The DNA Word Design Problem: A New Constraint Model and New Results*

Michael Codish and Michael Frank
Ben-Gurion University of the Negev
{mcodish,frankm}@cs.bgu.ac.il

Vitaly Lagoon
Cadence Design Systems, USA
lagoon@cadence.com

Abstract

A fundamental problem in coding theory concerns the computation of the maximum cardinality of a set S of length n code words over an alphabet of size q, such that every pair of code words has Hamming distance at least d, and the set of additional constraints U on S is satisfied. This problem has application in several areas, one of which is the design of DNA codes where $q = 4$ and the alphabet is $\{ A, C, G, T \}$. We describe a new constraint model for this problem and demonstrate that it improves on previous solutions (computes better lower bounds) for various instances of the problem. Our approach is based on a clustering of DNA words into small sets of words. Solutions are then obtained as the union of such clusters. Our approach is SAT based: we specify constraints on clusters of DNA words and solve these using a Boolean satisfiability solver.

1 Introduction

This paper is about the design of error-correcting codes [MacWilliams and Sloane, 1977] for biological applications. The goal is to compute as large as possible a set S of length n code words over the alphabet $\{A, C, G, T\}$ such that every pair of code words has Hamming distance (HD) at least d, and the set of additional constraints U on S is satisfied. The maximum cardinality of such a set S is denoted $A_4^U(n, d)$. The constraints in U typically include a subset of: the reverse complement constraint (RC) which requires that for each pair of words, the Hamming distance between the reverse of one and the Watson-Crick complement of the other is at least d; and a percentage constraint (P) which requires that 50% of letters in each word are from $\{C, G\}$.

The design of DNA code words has its motivation in tasks related to: information storage in DNA computing [Adleman, 1994; Lipton, 1995], storing information in DNA strands [Brenner and Lerner, 1992], DNA computing on surfaces [Frutos *et al.*, 1997], probes in DNA micro-array technologies [Fodor *et al.*, 1991; Schena *et al.*, 1995], use as tracking tags [Qiu *et al.*, 2003], and many others. DNA code words are also used as molecular bar codes enabling to manipulate and identify individual molecules in complex chemical libraries [Shoemaker *et al.*, 1996]

The impact of DNA codes is notable also today. Recent applications of DNA-encoded libraries to drug discovery are described, for example, in [Mullard, 2016; Goodnow Jr *et al.*, 2017; Yuen and Franzini, 2017]. In [Buschmann and Bystrykh, 2013] the authors present recent results on error-correcting DNA codes. A recent application in the area of DNA data storage is described in [Blawat *et al.*, 2016].

For the combinatorial constraints that arise in this type of problem, there are no known efficient algorithms. Techniques from coding theory have been applied [Brenner and Lerner, 1992; Frutos *et al.*, 1997], stochastic local search is applied [Tulpan, 2000; Tulpan and Hoos, 2003; Chee and Ling, 2008], and genetic algorithms are also considered [Garzon *et al.*, 1999; Ashlock *et al.*, 2002; Ashlock and Houghten, 2005; 2009]. In [Montemanni *et al.*, 2014], the authors present three different meta-heuristic approaches for these types of problems. They show improvements on some of the best known codes and present an extensive experimental comparison with previous works. Constraint and SAT programming [van Dongen, 1999; Metodi *et al.*, 2011] have also been applied.

The instance $A_4^{\{\mathrm{P,RC}\}}(8, 4)$ occurs as problem `CSPLib 033: Word Design for DNA Computing on Surfaces` [van Dongen, 1999] and we focus special attention on this instance in the presentation of this paper. For this instance we mention the following results reported in the literature: In [Frutos *et al.*, 1997] (1997) the authors report a solution with **108** DNA words. They search for solutions that have a specific form which they call template—map. In [Tulpan and Hoos, 2003] (2003) the authors present an algorithm based on stochastic local search. Initializing their algorithm with the 108 word set found by Frutos *et al.* [Frutos *et al.*, 1997] they construct a set with **112** words in "less than one day of CPU time". Marc van Dongen also reports a solution with **112** words [van Dongen, 1999]. In [Mancini *et al.*, 2008] (2008) the authors present a solution consisting of **87** DNA words, reported to be found in 554 seconds of CPU time using an OPL [Hentenryck *et al.*, 1999] model. In [Metodi *et al.*, 2011] (2011), the authors apply the BEE constraint solver and report finding a solution of **112** words "in a fraction of a second". They also show that this is the optimal code size for the DNA word design

*Supported by the Israel Science Foundation, grant 182/13.

problem when using the template—map strategy of Frutos *et al.* [Frutos *et al.*, 1997]. In his PhD thesis, Tulpan [Tulpan, 2000] presents a solution with **128** words. Prior to 2016, this solution was not known to the CSPLIB library of constraints. The technique presented in this paper finds a solution with 128 words in under 30 seconds of computation time. This is the first time that a SAT/CSP-based approach is able to match the best known solution for this instance.

In this paper we describe a new constraint model for the design of error-correcting codes for biological applications based on clustering of DNA words. We report solutions which improve the prior state-of-the-art.

2 The Problem Specification

For given parameters (n, d, U) (an instance of the DNA word problem) we seek a set S of cardinality $|S| = m$ consisting of n-letter words over the alphabet $\Sigma = \{A, C, G, T\}$ which satisfies the constraint `(HD)` as well as a subset U of the constraints $\{$ `(RC)`, `(P)` $\}$ detailed below:

The Hamming distance constraint `(HD)`: Each pair of distinct words in S differ in at least d positions.

The reverse complement constraint `(RC)`: For every $w_1, w_2 \in S$ (not necessarily distinct): w_1^R (the reverse of w_1) and w_2^c (the word obtained from w_2 by replacing each A by T, each C by G, and vice versa) differ in at least d positions.

The percentage constraint `(P)`: The number of letters from the set $\{C, G\}$ in each word from S is between $\lfloor n/2 \rfloor$ and $\lceil n/2 \rceil$ (50%).

We denote by $A_4^U(n, d)$ the maximum cardinality of such a set S and we seek lower bounds of $A_4^U(n, d)$ together with corresponding sets S of words which satisfy the constraints.

For a letter $x \in \{A, C, G, T\}$, we denote by x^c the complement letter of x: $A^c = T, T^c = A, C^c = G$, and $G^c = C$. We sometimes express the `(RC)` constraint in terms of two separate cases. ($\texttt{RC}_1$): For every $w \in S$, w^R and w^c differ in at least d positions; and ($\texttt{RC}_2$): Each pair of distinct words $w_1, w_2 \in S$, w_1^R and w_2^c differ in at least d positions. For a word w we write $\texttt{P}(\texttt{w})$ and $\texttt{RC}_1(\texttt{w})$ to specify that w satisfies the constraints `(P)` and ($\texttt{RC}_1$). Similarly, for a pair u, v of words, we write $\texttt{HD}(\texttt{u,v})$ and $\texttt{RC}_2(\texttt{u,v})$.

3 Base Words and their Generated Images

Let $\Sigma = \{\texttt{C}, \texttt{G}, \texttt{T}, \texttt{A}\}$ and denote Σ^n, the set of all n letter words over Σ. A word transformation is a mapping $f\colon \Sigma^n \to \Sigma^n$. We are interested in (small) sets of word transformations which are self-inverse and which commute with respect to composition. A word transformation, f, is self-inverse if $f \circ f = id$. A pair of word transformations, f, g is commutative with respect to composition if $f \circ g = g \circ f$. The composition of word transformations is always associative — a property inherited from the composition of relations. So for word transformations, f, g, h we always have $(f \circ g) \circ h = f \circ (g \circ h)$. We say that a set of word transformations, $\mathcal{T}$, is `SCA` if for all $f, g, h \in \mathcal{T}$, f is self-inversive, f, g are commutative, and f, g, h are associative.

Let $B \subseteq \Sigma^n$ and $\mathcal{T}$ be a set of word transformations. We denote by $\mathcal{T}(B) \subseteq \Sigma^n$ the closure of B under composition of transformations from $\mathcal{T}$. These are all of the words that can be generated from B by repetitive application of transformations from $\mathcal{T}$. Note that if $\mathcal{T}$ is `SCA` then $\mathcal{T}(B)$ consists of (at most) $|B| \times 2^{|\mathcal{T}|}$ words. For simplicity, for $w \in \Sigma^n$, we write $\mathcal{T}(w)$ instead of $\mathcal{T}(\{w\})$. We write $\mathcal{T}'(w)$ to denote $\mathcal{T}(w) \setminus \{w\}$.

In this paper we focus on 4 simple word transformations introduced in the following example. In practice, any set of word transformations defined in terms of a self-inversive permutation of the alphabet Σ and a self-inversive permutation on the letters in a word is `SCA`.

Example 1 Consider the set $\mathcal{T}_4 = \{wc, fc, hs, ms\}$ consisting of four word transformations defined by:

1. wc: (Watson-Crick complement) applies to a word swapping letters $A \leftrightarrow T$, $C \leftrightarrow G$.
2. fc: (full complement) applies to a word swapping letters $A \leftrightarrow G$, $C \leftrightarrow T$.
3. hs: (half swap) applies to a word by swapping the left $\lfloor n/2 \rfloor$ letters with the right $\lfloor n/2 \rfloor$ letters. If n is odd then the middle letter of the word remains in place.
4. ms: (middle swap) applies to a word by swapping letters in positions $i \leq \lfloor n/2 \rfloor$ and $n - i$. If n is odd then the middle letter of the word remains in place.

One can check that (any subset of) $\mathcal{T}_4$ is `SCA`. We denote $\mathcal{T}_3 = \{wc, fc, hs\}$, $\mathcal{T}_2 = \{fc, wc\}$, $\mathcal{T}_1 = \{fc\}$ and $\mathcal{T}_0 = \varnothing$.

Example 2 Consider the context where $n = 8$, the word $w = \texttt{C T A C G A A C}$ and the set of word transformations $\mathcal{T}_3$ introduced in Example 1. The set of eight words generated from w under $\mathcal{T}_3$ is detailed below (to the right of each word the transformation under which it is obtained from w).

CTACGAAC	w	AGGTTCGT	$hs(fc(w))$
GAACCTAC	$hs(w)$	CTTGGATG	$wc(hs(fc(w)))$
TCGTAGGT	$fc(w)$	AGCATCCA	$wc(fc(w))$
GATGCTTG	$wc(w)$	TCCAAGCA	$hs(wc(fc(w)))$

The following two examples illustrate that $\mathcal{T}_3(w)$, for some words w might consist of less than eight words. In Example 3, each (individual) word in $\mathcal{T}_3(w)$ is closed under the half swap transformation. In Example 4, the set of words $\mathcal{T}_3(w)$ is closed under the half swap transformation.

Example 3 Consider the word $w = \texttt{C T T G C T T G}$ and the set of word transformations $\mathcal{T}_3$ introduced in Example 1. The set $\mathcal{T}_3(w)$ consists of the following four words (to the right of each word we detail the transformation under which it is obtained from w). Note that each word $v \in \mathcal{T}_3(w)$ is closed under the half swap transformation. Namely, $hs(v) = v$.

CTTGCTTG	w	GAACGAAC	$wc(w)$
AGGTAGGT	$wc(fc(w))$	TCCATCCA	$fc(w)$

Example 4 Consider the word $w = \texttt{C A C T G T G A}$ and the set of word transformations $\mathcal{T}_3$ introduced in Example 1. The set $\mathcal{T}_3(w)$ consists in the following four words (to the right of each word we detail the transformation under which it is obtained from w). Note that $\mathcal{T}_3(w)$ is closed under the half swap transformation. Namely, $hs(\mathcal{T}_3(w)) = \mathcal{T}_3(w)$.

CACTGTGA	w	GTGACACT	$wc(w)$
TGTCACAG	$wc(fc(w))$	ACAGTGTC	$fc(w)$

In this paper we represent elements of Σ^n as length $2n$ bit vectors. We adopt the convention that each letter of $\Sigma = \{A, C, G, T\}$ is represented as follows by two bits:

$$C = 00 \quad G = 01 \quad \text{(both have msb=0)}$$
$$A = 10 \quad T = 11 \quad \text{(both have msb=1)}$$

Example 5 The eight words detailed in Example 2 are represented respectively by the following eight bit vectors:

```
00 11 10 00 01 10 10 00    10 01 01 11 11 00 01 11
01 10 10 00 00 11 10 00    00 11 11 01 01 10 11 01
11 00 01 11 10 01 01 11    10 01 00 10 11 00 00 10
01 10 11 01 00 11 11 01    11 00 00 10 10 01 00 10
```

For a given word $w \in \Sigma^n$ in bit vector representation, and a set of word transformations $\mathcal{T} \subseteq \mathcal{T}_4$ of Example 1 the set $\mathcal{T}(w)$ can be specified by a corresponding set of literals. Thus, when encoding to CNF a constraint model to represent the set $\mathcal{T}(w)$ no CNF clauses are required — all of the information is in the Boolean variables.

Example 6 Consider $w \in \Sigma^8$ represented by the bit vector $\langle x_1x_2\ x_3x_4\ x_5x_6\ x_7x_8\ x_9x_{10}\ x_{11}x_{12}\ x_{13}x_{14}\ x_{15}x_{16}\rangle$. Then, the following are the bit-vector representations of w, $hs(w)$, $fc(w)$, $wc(w)$, $hs(fc(w))$, $wc(hs(fc(w)))$, $wc(fc(w))$, and $hs(wc(fc(w)))$, respectively:

1. $\langle x_1x_2\ x_3x_4\ x_5x_6\ x_7x_8\ x_9x_{10}\ x_{11}x_{12}\ x_{13}x_{14}\ x_{15}x_{16}\rangle$
2. $\langle x_9x_{10}\ x_{11}x_{12}\ x_{13}x_{14}\ x_{15}x_{16}\ x_1x_2\ x_3x_4\ x_5x_6\ x_7x_8\rangle$
3. $\langle \bar{x}_1\bar{x}_2\ \bar{x}_3\bar{x}_4\ \bar{x}_5\bar{x}_6\ \bar{x}_7\bar{x}_8\ \bar{x}_9\bar{x}_{10}\ \bar{x}_{11}\bar{x}_{12}\ \bar{x}_{13}\bar{x}_{14}\ \bar{x}_{15}\bar{x}_{16}\rangle$
4. $\langle \bar{x}_9\bar{x}_{10}\ \bar{x}_{11}\bar{x}_{12}\ \bar{x}_{13}\bar{x}_{14}\ \bar{x}_{15}\bar{x}_{16}\ \bar{x}_1\bar{x}_2\ \bar{x}_3\bar{x}_4\ \bar{x}_5\bar{x}_6\ \bar{x}_7\bar{x}_8\rangle$
5. $\langle x_1\bar{x}_2\ x_3\bar{x}_4\ x_5\bar{x}_6\ x_7\bar{x}_8\ x_9\bar{x}_{10}\ x_{11}\bar{x}_{12}\ x_{13}\bar{x}_{14}\ x_{15}\bar{x}_{16}\rangle$
6. $\langle x_9\bar{x}_{10}\ x_{11}\bar{x}_{12}\ x_{13}\bar{x}_{14}\ x_{15}\bar{x}_{16}\ x_1\bar{x}_2\ x_3\bar{x}_4\ x_5\bar{x}_6\ x_7\bar{x}_8\rangle$
7. $\langle \bar{x}_1x_2\ \bar{x}_3x_4\ \bar{x}_5x_6\ \bar{x}_7x_8\ \bar{x}_9x_{10}\ \bar{x}_{11}x_{12}\ \bar{x}_{13}x_{14}\ \bar{x}_{15}x_{16}\rangle$
8. $\langle \bar{x}_9x_{10}\ \bar{x}_{11}x_{12}\ \bar{x}_{13}x_{14}\ \bar{x}_{15}x_{16}\ \bar{x}_1x_2\ \bar{x}_3x_4\ \bar{x}_5x_6\ \bar{x}_7x_8\rangle$

The interested reader can double check that for $w =$ C T A C G A A C, represented by 00 11 10 00 01 10 10 00 these eight bit vectors correspond precisely to those given as Examples 2 and 5.

Let $\mathcal{T} \subseteq \mathcal{T}_4$ and $U \subseteq \{$(P), (HD), (RC)$\}$. We say that $w \in \Sigma^n$ is a *base word* with respect to $\mathcal{T}$ and U if the words in $\mathcal{T}(w)$ satisfy the constraints U, and w is the minimal element of $\mathcal{T}(w)$ (under the lexicographic order on the bit vector representation of the words).

Example 7 The word $w_1 =$ C T A C G A A C is a base word with respect to $\mathcal{T}_3$ and $U = \{$(P), (HD), (RC)$\}$. The eight words of $\mathcal{T}_3(w_1)$ detailed in Example 2 satisfy the constraints (P), (HD), and (RC), and the bit vector representation of w_1 is minimal in the lexicographic order of the bit vector representations detailed in Example 5. The word $w_2 =$ C T T G C T T G is also a base word with respect to $\mathcal{T}_3$ and U. The four words of $\mathcal{T}_3(w_2)$ detailed in Example 3 satisfy the constraints, and w_2 is minimal (in the lexicographic order of the respective bit vector representations).

Our strategy to solve the DNA word design problem is to seek a solution S constructed in terms of a much smaller set, B, of base words such that $S = \mathcal{T}(B)$ for a set of word transformations $\mathcal{T}$. In this sense, one can view $\mathcal{T}$ as inducing a clustering of the words in Σ^n. However, it is not just about the size of the set B. Expressing the constraints of the DNA word design problem in terms of the set B turns out to be much more compact as well. We demonstrate this in the next three lemmas where we focus on the case where $\mathcal{T} \subseteq \mathcal{T}_4$.

Let $x_1 \dots x_{2n}$ be a bit vector representing a word $w \in \Sigma^n$ and $B = \mathcal{T}(w)$ represented as a set of bit vectors. A model that constrains w to be a base word must constrain w to be the lexicographic minimum in $\mathcal{T}(w)$. That means imposing at most 16 constraints (assuming that $\mathcal{T} \subseteq \mathcal{T}_4$). A model that constrains w to be a base word must impose constraints (P) and ($\mathtt{RC}_1$) on each word of $\mathcal{T}(w)$. The following lemma implies that it suffices to impose them only on w.

Lemma 1 *Let $w \in \Sigma^n$. If w satisfies Constraint (P), then so does $\mathcal{T}(w)$. If w satisfies Constraint ($\mathtt{RC}_1$), then so does $\mathcal{T}(w)$.*

Proof: Let $w \in \Sigma^n$ and $f \in \{hs, wc, fc, ms\}$. We show that if w satisfies the constraint, then so does $f(w)$. Denote $w = \langle x_1, \dots, x_n\rangle$ and $f(w) = \langle y_1, \dots, y_n\rangle$ and assume the constraints hold for w. The proof for constraint (P): If $f \in \{hs, wc, ms\}$ then w and $f(w)$ have the same number of letters from $\{C, G\}$. If $f = fc$ and n is even, then the number of $\{C, G\}$ in $f(w)$ equals the number of $\{A, T\}$ in w, which by constraint (P), equals the number of $\{C, G\}$ in w. For odd n the proof follows with a similar argument. The proof for constraint ($\mathtt{RC}_1$): consider the Boolean values $b_i \leftrightarrow x_i \neq x^c_{n-i+1}$ and $d_i \leftrightarrow y_i \neq y^c_{n-i+1}$ for $1 \leq i \leq n$. For each choice of f, direct inspection reveals that $\sum_{i=1}^n b_i = \sum_{i=1}^n d_i$. So ($\mathtt{RC}_1$) holds for $f(w)$. □

A model that constrains w to be a base word must impose constraints (HD) and ($\mathtt{RC}_2$) on each pair of distinct words from $\mathcal{T}(w)$ i.e., two constraints for each pair of distinct words from $\mathcal{T}(w)$. In the case that $\mathcal{T}(w)$ contains k words that means at most $\binom{k}{2}$ constraints in total. The following lemma implies that it suffices to consider only the pairs including the base word w i.e., at most $k - 1$ constraints only.

Lemma 2 *Let $w \in \Sigma^n$. If all pairs (u, w), where $u \in \mathcal{T}(w)$ and $u \neq w$ satisfy Constraints (HD) and ($\mathtt{RC}_2$), then $\mathcal{T}(w)$ satisfies Constraints (HD) and ($\mathtt{RC}_2$).*

Proof: We detail the proof for constraint (HD). The proof for constraint ($\mathtt{RC}_2$) is similar. Assume that w is a base word and that w differs from every other word $w' \in \mathcal{T}_4(w)$ in at least $n/2$ positions. We argue $(*)$ that for any pair of distinct words $u, v \in \mathcal{T}_4(w)$, also u and v differ in at least 4 positions. By definition of $\mathcal{T}_4(w)$, $u = f(w)$ where f is some composition of functions from $\{fc, wc, hs, ms\}$. The pair u, v differ in the same number of positions as the pair $f(u), f(v)$. The argument $(*)$ follows because the functions composing to f are all inversive and so $f(u) = w$. But we already know that w and $f(v)$ differ in at least four positions. □

Now consider a set B of base words. A model that constrains $S = \mathcal{T}(B)$ to be a solution of the DNA word design problem must impose constraints (HD) and ($\mathtt{RC}_2$) on pairs of words: for each $w_1, w_2 \in B$ (viewing B as a sequence we can assume that w_1 occurs before w_2) all pairs u, v where

$u \in \mathcal{T}(w_1)$ and $v \in \mathcal{T}(w_2)$. The following lemma implies that it suffices, for the pair (w_1, w_2) to impose the constraints for pairs of the form (w_1, v) with $v \in \mathcal{T}(w_2)$ and ignoring the symmetric ones of the form (u, w_2). That means checking a linear number of constraints, two for each pair (u, w_2), instead of a quadratic number.

Lemma 3 *Let $w_1, w_2 \in \Sigma^n$. If all pairs (w_1, v) with $v \in \mathcal{T}(w_2)$ satisfy Constraints (HD) and ($\mathtt{RC}_2$), then all pairs u, v in $\mathcal{T}(w_1) \times \mathcal{T}(w_2)$ satisfy Constraints (HD) and ($\mathtt{RC}_2$).*

Proof: The proof is similar to that of Lemma 2. □

4 The Constraint Model

Let (n, d, U) specify an instance of the DNA word problem. To ease presentation, we introduce the model in three steps.

Step 1: Consider the search for a solution S given in terms of m base words with respect to a single set of word transformations $\mathcal{F}_1 \subseteq \mathcal{T}_4$. Let M be a $m \times 2n$ Boolean matrix. Each row in M represents an n-letter base word (in $2n$ bits) and represents $2^{|\mathcal{F}_1|}$ different words. We view a matrix as the set of its rows. We write $w \in M$ to indicate that w is a row of M. If $w, w' \in M$ we write $w <_M w'$ to indicate that row w occurs before row w' in M. We denote by $\prec$ the lexicographic order on Boolean vectors. The solution S represented by matrix M is $S = \cup\{\, \mathcal{F}_1(w) \mid\ w \in M \ \}$.

For each base word $w \in M$, each single word u in $\mathcal{F}_1(w)$ is required to satisfy constraints $\mathtt{P}(\mathtt{u})$ and $\mathtt{RC}_1(\mathtt{u})$. Moreover, each pair of distinct words u and v in $\mathcal{F}_1(w)$ is required to satisfy constraints $\mathtt{HD}(\mathtt{u}, \mathtt{v})$ and $\mathtt{RC}_2(\mathtt{u}, \mathtt{v})$. Here, according to Lemma 1 and 2, it is sufficient to require $\mathtt{P}$ and $\mathtt{RC}_1$ only for the base word w, and it is sufficient to require $\mathtt{HD}$ and $\mathtt{RC}_2$ only for pairs that include w.

$$\bigwedge_{w \in M} \left(\mathtt{P}(\mathtt{w}) \wedge \mathtt{RC}_1(\mathtt{w}) \wedge \bigwedge_{v \in \mathcal{F}_1(\mathtt{w})} (\mathtt{HD}(\mathtt{w}, \mathtt{v}) \wedge \mathtt{RC}_2(\mathtt{w}, \mathtt{v})) \right) \quad (1)$$

Observe that Constraint (1) implies that for each base word w, a row in M, the words in $\mathcal{F}_1(w)$ are pairwise distinct.

The next constraint ensures that words u, v generated from different base words $w, w' \in M$ satisfy constraints $\mathtt{HD}(\mathtt{u}, \mathtt{v})$ and $\mathtt{RC}_2(\mathtt{u}, \mathtt{v})$. Here, according to Lemma 3, it is sufficient to consider pairs which include w.

$$\bigwedge_{\substack{w, w' \in M \\ w <_M w'}} \bigwedge_{v \in \mathcal{F}_1(w')} (\mathtt{HD}(\mathtt{w}, \mathtt{v}) \wedge \mathtt{RC}_2(\mathtt{w}, \mathtt{v})) \quad (2)$$

The next constraint in our model is a symmetry breaking constraint. It ensures that words in M are minimal in the sets they generate under $\mathcal{F}_1$, and that the matrix of base words is sorted.

$$\bigwedge_{w \in M} \bigwedge_{v \in \mathcal{F}_1(\mathtt{w})} w \prec v \ \wedge \ \bigwedge_{i=1}^{m-1} \mathtt{M_i} \prec \mathtt{M_{i+1}} \quad (3)$$

Example 8 Consider the instance $(12, 8, \varnothing)$, in which we seek a solution S that contains 12-letter words over the alphabet $\{A, C, G, T\}$ such that every pair $w_1, w_2 \in S$ has Hamming distance at least 8. Consider a clustering based on $\mathcal{T}_4$ as in Example 1 where each cluster consists of 16 DNA words. The following 7 base words (translated from the corresponding 5×24 Boolean matrix) provide a solution with $7 \times 16 = 112$ DNA code words, improving the previous best known result of 64 due to [Bogdanova *et al.*, 2001].

CCCGTCGTATCG
CCGCCTTAGCAC
CCGAACATGAGA
CCAGGAACTCGC

CCTCCGAGCTGT
CCTTATCAAACT
CCTTTATCGGTG

Step 2: Now, consider an instance (n, d, U) and a pair of sets of word transformations, $\mathcal{F}_1, \mathcal{F}_2 \subseteq \mathcal{T}_4$. We seek a solution S represented by m_1 base words with respect to $\mathcal{F}_1$ and m_2 base words with respect to $\mathcal{F}_2$. We model the solution using two matrices: M_1 with m_1 rows, and M_2 with m_2 rows. Both with $2n$ columns to represent n-letter words.

The constraint model includes Constraints (1–3) on each of the two matrices, and additional constraints on pairs of words $u \in \mathcal{F}_1(w_1)$ and $v \in \mathcal{F}_2(w_2)$ where $w_1 \in M_1$ and $w_2 \in M_2$ satisfy constraints $\mathtt{HD}(\mathtt{u}, \mathtt{v})$ and $\mathtt{RC}_2(\mathtt{u}, \mathtt{v})$.

$$\bigwedge_{\substack{w_1 \in M_1 \\ w_2 \in M_2}} \bigwedge_{\substack{u \in \mathcal{F}_1(w_1) \\ v \in \mathcal{F}_2(w_2)}} (\mathtt{HD}(\mathtt{u}, \mathtt{v}) \wedge \mathtt{RC}_2(\mathtt{u}, \mathtt{v})) \quad (4)$$

In some cases we can do better as indicated by the following generalization of Lemma 3.

Lemma 4 *Let $\mathcal{T}' \subseteq \mathcal{T}$ be sets of word transformations and $w_1, w_2 \in \Sigma^n$. If all pairs (w_1, v) with $v \in \mathcal{T}(w_2)$ satisfy Constraints (HD) and ($\mathtt{RC}_2$), then all pairs u, v in $\mathcal{T}'(w_1) \times \mathcal{T}(w_2)$ satisfy Constraints (HD) and ($\mathtt{RC}_2$).*

So, if $\mathcal{F}_1 \subseteq \mathcal{F}_2$, we can introduce a refined constraint to the model:

$$\bigwedge_{\substack{w_1 \in M_1 \\ w_2 \in M_2}} \bigwedge_{v \in \mathcal{F}_2(w_2)} (\mathtt{HD}(\mathtt{w_1}, \mathtt{v}) \wedge \mathtt{RC}_2(\mathtt{w_1}, \mathtt{v})) \quad (4')$$

Example 9 Consider the setting of Example 8 and a clustering based on $\mathcal{T}_4$ and $\mathcal{T}_3$ as in Example 1 where each cluster consists of 16 or 8 DNA words respectively. The following base words in M_1 and M_2 (translated from the corresponding 5×24 and 2×24 Boolean matrices) provide a solution with $7 \times 16 + 2 \times 8 = 128$ DNA code words, doubling the previous best known result of 64 due to [Bogdanova *et al.*, 2001].

CCCGACCTATGC
CCCTCGTCAGCA
CCGTGCGAGTTG
CCAAGTATAAAG
CCATTACATAGA
CCTCGATGTCCG
CCTACTCGCGGT

$\} M_1$

CCGGAGACCCAT
CGTTTGAGAATT

$\} M_2$

Step 3: The generalization to a solution based on a finite number of sets, $\mathcal{F}_1, \ldots, \mathcal{F}_k$ of word transformations follows the same idea as in step 2.

n	d	U	$lit.$	new	$clustering$	$\#clauses$	$\#vars$	$time\ (sec.)$
7	3	{hd,rc,p}	135_1	147	$(\mathcal{T}_0,1),(\mathcal{T}_1,3),(\mathcal{T}_2,9),(\mathcal{T}_3,13)$	529,280	122,711	49,852
8	4	{hd,rc,p}	128_4	128	$(\mathcal{T}_2,8),(\mathcal{T}_3,12)$	394,474	90,398	21
9	5	{hd,rc,p}	67_1	76	$(\mathcal{T}_1,2),(\mathcal{T}_2,4),(\mathcal{T}_3,7)$	196,265	44,576	120,728
10	6	{hd,rc,p}	54_1	63	$(\mathcal{T}_0,1),(\mathcal{T}_1,1),(\mathcal{T}_2,4),(\mathcal{T}_3,7)$	146,270	32,819	512
11	7	{hd,rc,p}	37_2	56	$(\mathcal{T}_2,2),(\mathcal{T}_3,6)$	108,829	24,209	23
12	8	{hd,rc,p}	29_2	39	$(\mathcal{T}_0,7),(\mathcal{T}_2,8)$	180,545	39,763	1,279
14	10	{hd,rc,p}	20_2	23	$(\mathcal{T}_0,7),(\mathcal{T}_3,2)$	94,640	20,415	263
15	10	{hd,rc,p}	37_2	39	$(\mathcal{T}_0,1),(\mathcal{T}_1,1),(\mathcal{T}_2,1),(\mathcal{T}_3,4)$	119,611	25,574	136,072
9	5	{hd,p}	134_2	140	$(\mathcal{T}_2,3),(\mathcal{T}_4,8)$	335,895	75,969	339
12	8	{hd}	64_3	128	$(\mathcal{T}_3,4),(\mathcal{T}_4,6)$	184,769	40,501	878

1. [Chee and Ling, 2008] 2. [Montemanni *et al.*, 2014] 3. [Bogdanova *et al.*, 2001] 4. [Tulpan, 2000]
$\mathcal{T}_0=\varnothing$ $\mathcal{T}_1=\{fc\}$ $\mathcal{T}_2=\{wc,fc\}$ $\mathcal{T}_3=\{wc,fc,hs\}$ $\mathcal{T}_4=\{wc,fc,hs,ms\}$

Table 1: Computing lower bounds of $A_4^U(n,d)$.

5 Implementing the Constraint Model

The computations reported in this paper are performed using the finite-domain constraint compiler BEE [Metodi *et al.*, 2013] which compiles constraints to a CNF, and solves it applying an underlying SAT solver. In the configuration for this paper we use CryptoMiniSAT v2.5.1 as the underlying SAT solver. All computations were performed on a cluster of Intel E8400 cores, each clocked at 2 GHz. Each of the cores in the cluster has computational power comparable to a core on a standard desktop computer. Each SAT instance is run on a single thread.

A solution is a tuple of matrices $M_1,\ldots,M_k$ as described in step 3 of Section 4, representing $m_1,\ldots,m_k$ base words respectively. These are then applied to generate a set S of corresponding DNA words which are a solution to the DNA word design problem.

Several optimizations are applied in the implementation. Constraint (3), introduced above, is a first symmetry break. Base words are selected to be the minimal words in the corresponding clusters of words generated by $\mathcal{F}_1,\ldots,\mathcal{F}_k$, and the rows of the matrices $M_1,\ldots,M_k$ are constrained to be sorted. Columns in the matrices cannot be arbitrarily reordered. However there is a symmetry that allows to swap the letters at positions i and $n+1-i$ in the n-letter base words and we have introduced this symmetry break.

For experimentation, an instance takes the form (n,d,U) together with a tuple of pairs $\langle(m_1,\mathcal{F}_1),\ldots,(m_k,\mathcal{F}_k)\rangle$ where $\mathcal{F}_1,\ldots,\mathcal{F}_k\subseteq\mathcal{T}_4$ are sets of word transformations and we seek a solution with m_1 base words with respect to $\mathcal{F}_1$ and m_2 base words with respect to $\mathcal{F}_2$ etc.

Table 1 summarizes our experimentation and details the best lower bounds for $A_4^U(n,d)$ that we have found using our approach. The first three columns detail the instance (n,d,U). The forth column ("lit") details the best previous result that we found in the literature together with a reference indicating where this result can be found in literature as detailed in the legend of Table 1. The fifth column ("new") details our result. The sixth column ("clustering") details the word transformations that were used and the number of base words per set of transformations (e.g, $\langle(\mathcal{T}_4,4),(\mathcal{T}_0,3)\rangle$ means that 4 base words were found for the word transformations in $\mathcal{T}_4$ and 3 base words were found for the word transformations in $\mathcal{T}_0$, summing up to a total of $4\times16+3\times1=67$ DNA code words). The last three columns describe the CNF encoding (number of clauses and variables) and the SAT solving time required to find the corresponding DNA code.

$\mathcal{T}_3$	$\mathcal{T}_2$	$\mathcal{T}_1$	$\mathcal{T}_0$	total	$time$
13	0	0	0	104	1
14	0	0	0	112	21
15	0	0	0	120	t.o.
14	6	0	0	136	255
13	9	0	0	140	3,229
14	7	0	0	140	651
14	7	2	0	144	712
13	9	2	1	145	1,057
14	7	2	2	146	1,697
13	9	2	2	146	3,423
13	9	3	1	147	49,852

Table 2: Some results in the search for a solution for instance $(7,3,\{hd,rc,p\})$. Time is in seconds (24 hr. timeout).

In Table 1 results are for a given type of clustering. We have not addressed in this paper the issue of how to find a suitable type of clustering. For each instance of the DNA word problem we have experimented with a wide range of possible choices applying a mixture of intuition and brute force search. We only present here the instances for which we have improved on the best previous result, with the exception of instance $(8,4,\{hd,rc,p\})$ for which we obtained a solution with 128 words, equal in size to the result by Tulpan [Tulpan, 2000]. In this case, we report the result as it is obtained in less than 30 seconds of computation, as opposed to the result by Tulpan which takes orders of magnitude longer.

Tables 2, and 3 illustrate some of the results in the search process for a solution for the instances $(7,3,\{hd,rc,p\})$ and $(9,5,\{hd,rc,p\})$. For these instances, there is no solution that contains a cluster of type $\mathcal{T}_4$. The first four columns detail the number of clusters of type $\mathcal{T}_3$, $\mathcal{T}_2$, $\mathcal{T}_1$ and $\mathcal{T}_0$. The fifth column details the total number of words represented by the corresponding clustering. The final column indicates the time (in seconds) to find a solution with the corresponding

$\mathcal{T}_3$	$\mathcal{T}_2$	$\mathcal{T}_1$	$\mathcal{T}_0$	total	$time$
8	0	0	0	64	47
7	3	0	0	68	238
8	1	1	0	70	452
8	1	2	0	72	1,790
7	4	0	0	72	2,833
9	0	0	0	72	t.o.
7	4	1	0	74	7,451
7	4	1	1	75	75,817

Table 3: Some results in the search for a solution for instance $(9, 5, \{hd, rc, p\})$. Time is in seconds (24 hr. timeout).

$\mathcal{T}_4$	$\mathcal{T}_3$	$\mathcal{T}_2$	$\mathcal{T}_1$	$\mathcal{T}_0$	total	$time$
6	0	0	0	0	96	180
7	0	0	0	0	112	142
6	3	0	0	0	120	91
7	1	0	0	0	120	89
6	4	0	0	0	128	878
7	2	0	0	0	128	1,182
8	0	0	0	0	128	t.o.

Table 4: Some results in the search for a solution for instance $(12, 8, \{hd\})$. Time is in seconds (24 hr. timeout).

clustering. Here we assumed a timeout of 24 hours. Table 4 illustrates the search for the instance $(12, 8, \{hd\})$. Here solutions include also clusters of type $\mathcal{T}_4$.

6 Bottom Line and Discussion

We have improved on several lower bounds $A_4^U(n, d)$ in the context of error-correcting codes for DNA word problems. Our approach is constraint based and expressed in terms of a partition of the class of all n-letter words into clusters of words as determined by subsets of a set $\mathcal{T}$ of word transformations. Each cluster is required to satisfy the constraints of the word problem. A pair of clusters that contain conflicting words cannot both be in a solution.

To evaluate the quality of our proposed clustering of DNA words let us consider the instance $(8, 4, \{\texttt{(P)}, \texttt{(HD)}, \texttt{(RC)}\})$ with a clustering based on $\mathcal{T}_3$. There are $4^8 = 65{,}536$ DNA words (of length eight). From these, there are only 17,440 words which satisfy constraints (P) and $(\texttt{RC}_1)$ and hence can appear in a solution to the DNA word design problem. These 17,440 words partition into a total of 2262 clusters when applying $\mathcal{T}_3$. From these 2262 clusters only 1544 satisfy constraints (HD) and $(\texttt{RC}_2)$. From these 1544, 1424 are clusters of size eight, and 120 are of size four. The 1544 clusters that could appear in a solution comprise 11,872 DNA words. This is 68% of the 17,440 that could appear in a solution. We also note that there are $2262 - 1544 = 718$ clusters that satisfy constraints (P) and $(\texttt{RC}_1)$ but does not satisfy constraints (HD) or $(\texttt{RC}_2)$. We observe that these "bad" clusters, are not so due to a single contradicting pair of words: by applying Lemma 2 it happens that *every* word in a "bad" cluster is in conflict with at least one other word in that cluster. In clusters of size four there is an average of 2 contradicting pairs. In clusters of size eight there is an average of 6.06 contradicting pairs. Furthermore, when a pair of "good" clusters (from the 1544) does not satisfy either (HD) or (RC), then this is on average because of 8.68 contradicting pairs. Indeed, by applying Lemma 3 it is obtained that in this case *every* word from one cluster is in conflict with at least one word from the other cluster.

As can be seen in Table 1, most of the new codes obtained applying our technique are found in under 10 minutes of computation time. Several of the codes require considerable more time. All of the codes are available on request from the authors and will be made publicly available,

Finally, we note that our approach is incomplete. There may be (larger) solutions which cannot be found using our approach as they cannot be expressed in terms of the specific clusters we have selected.

References

[Adleman, 1994] Leonard M. Adleman. Molecular computation of solutions to combinatorial problems. *Science*, 266(5187):1021–1024, 1994.

[Ashlock and Houghten, 2005] Daniel A. Ashlock and Sheridan K. Houghten. A novel variation operator for more rapid evolution of DNA error correcting codes. In *Proceedings of the 2005 Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, pages 53–60. IEEE, 2005.

[Ashlock and Houghten, 2009] Daniel A. Ashlock and Sheridan K. Houghten. DNA error correcting codes: No crossover. In *Proceedings of the 2009 Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, pages 38–45. IEEE, 2009.

[Ashlock *et al.*, 2002] Daniel Ashlock, Ling Guo, and Fang Qiu. Greedy closure evolutionary algorithms. In *Proceedings of the Evolutionary Computation*, CEC '02, pages 1296–1301. IEEE Computer Society, 2002.

[Blawat *et al.*, 2016] Meinolf Blawat, Klaus Gaedke, Ingo Htter, Xiao-Ming Chen, Brian Turczyk, Samuel Inverso, Benjamin W. Pruitt, and George M. Church. Forward error correction for DNA data storage. *Procedia Computer Science*, 80:1011 – 1022, 2016.

[Bogdanova *et al.*, 2001] Galina T. Bogdanova, Andries E. Brouwer, Stoian N. Kapralov, and Patric R. J. Östergård. Error-correcting codes over an alphabet of four elements. *Designs, Codes and Cryptography*, 23(3):333–342, 2001.

[Brenner and Lerner, 1992] Sydney Brenner and Richard A. Lerner. Encoded combinatorial chemistry. *Proceedings of the National Academy of Sciences of the United States of America*, 89(12):5381–5383, 1992.

[Buschmann and Bystrykh, 2013] Tilo Buschmann and Leonid V. Bystrykh. Levenshtein error-correcting barcodes for multiplexed DNA sequencing. *BMC Bioinformatics*, 14(1):272, 2013.

[Chee and Ling, 2008] Yeow Meng Chee and San Ling. Improved lower bounds for constant GC-content DNA codes. *IEEE Trans. Information Theory*, 54(1):391–394, 2008.

[Fodor *et al.*, 1991] Stephen P.A. Fodor, Leighton J. Read, Michael C. Pirrung, Lubert Stryer, Amy Tsai Lu, and Dennis Solas. Light-directed, spatially addressable parallel chemical synthesis. *Science*, 251(4995):767–773, 1991.

[Frutos *et al.*, 1997] Anthony G. Frutos, Qinghua Liu, Andrew J. Thiel, Anne Marie W. Sanner, Anne E. Condon, Lloyd M. Smith, and Robert M. Corn. Demonstration of a word design strategy for DNA computing on surfaces. *Journal of Nucleic Acids Research*, 25(23):4748–4757, 1997.

[Garzon *et al.*, 1999] Max H. Garzon, Russell J. Deaton, and John A. Rose. Soft molecular computing. In Erik Winfree and David K. Gifford, editors, *DNA Based Computers, Proceedings of a DIMACS Workshop*, volume 54 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 91–100. DIMACS/AMS, 1999.

[Goodnow Jr *et al.*, 2017] Robert A. Goodnow Jr, Christoph E. Dumelin, and Anthony D. Keefe. DNA-encoded chemistry: enabling the deeper sampling of chemical space. *Nat Rev Drug Discov*, 16(2):131–147, Feb 2017. Review.

[Hentenryck *et al.*, 1999] Pascal Van Hentenryck, Laurent Michel, Laurent Perron, and Jean-Charles Régin. Constraint programming in OPL. In Gopalan Nadathur, editor, *Principles and Practice of Declarative Programming, International Conference Proceedings*, volume 1702 of *LNCS*, pages 98–116. Springer, 1999.

[Lipton, 1995] RJ Lipton. Dna solution of hard computational problems. *Science*, 268(5210):542–545, 1995.

[MacWilliams and Sloane, 1977] Florence Jessie MacWilliams and N. J. A. Neil James Alexander Sloane. *The theory of error correcting codes*. North-Holland mathematical library. North-Holland Pub. Co. New York, Amsterdam, New York, 1977.

[Mancini *et al.*, 2008] Toni Mancini, Davide Micaletto, Fabio Patrizi, and Marco Cadoli. Evaluating ASP and commercial solvers on the CSPLib. *Constraints*, 13(4):407–436, 2008.

[Metodi *et al.*, 2011] Amit Metodi, Michael Codish, Vitaly Lagoon, and Peter J. Stuckey. Boolean equi-propagation for optimized SAT encoding. In Jimmy Ho-Man Lee, editor, *Principles and Practice of Constraint Programming - 17th International Conference Proceedings*, volume 6876 of *LNCS*, pages 621–636. Springer, 2011.

[Metodi *et al.*, 2013] Amit Metodi, Michael Codish, and Peter J. Stuckey. Boolean equi-propagation for concise and efficient SAT encodings of combinatorial problems. *J. Artif. Intell. Res. (JAIR)*, 46:303–341, 2013.

[Montemanni *et al.*, 2014] Roberto Montemanni, Derik H. Smith, and N. Koul. Three metaheuristics for the construction of constant GC-content dna codes. In *Proceedings of the 6th International Conference on Applied Operational Research*, volume 6, pages 167–175, 2014.

[Mullard, 2016] Asher Mullard. DNA tags help the hunt for drugs. *Nature*, 530(7590):367–369, Feb 2016. Technology Feature.

[Qiu *et al.*, 2003] F. Qiu, L. Guo, T.-J. Wen, F. Liu, D Ashlock, and P. S. Schnable. Dna sequence-based bar codes for tracking the origins of expressed sequence tags from a maize cdna library constructed using multiple mrna sources. *Plant Physiol.*, 2003.

[Schena *et al.*, 1995] Mark Schena, Dari Shalon, Ronald W. Davis, and Patrick O. Brown. Quantitative monitoring of gene expression patterns with a complementary dna microarray. *Science*, 270(5235):467–470, 1995.

[Shoemaker *et al.*, 1996] Daniel D. Shoemaker, Deval A. Lashkari, Don Morris, and Ronald W. Davis. Quantitative phenotypic analysis of yeast deletion mutants using a highly parallel molecular bar-coding strategy. *Nat Genet*, 1996.

[Tulpan and Hoos, 2003] Dan C. Tulpan and Holger H. Hoos. Hybrid randomised neighbourhoods improve stochastic local search for DNA code design. In Yang Xiang and Brahim Chaib-draa, editors, *Advances in Artificial Intelligence, 16th Conference of the Canadian Society for Computational Studies of Intelligence, AI 2003, Halifax, Canada, June 11-13, 2003, Proceedings*, volume 2671 of *Lecture Notes in Computer Science*, pages 418–433. Springer, 2003.

[Tulpan, 2000] Dan Tulpan. *Effective Heuristic Methods for DNA Strand Design (PhD Thesis)*. PhD thesis, Politehnica University of Bucharest, 2000.

[van Dongen, 1999] Marc van Dongen. CSPLib problem 033: Word design for dna computing on surfaces. `http://www.csplib.org/Problems/prob033`, 1999.

[Yuen and Franzini, 2017] Lik Hang Yuen and Raphael M. Franzini. Achievements, challenges, and opportunities in dna-encoded library research: An academic point of view. *ChemBioChem*, pages n/a–n/a, 2017.