where each (p, q)-th entry is 0, 1, or a distinct unknown $x_{p,q}$. Such a matrix is called a *partially assigned* matrix.

Now consider a partially assigned matrix M of dimension $n \times \ell$ as a random variable where each unknown $x_{p,q}$ can assume the value of 0 or 1 with equal probability 1/2. Next consider the expected number of ordered pairs of distinct rows r_α and r_β in M that satisfy constraints $C_1(k_1)$ and $C_4(k_4)$ where $Y = r_\alpha$ and $X = r_\beta$. As a first attempt [14], we have wished to use this expectation in Algorithm DetWords for the purpose of randomization. However, it is not clear how to compute this expectation in polynomial time. Therefore, in Algorithm DetWords, we will use a different expectation $\text{ExpCount}(M, k_1, k_4)$ that also works for derandomization but can be computed in polynomial time. The expectation $\text{ExpCount}(M, k_1, k_4)$ is developed as follows.

- $E_1(M, \alpha, \beta, k_1)$ denotes the event that r_α and r_β satisfy Inequality (1) for $C_1(k_1)$ with $Y = r_\alpha$ and $X = r_\beta$.
- $E_4(M, \alpha, \beta, k_4, i)$ denotes the event that r_α and r_β satisfy case i of Inequality (2) for $C_4(k_4)$ $Y = r_\alpha$ and $X = r_\beta$.
- $\text{ExpE}_1(M, k_1)$ denotes the expected number of unordered pairs of distinct α and β for which $E_1(M, \alpha, \beta, k_1)$ holds.
- $\text{ExpE}_4(M, k_4, i)$ denotes the expected number of ordered pairs of distinct α and β for which $E_4(M, \alpha, \beta, k_4, i)$ holds.

Note that for $\text{ExpE}_1(M, k_1)$, we count unordered pairs of α and β but for $\text{ExpE}_4(M, k_4, i)$, we count ordered pairs. This difference is due to the following reasons. Y and X are symmetric in Inequality (1); therefore, α and β are symmetric for E_1. In contrast, Y and X are symmetric in Inequality (2) only for $i = \ell$ but asymmetric for all other i; therefore α and β are symmetric for E_1 only for $i = \ell$ but asymmetric for all other i.

Now, let

$$\text{ExpCount}(M, k_1, k_4) = \text{ExpE}_1(M, \max\{k_1, k_4\}) + \sum_{i=\ell-k_4+1}^{\ell-1} \text{ExpE}_4(M, k_4, i) \tag{3}$$

Note that in the right-hand side of Equality (3), the second argument of ExpE_1 is $\max\{k_1, k_4\}$ rather than k_1 as used in the definition of constraint $C_1(k_1)$. Also, the upper limit of the summation is

$\ell - 1$ rather than ℓ and the lower limit is $\ell - k_4 + 1$ rather than $\ell - k_4$ as used in the definition of constraint $C_4(k_4)$. We will justify these details in Lemma 3 and its proof below.

We next develop two expressions for $\text{ExpCount}(M, k_1, k_4)$ as alternatives to Equality (3) in order to analyze and efficiently compute $\text{ExpCount}(M, k_1, k_4)$.

For an event E of a probability space, let $\overline{E}$ denote the complement of E, and let $\Pr(E)$ denote the probability of E. For a real-valued random variable V, let $\text{Exp}(V)$ denote the expectation of V.

Equalities (4) and (5) below in conjunction with Equality (3) give one of two alternative expressions for $\text{ExpCount}(M, k_1, k_4)$.

$$\text{ExpE}_1(M, \max\{k_1, k_4\}) = \sum_{1 \le \alpha < \beta \le n} \left\{1 - \Pr\left(\overline{E_1(M, \alpha, \beta, \max\{k_1, k_4\})}\right)\right\}; \tag{4}$$

$$\begin{aligned}\text{ExpE}_4(M, k_4, i) = \sum_{1 \le \alpha < \beta \le n} \Big\langle &\left\{1 - \Pr\left(\overline{E_4(M, \alpha, \beta, k_4, i)}\right)\right\} + \\ &\left\{1 - \Pr\left(\overline{E_4(M, \beta, \alpha, k_4, i)}\right)\right\}\Big\rangle.\end{aligned} \tag{5}$$

For $k_1, k_4, k = \max\{k_1, k_4\}$, and a binary matrix M' of dimension $n \times \ell$, consider the following two functions:

- $V_1(M', k)$ denotes the number of unordered pairs of distinct α and β such that rows r'_α and r'_β of M' satisfy Inequality (1) for $C_1(k)$ with $Y = r'_\alpha$ and $X = r'_\beta$.
- $V_4(M', k_4)$ denotes the number of triplets (α, β, i) such that distinct rows r'_α and r'_β in M' satisfy case i of Inequality (2) for $C_4(k_4)$ with $Y = r_\alpha$ and $X = r_\beta$, where $n \ge \alpha \ne \beta \ge 1$ and $\ell - 1 \ge i \ge \ell - k_4 + 1$.

Note that V_1 is an integer function and $\binom{n}{2} \ge V_1(M', k) \ge 0$. Similarly, V_4 is an integer function and $n(n-1)\cdot(k_4 - 1) \ge V_4(M', k_4) \ge 0$. Consequently, $V_1(M', k) + V_4(M', k_4)$ is an integer and $\binom{n}{2} \cdot (1 + 2(k_4 - 1)) \ge V_1(M', k) + V_4(M', k_4) \ge 0$.

Next we combine the random variable M and the functions V_1 and V_4 to form two random variables $V_1(M, k)$ and $V_4(M, k_4)$. Then, the following equalities give the other alternative expression for $\text{ExpCount}(M, k_1, k_4)$.

$$\text{ExpE}_1(M, \max\{k_1, k_4\}) = \text{Exp}(V_1(M, k)); \tag{6}$$

$$\sum_{i=\ell-k_4+1}^{\ell-1} \text{ExpE}_4(M, k_4, i) = \text{Exp}(V_4(M, k_4)); \tag{7}$$

$$\text{ExpCount}(M, k_1, k_4) = \text{Exp}(V_1(M, k)) + \text{Exp}(V_4(M, k_4)). \tag{8}$$

Lemmas 3 through 5 below analyze $\text{ExpCount}(M, k_1, k_4)$.

Lemma 3. *Let M be a partially assigned matrix of dimension $n \times \ell$. If*

$$\text{ExpCount}(M, k_1, k_4) > \binom{n}{2} \cdot (1 + 2(k_4 - 1)) - 1, \tag{9}$$

then there exists an assignment of 0's and 1's to the unknowns in M so that the resulting binary matrix M' is a (k_1, k_4)-distance matrix.

Proof. Recall that for every binary matrix M'' generated from M, $V_1(M'', k) + V_4(M'', k_4)$ is an integer and $\binom{n}{2} \cdot (1 + 2(k_4 - 1)) \geq V_1(M'', k) + V_4(M'', k_4)$. Therefore, Inequalities (9) and (8) imply that there exists a binary matrix M' generated from M such that $V_1(M', k) + V_4(M', k_4) = \binom{n}{2} \cdot (1 + 2(k_4 - 1))$. Then, since $\binom{n}{2} \geq V_1(M', k)$ and $n(n-1) \cdot (k_4 - 1) \geq V_4(M', k_4)$, we have $V_1(M', k) = \binom{n}{2}$ and $V_4(M', k) = n(n-1) \cdot (k_4 - 1)$.

Next, since $V_1(M', k) = \binom{n}{2}$ and there are $\binom{n}{2}$ unordered pairs of distinct rows in M', the n rows of the binary matrix M' satisfy $C_1(k)$. Since $k = \max\{k_1, k_4\}$, by Lemma 1(2), the n rows of M' satisfy $C_1(k_1)$.

Likewise, the n rows of M' satisfy $C_1(k_4)$. Now observe that Inequality (1) for $C_1(k_4)$ is the same as case $i = \ell$ in Inequality (2) for $C_4(k_4)$. Therefore, the n rows of M' satisfy case $i = \ell$ in Inequality (2) for $C_4(k_4)$ as well. Next, since $V_4(M', k_4) = n(n-1) \cdot (k_4 - 1)$ and there are $n(n-1) \cdot (k_4 - 1)$ triplets (α, β, i) with $n \geq \alpha \neq \beta \geq 1$ and $\ell - 1 \geq i \geq \ell - k_4 + 1$, the n rows of M' satisfy Inequality (2) of $C_4(k_4)$ for $\ell - 1 \geq i \geq \ell - k_4 + 1$. Furthermore, since case $i = \ell - k_4$ in Inequality (2) for $C_4(k_4)$ always holds, the n rows of M' satisfy the entire $C_4(k_4)$ constraint as well.

In sum, the n rows of M' satisfy both constraints $C_1(k_1)$ and $C_4(k_4)$. This finishes the proof. □

Lemma 4. *Let M be a partially assigned matrix of dimension $n \times \ell$. Assume that the (p, q)-th entry of M is an unknown. Let M_0 (respectively, M_1) be M with the (p, q)-th entry assigned 0 (respectively, 1). Then*

$$\text{ExpCount}(M, k_1, k_4) = \frac{1}{2} \cdot \text{ExpCount}(M_0, k_1, k_4) + \frac{1}{2} \cdot \text{ExpCount}(M_1, k_1, k_4).$$

Proof. This lemma follows from Equality (8), the linearity of expectations $\text{Exp}(V_1(M, k))$ and $\text{Exp}(V_4(M, k_4))$, and the fact that M is considered a random variable where each of the unknown entries is independently assigned 0 or 1 with equal probability 1/2. □

Lemma 5. *Let $k = \max\{k_1, k_4\}$. Given $r_\alpha, r_\beta, k_1, k_4$, and i as the input, each of the probabilities in the right-hand sides of Equalities (4) and (5) can be computed in $O(\ell + k)$ time.*

Proof. The specified probabilities can be computed in essentially the same manner. Here, we only show how to compute $\Pr\left(\overline{E_1(M, \alpha, \beta, \max\{k_1, k_4\})}\right)$ in the desired time complexity. Let s be the number of positions at which r_α and r_β assume values of 0 or 1 and are not unknowns. Let t be the number of these s positions where r_α and r_β assume different binary values. Then,

$$\Pr\left(\overline{E_1(M, \alpha, \beta, \max\{k_1, k_4\})}\right) = \sum_{j=0}^{k-1-t} \binom{\ell - s}{j} \left(\frac{1}{2}\right)^{\ell - s}. \tag{10}$$

It is elementary to first determine s and then compute the right-hand side of Equality (10) in $O(\ell + \log(\ell - s) + k - t)$ total time, which is $O(\ell + k)$ time. □

3.2 Algorithm DetWords for Designing Words for $C_1(k_1)$ and $C_4(k_4)$

With $\text{ExpCount}(M, k_1, k_4)$ defined and analyzed in Section 3.1, we describe Algorithm DetWords in Algorithm 1.

We analyze the correctness and computational complexity of Algorithm DetWords (Algorithm 1) with several lemmas and a theorem below. Lemmas 6 and 7 first analyze the existence of (k_1, k_4)-distance matrices.

Algorithm 1 DetWords(n, ℓ, k_1, k_4)

Input: integers n, ℓ, k_1, and k_4.
Output: a (k_1, k_4)-distance matrix M of dimension $n \times \ell$.
Steps:
Construct a partially assigned matrix M of dimension $n \times \ell$ where every entry is an unknown.
for $p = 1$ to ℓ **do**
  for $q = 1$ to n **do**
    Compute $\mathrm{E}_0 = \mathrm{ExpCount}(M_0, k_1, k_4)$, where M_0 is M with the unknown at the (p, q)-th entry set to 0.
    Compute $\mathrm{E}_1 = \mathrm{ExpCount}(M_1, k_1, k_4)$, where M_1 is M with the unknown at the (p, q)-th entry set to 1.
    if $\mathrm{E}_0 \geq \mathrm{E}_1$ **then**
      Update M by setting the unknown at the (p, q)-th entry to 0.
    else
      Update M by setting the unknown at the (p, q)-th entry to 1.
    end if
  end for
end for
return M, which is now a binary matrix.

Lemma 6. *Given n, k_1, k_4, and $k = \max\{k_1, k_4\}$, if ℓ satisfies the following two inequalities*

$$2k \leq \ell \tag{11}$$

$$0 < \ell - k\log e - k\log\frac{\ell}{k} - 2\log n + 2\log k, \tag{12}$$

then ℓ satisfies Inequality (9) in Lemma 3 and thus there exists a (k_1, k_4)-distance matrix of dimension $n \times \ell$.

Proof. Throughout this proof, we assume $\ell \geq 2k$. Consider a partially assigned matrix M of dimension $n \times \ell$ where every entry is an unknown. To prove this lemma by means of Equalities (3), (4), and (5), we will solve for ℓ the following equivalent inequality of Inequality (9):

$$\begin{aligned}
&\binom{n}{2}\cdot(1 + 2(k_4 - 1)) - 1 \\
< &\sum_{1\leq\alpha<\beta\leq n} \left\{1 - \Pr\left(\overline{E_1(M, \alpha, \beta, \max\{k_1, k_4\})}\right)\right\} \\
&+ \sum_{i=\ell-k_4+1}^{\ell-1} \sum_{1\leq\alpha<\beta\leq n} \left\langle \left\{1 - \Pr\left(\overline{E_4(M, \alpha, \beta, k_4, i)}\right)\right\} + \left\{1 - \Pr\left(\overline{E_4(M, \beta, \alpha, k_4, i)}\right)\right\}\right\rangle .
\end{aligned} \tag{13}$$

Simplifying the above inequality, we have the following equivalent inequality:

$$\begin{aligned}
1 > &\sum_{1\leq\alpha<\beta\leq n} \Pr\left(\overline{E_1(M, \alpha, \beta, \max\{k_1, k_4\})}\right) \\
&+ \sum_{i=\ell-k_4+1}^{\ell-1} \sum_{1\leq\alpha<\beta\leq n} \left\langle \Pr\left(\overline{E_4(M, \alpha, \beta, k_4, i)}\right) + \Pr\left(\overline{E_4(M, \beta, \alpha, k_4, i)}\right)\right\rangle .
\end{aligned}$$

Working out the probabilities in the above inequality, we have the following equivalent inequality:

$$1 > \binom{n}{2}\sum_{j=0}^{k-1}\binom{\ell}{j}2^{-\ell} + \sum_{i=\ell-k_4+1}^{\ell-1}\binom{n}{2}\cdot 2\cdot\left\{\sum_{j=0}^{k_4-(\ell-i)-1}\binom{i}{j}2^{-i}\right\}.$$

Simplifying the above inequality, we have the following equivalent inequality:

$$1 > \binom{n}{2}\left(\sum_{j=0}^{k-1}\binom{\ell}{j}2^{-\ell} + 2\cdot\sum_{i=\ell-k_4+1}^{\ell-1}\sum_{j=0}^{k_4-(\ell-i)-1}\binom{i}{j}2^{-i}\right). \tag{14}$$

Next, replacing k_4 by k in Inequality (14) and moving the terms on the right-hand side to the left-hand side, we obtain the following non-equivalent inequality:

$$1 - \binom{n}{2}\left(\sum_{j=0}^{k-1}\binom{\ell}{j}2^{-\ell} + 2\cdot\sum_{i=\ell-k+1}^{\ell-1}\sum_{j=0}^{k-(\ell-i)-1}\binom{i}{j}2^{-i}\right) > 0. \tag{15}$$

Note that if ℓ satisfies Inequality (15), then ℓ satisfies Inequality (14) and thus Inequalities (13) and (9). Therefore, we will now solve Inequality (15) for ℓ as follows.

We will find a lower bound of the left-hand side of Inequality (15). For this purpose, we first bound the term in the rightmost summation of Inequality (15). Since $\ell \geq 2k$ and $\ell - 1 \geq i$, we have $i \geq 2\cdot(k-(\ell-i))$ and thus

$$\binom{i}{j} \leq \binom{i}{k-(\ell-i)}. \tag{16}$$

Furthermore, for all integers s with $\ell - i - 1 \geq s \geq 0$, since

$$\frac{i+(s+1)}{k-(\ell-i)+(s+1)} \geq 2,$$

we have

$$\begin{aligned}\binom{i+(s+1)}{k-(\ell-i)+(s+1)}\cdot\frac{1}{2} &= \binom{i+s}{k-(\ell-i)+s}\frac{i+(s+1)}{k-(\ell-i)+(s+1)}\cdot\frac{1}{2} \\ &\geq \binom{i+s}{k-(\ell-i)+s}\end{aligned} \tag{17}$$

By applying Inequality (16) once and applying Inequality (17) iteratively $\ell - i$ times, we have

$$\binom{i}{j}2^{-i} \leq \binom{\ell}{k}2^{-\ell}. \tag{18}$$

This finishes the bounding of the term in the rightmost summation of Inequality (15).

We next bound the term in the leftmost summation of Inequality (15). Since $\ell \geq 2k$, we have

$$\binom{\ell}{j} \leq \binom{\ell}{k}. \tag{19}$$

Plugging Inequalities (19) and (18) into the left-hand side of Inequality (15), we have

$$\begin{aligned}
& 1-\binom{n}{2}\left(\sum_{j=0}^{k-1}\binom{\ell}{j}2^{-\ell}+2\cdot\sum_{i=\ell-k+1}^{\ell-1}\sum_{j=0}^{k-(\ell-i)-1}\binom{i}{j}2^{-i}\right) \\
\geq\ & 1-\binom{n}{2}\left(k\cdot\binom{\ell}{k}2^{-\ell}+2\cdot(k-1)k\cdot\binom{\ell}{k}2^{-\ell}\right) \\
\geq\ & 1-n^2k^2\cdot\binom{\ell}{k}2^{-\ell} \\
\geq\ & 1-n^2k^2\cdot\left(\frac{e\ell}{k}\right)^k 2^{-\ell} \quad \left(\text{since } \binom{\ell}{k}\leq\left(\frac{e\ell}{k}\right)^k\right). \qquad (20)
\end{aligned}$$

Now consider the following inequality:

$$1-n^2k^2\cdot\left(\frac{e\ell}{k}\right)^k 2^{-\ell}>0. \qquad (21)$$

Note that by Inequality (20), if ℓ satisfies Inequality (21), then ℓ satisfies Inequalities (15), (14), (13), and (9). Consequently, the lemma follows from the fact that Inequality (21) is equivalent to Inequality (12). □

Lemma 7 below solves Inequalities (11) and (12) in Lemma 6 for a useful range of ℓ.

Lemma 7. *Given $n\geq 2$, k_1, k_4, and $k=\max\{k_1,k_4\}\geq 1$, if we set*

$$c_1=2+\delta \text{ for any real } \delta>0$$

and

$$c_2=\frac{c_1}{2}\left\{\log\left(\frac{c_1}{(c_1-2)\ln 2}\right)+2.5-\frac{1}{\ln 2}\right\}>0,$$

then $\ell^\star=\lceil c_1\log n+c_2k\rceil\geq 2k$ satisfies Inequalities (11) and (12) in Lemma 6, and thus there exists a (k_1,k_4)-distance matrix of dimension $n\times\ell^\star$. (As examples, when $\delta=1$, $\ell^\star=\lceil 3\log n+4.76k\rceil$; and when $\delta=0.1$, $\ell^\star=\lceil 2.1\log n+6.28k\rceil$.)

Proof. Since $c_2\geq 2$ by calculus, we have $\ell^\star\geq 2k$, satisfying Inequality (11). Below we prove that $\ell^\star$ satisfies Inequality (12). Consider the function $f(x)=(c_2-2.5)+(c_1-2)x-\log(c_2+c_1x)$ and let $z^\star=\frac{\log n}{k}$. Next observe that

$$\begin{aligned}
0 &\leq f(z^\star)=(c_2-2.5)+(c_1-2)\cdot\tfrac{\log n}{k}-\log\left(c_2+c_1\cdot\tfrac{\log n}{k}\right) \\
\Longrightarrow\ 0 &\leq (c_2-2.5)k+(c_1-2)\log n-k\log\left(c_2+\tfrac{c_1\log n}{k}\right) \\
\Longrightarrow\ 0 &< (c_2k+c_1\log n)-k\log e-k\log\left(\tfrac{c_2k+c_1\log n}{k}\right)-2\log n-2\log k \\
\Longrightarrow\ 0 &< \ell^\star-k\log e-k\log\left(\tfrac{\ell^\star}{k}\right)-2\log n-2\log k.
\end{aligned}$$

Thus if $f(z^\star)\geq 0$, then $\ell^\star$ satisfies Inequality (12). To prove $f(z^\star)\geq 0$, we next solve the following equation:

$$\begin{aligned}
& 0 = f'(x) \\
\Longleftrightarrow\quad & 0 = (c_1-2)-\tfrac{c_1}{(c_2+c_1x)\ln 2} \\
\Longleftrightarrow\quad & c_2+c_1x = \tfrac{c_1}{(c_1-2)\ln 2} \\
\Longleftrightarrow\quad & x = \tfrac{1}{(c_1-2)\ln 2}-\tfrac{c_2}{c_1}.
\end{aligned}$$

Continuing the proof for $f(z^\star) \geq 0$, observe that since $f''(x) = \frac{(c_1)^2}{(c_2+c_1x)^2 \ln 2} > 0$, the minimum functional value of $f(x)$ occurs at $x_{\min} = \frac{1}{(c_1-2)\ln 2} - \frac{c_2}{c_1}$. Now, to show $f(z^\star) \geq 0$, we only need to show $f(x_{\min}) \geq 0$ by observing that the following four inequalities are all equivalent to $f(x_{\min}) \geq 0$.

$$\begin{aligned}
0 &\leq (c_2 - 2.5) + (c_1 - 2)\left(\frac{1}{(c_1-2)\ln 2} - \frac{c_2}{c_1}\right) - \log\left(c_2 + c_1\left(\frac{1}{(c_1-2)\ln 2} - \frac{c_2}{c_1}\right)\right) \\
0 &\leq (c_2 - 2.5) + \frac{1}{\ln 2} - \frac{c_2(c_1-2)}{c_1} - \log\left(c_2 + \frac{c_1}{(c_1-2)\ln 2} - c_2\right) \\
0 &\leq c_2 - 2.5 + \frac{1}{\ln 2} - c_2 + \frac{2c_2}{c_1} - \log\left(\frac{c_1}{(c_1-2)\ln 2}\right) \\
c_2 &\geq \frac{c_1}{2}\left\{\log\left(\frac{c_1}{(c_1-2)\ln 2}\right) + 2.5 - \frac{1}{\ln 2}\right\} \qquad (22)
\end{aligned}$$

The lemma follows from the fact that Inequality (22) follows from the definition of c_2. □

Lemma 8 below sets up the base case and the induction step of the iterative derandomization process of Algorithm DetWords in Algorithm 1.

Lemma 8. *Given $n \geq 2$, k_1, k_4, and $k = \max\{k_1, k_4\} \geq 1$, if we set $\ell = \ell^\star$, then the following statements hold for Algorithm* DetWords*.*

1. *(Base Case) At the end of Line 4 of Algorithm 1, the matrix M satisfies Inequality (9) in Lemma 3, namely,*

$$\text{ExpCount}(M, k_1, k_4) > \binom{n}{2} \cdot (1 + 2(k_4 - 1)) - 1.$$

2. *(Induction Step) For each of the $n\ell^\star$ iterations of the nested for-loops in Algorithm 1, at the end of Line 13, the matrix M also satisfies the above inequality.*

Proof.
Statement 1 follows from Lemmas 7, 6, and 3.
Statement 2 follows from Statement 1 and Lemma 4. □

Theorem 9 below summarizes the performance of Algorithm DetWords.

Theorem 9. *Given $n \geq 2$, k_1, k_4, and $k = \max\{k_1, k_4\} \geq 1$, if we set $\ell = \ell^\star$, then the following statements hold for Algorithm* DetWords*.*

1. *Algorithm 1 outputs a code $\mathcal{W}_{1,4}(n, \ell^\star, k_1, k_4)$ of n binary words (i.e., DNA words) of length $\ell^\star$ that satisfies $C_1(k_1)$ and $C_4(k_4)$.*

2. *The word length $\ell^\star$ is within a constant multiplicative factor of the smallest possible word length for a code of n binary words of equal length that satisfies $C_1(k_1)$ and $C_4(k_4)$.*

3. *Algorithm 1 runs in $O(n^2(\ell^\star)^3)$ time.*

Proof.
Statement 1. This statement follows from Lemmas 8 and 3 and the fact that the matrix output by Algorithm 1 is a binary matrix (i.e., has no unknowns).
Statement 2. This statement follows from the definition of $\ell^\star$ and Lemma 1(4).
Statement 3. We first analyze the running times of steps in Algorithm 1 as follows.

1. Line 4 takes $O(n\ell^\star)$ time to generate the initial M.

2. Then for each of the $n\ell^\star$ iterations of the nested for-loops to compute E_0, Line 7 does not explicitly compute M_0. Instead, Algorithm 1 will first compute $\mathrm{ExpCount}(M, k_1, k_4)$ for the initial M where every entry is an unknown. This initialization task takes $O(n^2(\ell^\star)^2)$ time by Lemma 5 and Equalities (3), (4), and (5). Then, Line 7 will update E_0 incrementally by recomputing

$$\Pr\left(\overline{E_1(M, \alpha, \beta, \max\{k_1, k_4\})}\right), \Pr\left(\overline{E_4(M, \alpha, \beta, k_4, i)}\right), \text{ and } \Pr\left(\overline{E_4(M, \beta, \alpha, k_4, i)}\right) \quad (23)$$

   for $\alpha = q$, all $\beta \neq q$ with $n \geq \beta \geq 1$, and all i with $\ell^\star - 1 \geq i \geq \ell^\star - k_4 + 1$. By Lemma 5, these recomputations and thus the incremental updating of E_0 take $O(n(\ell^\star)^2)$ time in total per loop iteration. In sum, the total running time of updating E_0 over the $n\ell^\star$ loop iterations is $O(n^2(\ell^\star)^3)$.

3. Once E_0 is updated, Algorithm 1 will update E_1 at Line 8 in $O(1)$ time per loop iteration using the linearity equality in Lemma 4.

4. Once E_0 and E_1 are updated, Algorithm 1 compares them at Line 9 and then updates M accordingly at Line 10 or 12 in $O(1)$ time per loop iteration.

5. In sum, the total running time of the $n\ell^\star$ iterations of the nested for-loops is dominated by the total running time of updating E_0 over the $n\ell^\star$ loop iterations and thus is $O(n^2(\ell^\star)^3)$ time.

6. Outputting the final matrix M at Line 16 takes $O(n\ell^\star)$ time.

In summary, the time complexity of Algorithm 1 is dominated by the total running time of the nested for-loops and thus is $O(n^2(\ell^\star)^3) = O(n^2(k + \log n)^3)$. □

Technical Remarks In the proof of Statement 3 of Theorem 9, the incremental updating of E_0 at Line 7 can be made somewhat more efficient by modifying the proof of Lemma 5 with more elaborate but still straightforward algorithmic details. Specifically, the right probability in Expression (23) can be updated in $O(k)$ time instead of $O(\ell)$ time. Also, each of the middle and right probabilities in Expression (23) can be updated in $O(k_4 - (\ell^\star - i)) = O(k)$ time instead of $O(\ell)$ time. Thus the total time for incrementally updating E_0 at Line 7 is $O(n\ell^\star k)$ per loop iteration, which is somewhat less than $O(n(\ell^\star)^2)$. For the sake of brevity, we omit the details of these improvements in this paper.

3.3 Improving Word Length $\ell^\star$ Computationally and with a Larger Alphabet

The word length $\ell^\star$ is obtained analytically. In order to make the analysis of $\ell^\star$ manageable, we sacrifice the quality of $\ell^\star$. In this section, we discuss two improvements of $\ell^\star$ by computation.

Improving Word Length $\ell^\star$ Computationally Lemma 11 computationally improves the word length $\ell^\star$ by means of binary search.

Lemma 10. *Let M be a partially assigned matrix of dimension $n \times \ell$ where every entry is an unknown. Given $n, k_1, k_4, k = \max\{k_1, k_4\}$, and ℓ as the input,* $\mathrm{ExpCount}(M, k_1, k_4)$ *can be computed in $O(k + (k_4)^2 + \log \ell)$ time.*

Proof. By Equalities (3), (4), and (5) and a similar analysis to the proof of Lemma 6, we have

$$\begin{aligned}\text{ExpCount}(M, k_1, k_4) &= \binom{n}{2} \cdot (1 + 2(k_4 - 1)) \\ &\quad - \binom{n}{2} \left(\sum_{j=0}^{k-1} \binom{\ell}{j} 2^{-\ell} + 2 \cdot \sum_{i=\ell-k_4+1}^{\ell-1} \sum_{j=0}^{k_4-(\ell-i)-1} \binom{i}{j} 2^{-i} \right).\end{aligned}$$

It is elementary to evaluate the right-hand side of this equality in $O(k + (k_4)^2 + \log \ell)$ time. □

Lemma 11. *Given n, k_1, k_4, and $k = \max\{k_1, k_4\}$ as the input, it takes $O((k + k_4^2 + \log(\log n + k)) \log(\log n + k))$ time to compute the smallest ℓ that satisfies Inequality (9) in Lemma 3, namely,*

$$\text{ExpCount}(M, k_1, k_4) > \binom{n}{2} \cdot (1 + 2(k_4 - 1)) - 1.$$

Proof. By Lemmas 7, 6, and 3, we use $\ell^\star$ as the initial upper bound for the desired smallest ℓ. We then use binary search and Lemma 10 to find this smallest desired ℓ. This search process takes $O(\log \ell^\star)$ applications of Lemma 10 and thus runs in $O((k + (k_4)^2 + \log \ell^\star) \log \ell^\star)$ time, which is $O((k + k_4^2 + \log(\log n + k)) \log(\log n + k))$ time . □

Further Improving Word Length $\ell^\star$ with a Larger Alphabet The smallest ℓ obtained by Lemma 11 can be further improved by replacing the binary alphabet with the DNA alphabet in the definition of a partially assigned matrix and modifying Algorithm 1 accordingly. This alphabet change will shorten the smallest ℓ obtained by Lemma 11 because it is intuitive to show that a random DNA matrix of dimension $n \times \ell$ has a larger probability to be a (k_1, k_4)-distance matrix than a random binary matrix of the same dimension. The analysis of the performance of such a modified Algorithm 1 remains essentially the same, and the smallest desired ℓ to input into the modified Algorithm 1 can be computed in the same manner and time complexity as by Lemma 11. For the sake of brevity, we omit the details of this modification.

4 Designing Words for More Constraints

In this section, we give deterministic polynomial-time algorithms to construct short DNA words for the following subsets of the constraints $C_1, \ldots, C_9$ based on Algorithm 1:

- C_1 through C_6 (see Theorem 13 in Section 4.1)
- C_1 through C_7 (see Theorem 15 in Section 4.2);
- C_1, C_2, C_3, C_7, and C_8 (see Theorem 17 in Section 4.3);
- C_1 through C_8 (see Theorems 19 and 21 in Section 4.4); and
- C_1 through C_6, and C_9 (see Theorem 23 in Section 4.5).

For the word constructions in this section, we will use Lemma 1(2) to simplify the constructions, and it follows from Lemma 1(4) that the simplifications do not sacrifice the word length by more than a constant multiplicative factor.

To implement the simplifications, we first clarify the notation $\ell^\star$ by attaching the parameters k_1 and k_4 to it as follows.

Given $n \geq 2$, $k_1 \geq 1$ and $k_4 \geq 1$, let

$$\begin{aligned} \delta &= \text{any positive real,} \\ c_1 &= 2 + \delta, \\ c_2 &= \frac{c_1}{2}\left\{\log\left(\frac{c_1}{(c_1-2)\ln 2}\right) + 2.5 - \frac{1}{\ln 2}\right\}, \text{ and} \\ \ell^\star(k_1, k_4) &= \lceil c_1 \cdot \log n + c_2 \cdot \max\{k_1, k_4\} \rceil. \end{aligned}$$

4.1 Designing Words for Constraints C_1 through C_6

Lemma 12 below shows how to transform a binary code that satisfies $C_1(k_1)$ and $C_4(k_4)$ to a DNA code that satisfies $C_1(k_1)$ through $C_6(k_6)$.

Lemma 12.

1. *Let $\mathcal{B}$ be a code of n distinct binary words of equal length $\ell_{1,4}$ that satisfies $C_1(k_1)$ and $C_4(k_4)$. Given $\mathcal{B}$, k_2, k_3, k_5, and k_6 as the input, we can deterministically construct a code $\mathcal{W}_{1\sim6}$ of n distinct DNA words of equal length that satisfies $C_1(k_1)$, $C_2(k_2)$, $C_3(k_3)$, $C_4(k_4)$, $C_5(k_5)$, and $C_6(k_6)$.*

2. *The length of the words in $\mathcal{W}_{1\sim6}$ is $\ell_{1,4} + \max\{k_2, k_3, k_5, k_6\}$.*

3. *The construction takes $O(n(\ell_{1,4} + \max\{k_2, k_3, k_5, k_6\}))$ time.*

Proof. Let $k = \max\{k_2, k_3, k_5, k_6\}$. We construct $\mathcal{W}_{1\sim6}$ with the following steps:

1. Convert the binary code $\mathcal{B}$ into a DNA code by changing 0 to the character A and changing 1 to the character T in each word. Let $\mathcal{D}$ denote the set of the new words.

2. Append k copies of the character C at the left end of each word in $\mathcal{D}$. Let $\mathcal{W}_{1\sim6}$ be the set of the new words.

It is clear that this construction takes $O(n(\ell_{1,4} + k))$ time, proving Statement 3. It is also clear that the words in $\mathcal{W}_{1\sim6}$ have equal length $\ell_{1,4} + k$, proving Statement 2. To prove Statement 1, we observe that the two construction steps are deterministic and $\mathcal{W}_{1\sim6}$ consists of n distinct DNA words of equal length. Below we verify that $\mathcal{W}_{1\sim6}$ satisfies $C_1(k_1)$, $C_2(k_2)$, $C_3(k_3)$, $C_4(k_4)$, $C_5(k_5)$, and $C_6(k_6)$.

- That $\mathcal{W}_{1\sim6}$ satisfies $C_1(k_1)$ and $C_4(k_4)$ follows directly from the assumption that $\mathcal{B}$ satisfies these two constraints.

- To check $C_2(k_2)$ and $C_3(k_3)$, consider two words Y and X in $\mathcal{W}_{1\sim6}$ ($Y \neq X$ for $C_2(k_2)$, but $Y = X$ for $C_3(k_3)$). Since the leftmost k characters in Y are all C. For these two constraints, these C's are compared with A, T, or G in X^{RC}. Therefore, the Hamming distance between Y and X^{RC} is at least k. Since $k \geq k_2$ and $k \geq k_3$, constraints $C_2(k_2)$ and $C_3(k_3)$ hold for $\mathcal{W}_{1\sim6}$.

- To check $C_5(k_5)$ and $C_6(k_6)$, since $k \geq k_5$ and $k \geq k_6$, by Lemma 1(2) we only need to check $C_5(k)$ and $C_6(k)$. Consider two words Y and X in $\mathcal{W}_{1\sim6}$ ($Y \neq X$ for $C_5(k)$, but $Y = X$ for $C_6(k)$). Let ℓ denote $\ell_{1,4} + k$. Also consider i where $\ell \geq i \geq \ell - k$. Let $j = k - (\ell - i)$. From the definitions of the constraints $C_1(k_1)$ through $C_6(k_6)$, we have $\ell \geq k$. Thus, $i \geq j$ and $Y[1 \cdots i]$ has at least j characters, The leftmost j characters of $Y[1 \cdots i]$

are all C. For these two constraints, these C's are compared with characters A, T, or G in $(X[1\cdots i])^{RC}$. Therefore the Hamming distance between $Y[1\cdots i]$ and $(X[1\cdots i])^{RC}$ is at least $j = k - (\ell - i)$, as required by $C_5(k)$ and $C_6(k)$. By a symmetrical argument for the right ends of $(X[(\ell-i+1)\cdots\ell])^{RC}$ and $Y[(\ell-i+1)\cdots\ell]$, the Hamming distance between $Y[(\ell-i+1)\cdots\ell]$ and $(X[(\ell-i+1)\cdots\ell])^{RC}$ is at least $k-(\ell-i)$, as required by $C_5(k)$ and $C_6(k)$.

□

Theorem 13 below uses Theorem 9 and Lemma 12 to show how to construct a DNA code that satisfies $C_1(k_1)$ through $C_6(k_6)$.

Theorem 13.

1. *Given $n \geq 2$, $k_1 \geq 1$, k_2, k_3, k_4, k_5, and k_6 as the input, we can deterministically construct a code $\mathcal{W}_{1\sim6}$ of n distinct DNA words of equal length that satisfies $C_1(k_1)$, $C_2(k_2)$, $C_3(k_3)$, $C_4(k_4)$, $C_5(k_5)$, and $C_6(k_6)$.*
2. *The length of the words in $\mathcal{W}_{1\sim6}$ is $\ell^\star(k_1,k_4) + \max\{k_2,k_3,k_5,k_6\}$.*
3. *The construction takes $T_{1,4}(n,\ell^\star(k_1,k_4),k_1,k_4) + O(n(\log n + \max\{k_1,k_2,k_3,k_4,k_5,k_6\})$ time, where $T_{1,4}(n,\ell^\star(k_1,k_4),k_1,k_4)$ is the running time of the call $\mathrm{DetWords}(n,\ell^\star(k_1,k_4),k_1,k_4)$.*

Proof. We construct $\mathcal{W}_{1\sim6}$ with the following steps:

1. Let $\ell_{1,4} = \ell^\star(k_1,k_4)$.
2. Construct a binary code $\mathcal{B} = \mathrm{DetWords}(n,\ell_{1,4},k_1,k_4)$ by means of Theorem 9.
3. Construct $\mathcal{W}_{1\sim6}$ by means of Lemma 12 using $\mathcal{B}$, k_2, k_3, k_5, and k_6 as the input.

With the above construction, this theorem follows directly from Theorem 9 and Lemma 12.

□

4.2 Designing Words for Constraints C_1 through C_7

Lemma 14 below shows how to transform a binary code that satisfies $C_1(k_1)$ and $C_4(k_4)$ to a DNA code that satisfies $C_1(k_1)$ through $C_7(\gamma)$.

Lemma 14.

1. *Let $\mathcal{B}$ be a code of n distinct binary words of equal length $\ell_{1,4}$ that satisfies $C_1(k_1)$ and $C_4(k_4)$. Given $\mathcal{B}$, k_2, k_3, k_5, k_6, and γ as the input, we can deterministically construct a code $\mathcal{W}_{1\sim7}$ of n distinct DNA words of equal length that satisfies $C_1(k_1)$, $C_2(k_2)$, $C_3(k_3)$, $C_4(k_4)$, $C_5(k_5)$, $C_6(k_6)$, and $C_7(\gamma)$.*
2. *The length of the words in $\mathcal{W}_{1\sim7}$ is $\ell_{1,4} + 2\max\{k_2,k_3,k_5,k_6\}$.*
3. *The construction takes $O(n(\ell_{1,4} + \max\{k_2,k_3,k_5,k_6\}))$ time.*

Proof. Let $k = \max\{k_2,k_3,k_5,k_6\}$. Let $\ell = \ell_{1,4} + 2k$. We construct $\mathcal{W}_{1\sim7}$ with the following steps:

1. Append k copies of 1 to each of the left and right ends of each word in $\mathcal{B}$. Let $\mathcal{B}'$ denote the set of the new binary words of equal length ℓ.

2. Choose $\lceil \gamma\ell \rceil$ arbitrary (e.g., evenly distributed) positions among $1, \ldots, \ell$ (see [16]).

3. For each word in $\mathcal{B}'$, at each of the above chosen $\lceil \gamma\ell \rceil$ positions, change 0 to C and change 1 to G, while at all the other positions, change 0 to A and change 1 to T. Let $\mathcal{W}_{1\sim7}$ be the set of the resulting DNA words (see [16]).

With the above construction, Statements 2 and 3 clearly hold. To prove Statement 1, observe that the above construction steps are deterministic and $\mathcal{W}_{1\sim7}$ consists of n distinct DNA words of equal length. Next, by a proof similar to but simpler than that of Lemma 12, $\mathcal{B}'$ satisfies $C_1(k_1)$ to $C_6(k_6)$ as constraints on binary words. Then, since the substitutions at Step 3 do not change Hamming distances for $C_1(k_1)$ and do not decrease Hamming distances for $C_2(k_2)$ through $C_6(k_6)$, these six constraints also hold for $\mathcal{W}_{1\sim7}$. Moreover, it follows from the substitutions at Step 3 that $C_7(\gamma)$ holds for $\mathcal{W}_{1\sim7}$. □

Theorem 15 below uses Theorem 9 and Lemma 14 to show how to construct a DNA code that satisfies $C_1(k_1)$ through $C_7(\gamma)$.

Theorem 15.

1. *Given $n \geq 2$, $k_1 \geq 1$, k_2, k_3, k_4, k_5, k_6, and γ as the input, we can deterministically construct a code $\mathcal{W}_{1\sim7}$ of n distinct DNA words of equal length that satisfies $C_1(k_1)$, $C_2(k_2)$, $C_3(k_3)$, $C_4(k_4)$, $C_5(k_5)$, $C_6(k_6)$, and $C_7(\gamma)$.*

2. *The length of the words in $\mathcal{W}_{1\sim7}$ is $\ell^\star(k_1, k_4) + 2\max\{k_2, k_3, k_5, k_6\}$.*

3. *The construction takes $T_{1,4}(n, \ell^\star(k_1, k_4), k_1, k_4) + O(n(\log n + \max\{k_1, k_2, k_3, k_4, k_5, k_6\})$ time, where $T_{1,4}(n, \ell^\star(k_1, k_4), k_1, k_4)$ is the running time of the call* $\mathrm{DetWords}(n, \ell^\star(k_1, k_4), k_1, k_4)$.

Proof. We construct $\mathcal{W}_{1\sim7}$ with the following steps:

1. Let $\ell_{1,4} = \ell^\star(k_1, k_4)$.

2. Construct a binary code $\mathcal{B} = \mathrm{DetWords}(n, \ell_{1,4}, k_1, k_4)$ by means of Theorem 9.

3. Construct $\mathcal{W}_{1\sim7}$ by means of Lemma 14 using $\mathcal{B}$, k_2, k_3, k_5, k_6, and γ as the input.

With the above construction, this theorem follows directly from Theorem 9 and Lemma 14. □

4.3 Designing Words for Constraints C_1, C_2, C_3, C_7, and C_8

To eliminate long runs in words to satisfy $C_8(d)$, we first detail an algorithm in Algorithm 2, which slightly modifies a similar algorithm of Kao et al. [16] to increase symmetry. Given a binary word X and d as the input, this algorithm inserts a character into X at the end of each interval of length $d-1$ from both the left end of X and the right end of X toward the middle of X. The algorithm also inserts two characters at the middle of X. The inserted characters are complementary to the ending character of each interval or complementary to the middle two characters of X. The complementarity of the inserted characters and the spacings of the insertions ensure that the resulting word X' does not have consecutive 0's or consecutive 1's of length more than d. The symmetrical manner in which the inserted characters are added to X facilitates the checking of constraints $C_2(k_2)$ and $C_3(k_3)$.

Lemma 16 below shows how to transform a binary code that satisfies $C_1(k_1)$ to a DNA code that satisfies $C_1(k_1)$, $C_2(k_2)$, $C_3(k_3)$, $C_7(\gamma)$, and $C_8(d)$. The proof of this lemma uses Algorithm 2 to satisfy $C_8(d)$.

Algorithm 2 BreakRuns(X, d)

Input: a binary word $X = x_1x_2 \dots x_\ell$ of length ℓ and an integer $d \geq 2$, where ℓ is assumed to be even.
Output: a binary word X' of length ℓ' that has at most d consecutive 0's or at most d consecutive 1's, where $\ell' = \ell + 2\lfloor \frac{\ell}{2(d-1)} \rfloor + 2$.
Let $u = d - 1$, $s = \lfloor \frac{\ell}{2u} \rfloor$, $t = su$, and mid $= \frac{\ell}{2}$.
for $1 \leq i \leq s$ **do**
    Let $\hat{\alpha}_i = (x_{iu})^c$ and $\hat{\beta}_i = (x_{\ell-iu+1})^c$.
end for
Let $\hat{\Delta} = (x_{\text{mid}})^c(x_{\text{mid}+1})^c$.
Split X into three segments $L = X[1 \cdots t]$, $U = X[(t+1) \cdots (\ell-t)]$, and $R = X[(\ell-t+1) \cdots \ell]$.
Let $L' = x_1 \dots x_u\hat{\alpha}_1 x_{u+1} \dots x_{2u}\hat{\alpha}_2 x_{2u+1} \dots x_t\hat{\alpha}_s$.
Let $R' = \hat{\beta}_s x_{\ell-t+1} \dots x_{\ell-2u}\hat{\beta}_2 x_{\ell-2u+1} \dots x_{\ell-u}\hat{\beta}_1 x_{\ell-u+1} \dots x_\ell$.
Let $U' = x_{t+1} \dots x_{\text{mid}}\hat{\Delta} x_{\text{mid}+1} \dots x_{\ell-t}$.
Let X' be the concatenation of L', U', and R'.
return X'.

Lemma 16.

1. *Let $\mathcal{B}_0$ be a code of n distinct binary words of equal length ℓ_0 that satisfies $C_1(k_1)$. Given $\mathcal{B}_0$, k_2, k_3, γ, and d as the input, we can deterministically construct a code $\mathcal{W}_{1\sim3,7,8}$ of n distinct DNA words of equal length that satisfies $C_1(k_1)$, $C_2(k_2)$, $C_3(k_3)$, $C_7(\gamma)$, and $C_8(d)$.*

2. *The length of the words in $\mathcal{W}_{1\sim3,7,8}$ is $\frac{d}{d-1}(\ell_0 + 2\max\{k_2, k_3\}) + O(1)$.*

3. *The construction takes $O(n(\ell_0 + \max\{k_2, k_3\}))$ time.*

Proof. Our construction of $\mathcal{W}_{1\sim3,7,8}$ is similar to the construction of $\mathcal{W}_{1\sim6}$ in Lemma 14 with additional work of using Algorithm 2 to break long runs in binary words. Specifically, we construct $\mathcal{W}_{1\sim3,7,8}$ with the following steps:

1. If ℓ_0 is odd, then append 0 at the right end of each word in $\mathcal{B}_0$; otherwise, do not change the words in $\mathcal{B}_0$. Let $\mathcal{B}_1$ be the set of the resulting words. Let ℓ_1 be the length of the resulting words; i.e., if ℓ_0 is odd, then $\ell_1 = \ell_0 + 1$, else $\ell_1 = \ell_0$.

2. Let $k = \max\{k_2, k_3\}$. Append k copies of 1 at each of the left and right ends of each word in $\mathcal{B}_1$. Let $\mathcal{B}_2$ be the set of the new binary words. Let ℓ_2 be the length of the new words; i.e., $\ell_2 = \ell_1 + 2k$.

3. Apply Algorithm 2 to each word in $\mathcal{B}_2$. Let $\mathcal{B}_3$ be the set of the output binary words. Let ℓ_3 be the length of the new words; i.e., $\ell_3 = \ell_2 + 2\lfloor \frac{\ell_2}{2(d-1)} \rfloor + 2 = \frac{d}{d-1}(\ell_0 + 2\max\{k_2, k_3\}) + O(1)$.

4. Choose $\lceil \gamma\ell_3 \rceil$ arbitrary (e.g., evenly distributed) positions among $1, \dots, \ell_3$ (see [16]).

5. For each word in $\mathcal{B}_3$, at each of the above chosen $\lceil \gamma\ell_3 \rceil$ positions, change 0 to C and change 1 to G, while at all the other positions, change 0 to A and change 1 to T. Let $\mathcal{W}_{1\sim3,7,8}$ be the set of the resulting DNA words (see [16]).

With the above construction, Statements 2 and 3 clearly hold. To prove Statement 1, observe that the above construction steps are deterministic and $\mathcal{W}_{1\sim3,7,8}$ consists of n distinct DNA words of equal length. We verify $C_1(k_1)$, $C_2(k_2)$, $C_3(k_3)$, $C_7(\gamma)$, and $C_8(d)$ as follows.

1. Since $\mathcal{B}_0$ satisfies $C_1(k_1)$, $\mathcal{B}_1$ also satisfies $C_1(k_1)$.

2. Next, by a proof similar to but simpler than that of Lemma 12, $\mathcal{B}_2$ satisfies $C_1(k_1)$ through $C_3(k_3)$ as constraints on binary words.

3. From the spacings of the insertions made by Algorithm 2, the insertions made at Step 3 do not decrease Hamming distances for $C_1(k_1)$ through $C_3(k_3)$, $\mathcal{B}_3$ continues to satisfy these three constraints.

4. Further from the spacings and the complementarity of the characters inserted by Algorithm 2, $\mathcal{B}_3$ additionally satisfies $C_8(d)$ as a constraint on binary words.

5. Since the substitutions made at Step 5 do not decrease Hamming stances for $C_1(k_1)$ through $C_3(k_3)$, $\mathcal{W}_{1\sim3,7,8}$ continues to satisfy $C_1(k_1)$ through $C_3(k_3)$.

6. Also, it follows from the substitutions made at Step 5 that $C_7(\gamma)$ holds for $\mathcal{W}_{1\sim3,7,8}$.

7. Finally, these substitutions do not increase lengths of consecutive occurrences of a character, $C_8(d)$ holds for $\mathcal{W}_{1\sim3,7,8}$.

□

Theorem 17 below uses Theorem 9 and Lemma 14 to show how to construct a DNA code that satisfies $C_1(k_1)$, $C_2(k_3)$, $C_3(k_3)$, $C_7(\gamma)$, and $C_8(d)$.

Theorem 17.

1. *Given $n \geq 2$, $k_1 \geq 1$, k_2, k_3, γ, and d as the input, we can deterministically construct a code $\mathcal{W}_{1\sim3,7,8}$ of n distinct DNA words of equal length that satisfies $C_1(k_1)$, $C_2(k_2)$, $C_3(k_3)$, $C_7(\gamma)$, and $C_8(d)$.*

2. *The length of the words in $\mathcal{W}_{1\sim3,7,8}$ is $\frac{d}{d-1}(\ell^\star(k_1,k_1) + 2\max\{k_2,k_3\}) + O(1)$.*

3. *The construction takes $T_{1,4}(n,\ell^\star(k_1,k_1),k_1,k_1) + O(n(\log n + \max\{k_1,k_2,k_3\})$ time, where $T_{1,4}(n,\ell^\star(k_1,k_1),k_1,k_1)$ is the running time of the call* $\mathrm{DetWords}(n,\ell^\star(k_1,k_1),k_1,k_1)$.

Proof. We construct $\mathcal{W}_{1\sim3,7,8}$ with the following steps:

1. Let $\ell_0 = \ell^\star(k_1,k_1)$.

2. Construct a binary code $\mathcal{B}_0 = \mathrm{DetWords}(n,\ell_0,k_1,k_1)$ by means of Theorem 9.

3. Construct $\mathcal{W}_{1\sim3,7,8}$ by means of Lemma 16 using $\mathcal{B}_0$, k_2, k_3, γ, and d as the input.

With the above construction, this theorem follows directly from Theorem 9 and Lemma 16. □

Technical Remarks. We can reduce the word length of $\mathcal{W}_{1\sim3,7,8}$ in Theorem 17(2) by simplifying Algorithm DetWords to satisfy only $C_1(k_1)$ rather than both $C_1(k_1)$ and $C_4(k_1)$. For the sake of brevity, we omit the details of this simplification.

4.4 Designing Words for Constraints C_1 through C_8

This section gives two ways to construct a DNA code that satisfies $C_1(k_1)$ through $C_8(d)$ in Theorems 19 and 21.

Lemma 18 below gives a way to transform a binary code that satisfies $C_1(k_1)$ to a DNA code that satisfies $C_1(k_1)$ through $C_8(d)$.

Lemma 18. *Assume* $\frac{1}{d+1} \leq \gamma \leq \frac{d}{d+1}$.

1. *Let $\mathcal{B}$ be a code of n distinct binary words of equal length ℓ_0 that satisfies $C_1(k_1)$. Given $\mathcal{B}$, k_2, k_3, k_4, k_5, k_6, γ, and d as the input, we can deterministically construct a code $\mathcal{W}_{1\sim 8}$ of n distinct DNA words of equal length that satisfies $C_1(k_1)$, $C_2(k_2)$, $C_3(k_3)$, $C_4(k_4)$, $C_5(k_5)$, $C_6(k_6)$, $C_7(\gamma)$, and $C_8(d)$.*

2. *The length of the words in $\mathcal{W}_{1\sim 8}$ is $\ell_0 + 2\max\{k_2, k_3, k_4, k_5, k_6\}$.*

3. *The construction takes $O(n(\ell_0 + \max\{k_2, k_3, k_4, k_5, k_6\}))$ time.*

Proof. Let $k = \max\{k_2, k_3, k_4, k_5, k_6\}$. Let $\ell = \ell_0 + 2k$. This proof assumes $\gamma \geq \frac{1}{2}$. This assumption is without loss of generality, since if $\gamma < \frac{1}{2}$, we can modify by symmetry the construction steps below to construct a DNA code whose AT content is $1 - \gamma$ fraction of the characters in each word.

We construct $\mathcal{W}_{1\sim 8}$ with the following steps:

1. Append k copies of 1 at each of the left and right ends of each word in $\mathcal{B}$. Let $\mathcal{B}'$ be the set of the new binary words, which have equal length ℓ.

2. Partition the integer interval $[1, \ell]$ into integer subintervals $Z_1, Z_2, \ldots, Z_s$ for some s such that (1) each subinterval consists of at most d integers and at least one integer and (2) the total number of integers in the odd-indexed subintervals is $\lfloor \gamma \ell \rfloor$.

3. For each word in $\mathcal{B}'$, change every 0 (respectively, 1) whose position is in the odd-indexed subintervals to C (respectively, G), and also change every 0 (respectively, 1) whose position is in the even-indexed subintervals to A (respectively, T). Let $\mathcal{W}_{1\sim 8}$ be the set of the resulting DNA words.

We now prove the three statements of this lemma. First of all, Statement 2 clearly holds. As for the other two statements, since $d \geq 2$ and $\frac{1}{2} \leq \gamma \leq \frac{d}{d+1}$, the partition of $[1, \ell]$ at Step 2 exists and can be computed in $O(\ell)$ time in a straightforward manner. With this fact, Statement 3 clearly holds. To prove Statement 1, observe that the above construction steps are deterministic and $\mathcal{W}_{1\sim 8}$ consists of n distinct DNA words of equal length ℓ. We verify $C_1(k_1)$ through $C_8(d)$ as follows.

By an analysis similar to but simpler than the proof of Lemma 12, $\mathcal{B}'$ satisfies $C_1(k_1)$ through $C_6(k_6)$ as constraints on binary words. At Step 3, the substitutions do not change Hamming distances for $C_1(k_1)$ and do not decrease Hamming distances for $C_2(k_2)$ through $C_6(k_6)$, so $\mathcal{W}_{1\sim 8}$ continues to satisfy $C_1(k_1)$ through $C_6(k_6)$. The aggregate size bound of the odd-indexed subintervals at Step 2 ensures that $\mathcal{W}_{1\sim 8}$ additionally satisfies $C_7(\gamma)$. The individual size bounds of the subintervals at Step 2 and the alternating CG-versus-AT substitutions between odd-indexed and even-indexed subintervals at Step 3 ensure that $\mathcal{W}_{1\sim 8}$ satisfies $C_8(d)$ as well. □

Theorem 19 below uses Theorem 9 and Lemma 18 to give our first way to construct a DNA code that satisfies $C_1(k_1)$ through $C_8(d)$.

Theorem 19. *Assume* $\frac{1}{d+1} \leq \gamma \leq \frac{d}{d+1}$.

1. *Given $n \geq 2$, $k_1 \geq 1$, k_2, k_3, k_4, k_5, k_6, γ, and d as the input, we can deterministically construct a code $\mathcal{W}_{1\sim 8}$ of n distinct DNA words of equal length that satisfies $C_1(k_1)$, $C_2(k_2)$, $C_3(k_3)$, $C_4(k_4)$, $C_5(k_5)$, $C_6(k_6)$, $C_7(\gamma)$, and $C_8(d)$.*

2. *The length of the words in $\mathcal{W}_{1\sim 8}$ is $\ell^\star(k_1, k_1) + 2\max\{k_2, k_3, k_4, k_5, k_6\}$.*

3. *The construction takes $T_{1,4}(n, \ell^\star(k_1, k_1), k_1, k_1) + O(n(\log n + \max\{k_1, k_2, k_3, k_4, k_5, k_6\})$ time, where $T_{1,4}(n, \ell^\star(k_1, k_1), k_1, k_1)$ is the running time of the call* $\mathrm{DetWords}(n, \ell^\star(k_1, k_1), k_1, k_1)$.

Proof. We construct $\mathcal{W}_{1\sim 8}$ with the following steps:

1. Let $\ell_0 = \ell^\star(k_1, k_1)$.

2. Construct a binary code $\mathcal{B} = \mathrm{DetWords}(n, \ell_0, k_1, k_1)$ by means of Theorem 9.

3. Construct $\mathcal{W}_{1\sim 8}$ by means of Lemma 18 using $\mathcal{B}$, k_2, k_3, k_4, k_5, k_6, γ, and d as the input.

With the above construction, this theorem follows directly from Theorem 9 and Lemma 18. □

Lemma 20 below gives our second way to transform a binary code that satisfies $C_1(k_1)$ to a DNA code that satisfies $C_1(k_1)$ through $C_8(d)$.

Lemma 20. *Assume $d \geq 3$.*

1. *Let $\mathcal{B}_0$ be a code of n distinct binary words of equal length ℓ_0 that satisfies $C_1(k_1)$. Given $\mathcal{B}_0$, k_2, k_3, k_4, k_5, k_6, γ, and d as the input, we can deterministically construct a code $\mathcal{W}_{1\sim 8}$ of n distinct DNA words of equal length that satisfies $C_1(k_1)$, $C_2(k_2)$, $C_3(k_3)$, $C_4(k_4)$, $C_5(k_5)$, $C_6(k_6)$, $C_7(\gamma)$, and $C_8(d)$.*

2. *The length of the words in $\mathcal{W}_{1\sim 8}$ is $\frac{d}{d-1}\ell_0 + \frac{d}{d+1}2\max\{k_2, k_3, k_4, k_5, k_6\} + O(d)$.*

3. *The construction takes $O(n(\ell_0 + \max\{k_2, k_3, k_4, k_5, k_6\}))$ time.*

Proof. Let $k = \max\{k_2, k_3, k_4, k_5, k_6\}$. We construct $\mathcal{W}_{1\sim 8}$ with the following steps:

1. For $\mathcal{B}_0$, partition each word into $\lceil \frac{\ell_0}{d-1} \rceil$ sub-words of length $d-1$ except that the rightmost sub-word may be shorter. For each sub-word Z, insert a bit at the right end of Z that is complementary to the original rightmost bit of Z. Let $\mathcal{B}_1$ be the set of the new binary words. Let ℓ_1 be the equal length of the new words; i.e., $\ell_1 = \ell_0 + \lceil \frac{\ell_0}{d-1} \rceil = \frac{d}{d-1}\ell_0 + O(1)$.

2. For $\mathcal{B}_1$, append one copy of 1 at the left end of each word and one copy of 0 at the right end of each word. Let $\mathcal{B}_2$ be the set of the new binary words. Let ℓ_2 be the equal length of the new words; i.e., $\ell_2 = \ell_1 + 2 = \frac{d}{d-1}\ell_0 + O(1)$.

3. For $\mathcal{B}_2$, append $\lceil \frac{k}{d-2} \rceil$ copies of length-d binary word $11\cdots110$ at each of the left and right ends of each word. Let $\mathcal{B}_3$ be the set of the new binary words. Let ℓ_3 be the equal length of the new words; i.e., $\ell_3 = \ell_2 + 2\lceil \frac{k}{d-2} \rceil d = \frac{d}{d-1}\ell_0 + \frac{d}{d+1}2k + O(d)$.

4. For $\mathcal{B}_3$, for the leftmost $\lceil \gamma\ell_3 \rceil$ characters in each word, change every 0 (respectively, 1) to C (respectively, G), and for the remaining $\ell_3 - \lceil \gamma\ell_3 \rceil$ characters in each word, change every 0 (respectively, 1) to A (respectively, T). Let $\mathcal{W}_{1\sim 8}$ be the set of the resulting DNA words. The new worlds have equal length ℓ_3.

We now prove the three statements of this lemma. First of all, Statements 2 and 3 clearly hold. To prove Statement 1, observe that the above construction steps are deterministic and $\mathcal{W}_{1\sim 8}$ consists of n distinct DNA words of equal length ℓ_3. We verify $C_1(k_1)$ through $C_8(d)$ as follows.

- Since $\mathcal{B}_0$ satisfies $C_1(k_1)$, the codes $\mathcal{B}_1$, $\mathcal{B}_2$, $\mathcal{B}_3$, and $\mathcal{W}_{1\sim 8}$ all satisfy $C_1(k_1)$.
- That $\mathcal{B}_3$ satisfies $C_2(k_2)$ through $C_6(k_6)$ follows from Step 3 and an analysis similar to the proof of Lemma 12. Consequently, $\mathcal{W}_{1\sim 8}$ also satisfies $C_2(k_2)$ through $C_6(k_6)$.
- From Step 4, $\mathcal{W}_{1\sim 8}$ also satisfies $C_7(\gamma)$.
- From Steps 1 through 3, $\mathcal{B}_3$ satisfies $C_8(d)$. Consequently, $\mathcal{W}_{1\sim 8}$ satisfies $C_8(d)$ as well.

□

Theorem 21 below uses Theorem 9 and Lemma 20 to give our second way to construct a DNA code that satisfies $C_1(k_1)$ through $C_8(d)$.

Theorem 21. *Assume $d \geq 3$.*

1. *Given $n \geq 2$, $k_1 \geq 1$, k_2, k_3, k_4, k_5, k_6, γ, and d as the input, we can deterministically construct a code $\mathcal{W}_{1\sim 8}$ of n distinct DNA words of equal length that satisfies $C_1(k_1)$, $C_2(k_2)$, $C_3(k_3)$, $C_4(k_4)$, $C_5(k_5)$, $C_6(k_6)$, $C_7(\gamma)$, and $C_8(d)$.*
2. *The length of the words in $\mathcal{W}_{1\sim 8}$ is $\frac{d}{d-1}\ell^\star(k_1,k_1) + \frac{d}{d+1}2\max\{k_2,k_3,k_4,k_5,k_6\} + O(d)$.*
3. *The construction takes $T_{1,4}(n,\ell^\star(k_1,k_1),k_1,k_1) + O(n(\log n + \max\{k_1,k_2,k_3,k_4,k_5,k_6\})$ time, where $T_{1,4}(n,\ell^\star(k_1,k_1),k_1,k_1)$ is the running time of the call* $\mathrm{DetWords}(n,\ell^\star(k_1,k_1),k_1,k_1)$.

Proof. We construct $\mathcal{W}_{1\sim 8}$ with the following steps:

1. Let $\ell_0 = \ell^\star(k_1,k_1)$.
2. Construct a binary code $\mathcal{B}_0 = \mathrm{DetWords}(n,\ell_0,k_1,k_1)$ by means of Theorem 9.
3. Construct $\mathcal{W}_{1\sim 8}$ by means of Lemma 20 using $\mathcal{B}_0$, k_2, k_3, k_4, k_5, k_6, γ, and d as the input.

With the above construction, this theorem follows directly from Theorem 9 and Lemma 20. □

Technical Remarks. As with Theorem 17(2), we can reduce the word lengths of $\mathcal{W}_{1\sim 8}$ in Theorems 19(2) and 21(2) by simplifying Algorithm DetWords to satisfy only $C_1(k_1)$ rather than both $C_1(k_1)$ and $C_4(k_1)$.

Furthermore, for the word length formulas in Lemma 20(2) and Theorem 21(2), the left and middle terms in each formula are decreasing functions of d while the right term is an increasing function of d. By Lemma 1(3), we can first computationally find an integer d' such that $d' \geq d$ and d' minimizes the value of the respective length formula and then apply Lemma 20 or Theorem 21 to this d' instead of d to compute $\mathcal{W}_{1\sim 8}$. Analytically, for example, when $d \geq \sqrt{\frac{2\ell}{k}} + 1$, a reasonable initial approximation for d' would be $\sqrt{\frac{2\ell}{k}} + 1$, where $\ell = \ell_0$ for Lemma 20(2) and $\ell = \ell^\star(k_1,k_1)$ for Theorem 21(2).

4.5 Designing Words for Constraints C_1 through C_6, and C_9

We now show how to construct DNA words that satisfy the free energy constraint $C_9(\sigma)$.

Following the approach of Breslauer et al. [8], the *free energy* of a DNA word $X = x_1x_2\ldots x_\ell$ is approximated by the formula

$$\mathrm{FE}(X) = \text{correction factor} + \sum_{i=1}^{\ell-1} \Gamma_{x_i,x_{i+1}},$$

where $\Gamma_{x,y}$ is an integer denoting the *pairwise free energy* between base x and base y.

Building on the work of Kao et al. [16], for simplicity and without loss of generality, we denote the free energy of X to be

$$\mathrm{FE}(X) = \sum_{i=1}^{\ell-1} \Gamma_{x_i,x_{i+1}},$$

with respect to a given *pairwise energy function* Γ. In other words, the correction factor is set to 0.

- Let $\Gamma_{\max}$ and $\Gamma_{\min}$ be the maximum and the minimum of the 16 entries of Γ, respectively.
- Let $D = \Gamma_{\max} - \Gamma_{\min}$.

Theorem 22 below gives a way to transform a DNA code that satisfies $C_1(k_1)$ through $C_6(k_6)$ to a DNA code that satisfies $C_1(k_1)$ through $C_6(k_6)$ and $C_9(4D + \Gamma_{\max})$.

Theorem 22 (Kao, Sanghi, and Schweller [16])**.**

1. *Let $\mathcal{B}_0$ be a code of n distinct DNA words of equal length ℓ_0 that satisfies $C_1(k_1)$, $C_2(k_2)$, $C_3(k_3)$, $C_4(k_4)$, $C_5(k_5)$, and $C_6(k_6)$. There is a deterministic algorithm that takes $\mathcal{B}_0$ and Γ as the input and constructs a code $\mathcal{W}_{1\sim6,9}$ of n distinct DNA words of equal length that satisfies $C_9(4D + \Gamma_{\max})$ in addition to satisfying $C_1(k_1)$ through $C_6(k_6)$.*
2. *The length of the words in $\mathcal{W}_{1\sim6,9}$ is $2\ell_0$.*
3. *The construction takes $O(\min\{n\ell_0 \log \ell_0, \ell_0^{1.5} \log^{0.5} \ell_0 + n\ell_0\})$ time.*

Theorem 23 below uses Theorems 22 and 13 to give a way to construct a DNA code that satisfies $C_1(k_1)$ through $C_6(k_6)$ and $C_9(4D + \Gamma_{\max})$..

Theorem 23.

1. *Given $n \geq 2$, $k_1 \geq 1$, k_2, k_3, k_4, k_5, k_6, and Γ as the input, we can deterministically construct a code $\mathcal{W}_{1\sim6,9}$ of n distinct DNA words of equal length that satisfies $C_1(k_1)$, $C_2(k_2)$, $C_3(k_3)$, $C_4(k_4)$, $C_5(k_5)$, $C_6(k_6)$, and $C_9(4D + \Gamma_{\max})$.*
2. *The length of the words in $\mathcal{W}_{1\sim6,9}$ is $\ell_0 = 2(\ell^\star(k_1, k_4) + \max\{k_2, k_3, k_5, k_6\})$.*
3. *The construction takes $T_{1,4}(n, \ell^\star(k_1, k_4), k_1, k_4) + O(\min\{n\ell_0 \log \ell_0, \ell_0^{1.5} \log^{0.5} \ell_0 + n\ell_0\})$ time, where $T_{1,4}(n, \ell^\star(k_1, k_4), k_1, k_4)$ is the running time of the call $\mathrm{DetWords}(n, \ell^\star(k_1, k_4), k_1, k_4)$.*

Proof. We construct $\mathcal{W}_{1\sim6,9}$ with the following steps:

1. Construct a DNA code $\mathcal{B}_0$ by means of Theorem 13 using n, k_1, k_2, k_3, k_4, k_5, and k_6 as the input.
2. Construct $\mathcal{W}_{1\sim6,9}$ by means of Theorem 22 using $\mathcal{B}_0$ and Γ as the input.

With the above construction, this theorem follows directly from Theorems 22 and 13. □

5 Further Research

In this paper, we have introduced deterministic polynomial-time algorithms for constructing n DNA words that satisfy various subsets of the constraints C_1 through C_9 and have length within a constant multiplicative factor of the shortest possible word length. However, no known algorithm can efficiently construct similarly short words that satisfy all nine constraints. It would be of significance to find efficient algorithms to construct short words that satisfy all nine constraints. Furthermore, it would be of interest to design efficient algorithms to construct short words for other useful constraints. In particular, observe that the constraints C_1 through C_6 are based on pair-wise relations of words. Conceivably, our derandomization techniques are applicable to other classes of codes based on m-wise relations of words for constant m.

Acknowledgments

We thank Robert Brijder, Francis Y. L. Chin, Robert Schweller, Thomas Zeugmann, and the anonymous referees of the conference submission and the journal submission for very helpful insights and comments. We thank the editors for the suggestion to remove the word erratum from the title of the paper and the header of the first section. We thank Matthew King for correcting a sign in the formula in the proof of Lemma 10.

References

[1] L. Adleman. Molecular computation of solutions to combinatorial problems. *Science*, 266:1021–1024, November 1994.

[2] G. Aggarwal, Q. Cheng, M. H. Goldwasser, M. Y. Kao, P. M. de Espanés, and R. T. Schweller. Complexities for generalized models of self-assembly. *SIAM Journal on Computing*, 34(6):1493–1515, 2005.

[3] A. Ben-Dor, R. Karp, B. Schwikowski, and Z. Yakhini. Universal DNA tag systems: A combinatorial design scheme. In *Proceedings of the 4th Annual International Conference on Computational Molecular Biology*, pages 65–75, 2000.

[4] R. S. Braich, C. Johnson, P. W. K. Rothemund, D. Hwang, N. V. Chelyapov, and L. M. Adleman. Solution of a satisfiability problem on a gel-based DNA computer. In A. Condon and G. Rozenberg, editors, *Proceedings of the 6th International Meeting on DNA Based Computers*, pages 27–42. Springer-Verlag, New York, NY, 2001.

[5] A. Brenneman and A. E. Condon. Strand design for bio-molecular computation. *Theoretical Computer Science*, 287(1):39–58, 2001.

[6] S. Brenner. Methods for sorting polynucleotides using oligonucleotide tags. US Patent Number 5,604,097, February 1997.

[7] S. Brenner and R. A. Lerner. Encoded Combinatorial Chemistry. In *Proceedings of the National Academy of Sciences of the USA*, volume 89, pages 5381–5383, June 1992.

[8] K. J. Breslauer, R. Frank, H. Blocker, and L. A. Marky. Predicting DNA duplex stability from the base sequence. In *Proceedings of the National Academy of Sciences of the USA*, volume 83, pages 3746–3750, 1986.

[9] R. Deaton, M. Garzon, R. Murphy, D. R. Franceschetti, and S. E. Stevens. Genetic search of reliable encodings for DNA based computation. In *Proceedings of the 1st Annual Conference on Genetic Programming*, pages 9–15, 1996.

[10] A. G. Frutos, Q. Liu, A. J. Thiel, A. M. W. Sanner, A. E. Condon, L. M. Smith, and R. M. Corn. Demonstration of a word design strategy for DNA computing on surfaces. *Nucleic Acids Research*, 25(23):4748–4757, December 1997.

[11] P. Gaborit and O. D. King. Linear constructions for DNA codes. *Theoretical Computer Science*, 334:99–113, 2005.

[12] M. Garzon, R. Deaton, P. Neathery, D. R. Franceschetti, and R. C. Murphy. A new metric for DNA computing. In *Proceedings of the 2nd Genetic Programming Conference*, pages 472–478. Morgan Kaufman, 1997.

[13] M. H. Garzon, V. Phan, and A. Neel. Optimal DNA codes for computing and self-assembly. *International Journal of Nanotechnology and Molecular Computation*, 1(1):117, 2009.

[14] M. Y. Kao, H. C. M. Leung, H. Sun, and Y. Zhang. Deterministic polynomial-time algorithms for designing short DNA words. In J. Kratochvil and A. Li, editors, *Lecture Notes in Computer Science 6108: Proceedings of the 7th Annual Conference on Theory and Applications of Models of Computation*, pages 308–319. Springer-Verlag, New York, NY, 2010.

[15] M. Y. Kao, M. Sanghi, and R. Schweller. Flexible word design and graph labeling. In T. Asano, editor, *Lecture Notes in Computer Science 4288: Proceedings of the 17th Annual International Symposium on Algorithms and Computation*, pages 48–60. Springer-Verlag, New York, NY, 2006.

[16] M. Y. Kao, M. Sanghi, and R. Schweller. Randomized fast design of short DNA words. *ACM Transactions on Algorithms*, 5(4), October 2009. Article 43, 24 pages.

[17] O. D. King. Bounds for DNA codes with constant GC-content. *Electronic Journal of Combinatorics*, 10:R33, 2003.

[18] A. Marathe, A. Condon, and R. M. Corn. On combinatorial DNA word design. *Journal of Computational Biology*, 8(3):201–219, 2001.

[19] V. Phan and M. H. Garzon. On codeword design in metric DNA spaces. *Natural Computing*, 8(3):571–588, 2008.

[20] D. D. Shoemaker, D. A. Lashkari, D. Morris, M. Mittman, and R. W. Davis. Quantitative phenotypic analysis of yeast deletion mutants using a highly parallel molecular bar-coding strategy. *Nature Genetics*, 14(4):450–456, December 1996.

[21] S. A. Tsaftaris, A. K. Katsaggelos, T. N. Pappas, and E. T. Papoutsakis. DNA computing from a signal processing viewpoint. *IEEE Signal Processing Magazine*, 21(5):100–106, September 2004.

[22] D. C. Tulpan and H. H. Hoos. Hybrid randomised neighbourhoods improve stochastic local search for DNA code design. In Y. Xiang and B. Chaib-draa, editors, *Lecture Notes in Computer Science 2671: Proceedings of the 16th Conference of the Canadian Society for Computational Studies of Intelligence*, pages 418–433. Springer-Verlag, New York, NY, 2003.

[23] D. C. Tulpan, H. H. Hoos, and A. Condon. Stochastic local search algorithms for DNA word design. In M. Hagiya and A. Ohuchi, editors, *Lecture Notes in Computer Science 2568: Proceedings of the 8th International Workshop on DNA-Based Computers*, pages 229–241. Springer-Verlag, New York, NY, 2003.

[24] H. Wang. Proving theorems by pattern recognition. *Bell System Technical Journal*, 40:1–42, 1961.

[25] E. Winfree. *Algorithmic Self-Assembly of DNA*. PhD thesis, California Institute of Technology, Pasadena, CA, 1998.

[26] E. Winfree, F. Liu, L. A. Wenzler, and N. C. Seeman. Design and self-assembly of two-dimensional DNA crystals. *Nature*, 394:539–544, August 1998.