ELSEVIER
Available online at www.sciencedirect.com
SCIENCE DIRECT®
Theoretical Computer Science 334 (2005) 99–113
Theoretical Computer Science
www.elsevier.com/locate/tcs

Linear constructions for DNA codes

Philippe Gaborit[a],*, Oliver D. King[b]

[a] *LACO, Université de Limoges, 123, av. A. Thomas, 87000 Limoges, France*
[b] *Department of Biological Chemistry and Molecular Pharmacology, Harvard Medical School, MA, USA*

Received 25 February 2004; received in revised form 27 October 2004; accepted 4 November 2004

Communicated by A. Condon

Abstract

In this paper we translate in terms of coding theory constraints that are used in designing DNA codes for use in DNA computing or as bar-codes in chemical libraries. We propose new constructions for DNA codes satisfying either a reverse-complement constraint, a GC-content constraint, or both, that are derived from additive and linear codes over four-letter alphabets. We focus in particular on codes over $GF(4)$, and we construct new DNA codes that are in many cases better (sometimes far better) than previously known codes. We provide updated tables up to length 20 that include these codes as well as new codes constructed using a combination of lexicographic techniques and stochastic search.
© 2004 Elsevier B.V. All rights reserved.

Keywords: DNA computing; Coding theory; DNA codes

1. Introduction

The problem of designing DNA codes (sets of words of fixed length n over the alphabets $\{A, C, G, T\}$) that satisfy certain combinatorial constraints has applications for reliably storing and retrieving information in synthetic DNA strands. These codes can be used in particular for DNA computing [1] or as molecular bar-codes [8,25].

In [13,17,20,22], four different constraints on DNA codes are considered: the Hamming constraint for a distance d, the reverse-complement constraint, the reverse constraint and

* Corresponding author. Fax: +33 5 55 457322.
E-mail addresses: gaborit@unilim.fr (P. Gaborit), oliver_king@hms.harvard.edu (O.D. King).

0304-3975/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2004.11.004

the fixed *GC*-content constraint. The purpose of the first three constraints is to make non-desirable hybridizations between different DNA strands less likely to happen. The fixed *GC*-content constraint is used to obtain similar melting temperatures [13].

Bounds for codes satisfying these constraints are presented in [17,22] and several constructions using stochastic search, a template-map strategy, genetic algorithms and lexicographic codes have been proposed [11,13,17,20,26,27]. As discussed in Section 4, there are parameter ranges for which these approaches are not entirely satisfactory.

In this paper we translate all of these constraints in terms of coding theory [21]. This point of view allows us to use classical coding results and leads to the construction of new good DNA codes that are in almost all cases better than previously known constructions for length n greater than 10 and minimum distance d up to roughly $n/2$. Moreover, these constructions are easily obtained and are scalable. Note that previous results were obtained in [23] with codes but only for reversible cyclic codes and asymptotic lengths.

It should be noted that the constraints we consider do not address certain issues related to hybridization which may be important in practical applications, for example insensitivity to frame-shifts, the avoidance of secondary structure, and the use of a more accurate model of melting temperature, see [7] for a survey of approaches to DNA word-design that address these and other issues. Some of these issues may be addressed computationally as a post-processing step (cf. Section 4); codes may also be screened experimentally [4]. In applications in which codewords (possibly of variable length) concatenate, additional constraints become important, some of which can be cast in coding-theoretic terms (e.g. [2,18]) and others which have been investigated from the perspective of formal languages (e.g. [16]) and symbolic dynamics (e.g. [10]).

The paper is organized as follows: in Section 2 we recall basic notions for DNA codes and linear codes; in Section 3 we translate the constraints on DNA codes into coding-theoretic terms; in Section 4 we detail our constructions; and in Section 5 we give tables of the best known DNA codes of length 20 or less satisfying the fixed *GC*-content constraint or the fixed *GC*-content constraint together with the *RC* constraint. These tables include linear constructions and new codes obtained through a combination of lexicographic constructions and stochastic search.

2. Background on linear codes and DNA codes

A DNA code of length n is a set of codewords $(x_1, \ldots, x_n)$ with $x_i \in \{A, C, G, T\}$ (representing the four nucleotides in DNA). We use a hat to denote the Watson–Crick complement of a nucleotide, so $\hat{A} = T$, $\hat{T} = A$, $\hat{C} = G$, and $\hat{G} = C$.

The *Hamming distance* $H(x, y)$ between two codewords is the number of coordinates in which x and y are distinct. The *reverse* of a codeword $x = (x_1, \ldots, x_n)$ is denoted by $x^{\mathrm{R}} = (x_n, \ldots, x_1)$, and the *reverse-complement* of $x = (x_1, \ldots, x_n)$ is denoted by $x^{\mathrm{RC}} = (\hat{x}_n, \ldots, \hat{x}_1)$.

In this paper we shall identify codes over $\{A, C, G, T\}$ with codes over other four-letter alphabets K, where K is either $GF(4) = \{0, \omega, \bar{\omega}, 1\}$ or $Z_4 = Z/4Z = \{0, 1, 3, 2\}$. The four symbols in $\{A, C, G, T\}$ are identified with the four symbols in K in the orders given above, so that $\hat{x} = x + 1$ for $x \in GF(4)$ and $\hat{x} = x + 2$ for $x \in Z_4$.

We refer to [15,21] for a general background on error-correcting codes. An additive code C over K of length n is an additive subgroup of K^n. If, moreover, C is a linear subspace of K^n the code is said to be linear. A $[n, k, d]$ code denotes a code of length n, dimension k and minimum distance d over a given field. Note that in this paper we mainly consider linear codes over $GF(4)$, although some theorems are valid over different alphabets. We denote the complete weight enumerator of a code $\mathcal{C}$ over $GF(4)$ by $CWE_{\mathcal{C}}(x, y, z, t) = \sum_{c\in\mathcal{C}} x^{n_0(c)} y^{n_1(c)} z^{n_\omega(c)} t^{n_{\bar{\omega}}(c)}$, where $n_k(c)$ is the number of occurrences of $k \in GF(4)$ in a codeword c. We denote by $GCW_{\mathcal{C}}(x, y) = CWE_{\mathcal{C}}(x, x, y, y)$ the GC-weight enumerator of $\mathcal{C}$, i.e., the weight enumerator that counts the number of coordinates in $\{0, 1\}$ and $\{\omega, \bar{\omega}\}$, and we denote by $BW_{\mathcal{C}}(x, y)$ the weight enumerator of the binary subcode of $\mathcal{C}$ (i.e., the subcode of $\mathcal{C}$ consisting of those words of $\mathcal{C}$ whose coordinates are either '0' or '1').

The dual of a code $\mathcal{C}$ of length n over K is defined as $\mathcal{C}^{\perp} = \{x \in K^n | x \cdot y = 0 \text{ for all } y \in \mathcal{C}\}$, where $x \cdot y$ is the standard inner product $x_1 y_1 + \cdots x_n y_n$ for $K = Z_4$ and the Hermitian inner product $x_1 \bar{y_1} + \cdots x_n \bar{y_n}$ for $K = GF(4)$.

Two codes over K are permutation-equivalent if one can be obtained from the other by permuting the columns (coordinates), and are equivalent if one can be obtained from the other by permuting the columns and multiplying columns by invertible elements of K.

The permutation group of a code of length n is the group of permutations of $\{1, 2, 3, \ldots, n\}$ that, when applied to the columns of the code, maps the code to itself. A permutation that is its own inverse is called an involution.

3. Constraints on DNA codes

3.1. Hamming distance constraint

The Hamming distance constraint for a DNA code $\mathcal{C}$ is that $H(x, y) \geqslant d$ for all $x, y \in \mathcal{C}$ with $x \neq y$, for some prescribed minimum distance d. This constraint will be enforced in all of the codes we consider, in addition to some combination of the constraints described below.

3.2. Reverse constraint

The reverse constraint is that $H(x^{\mathrm{R}}, y) \geqslant d$ for all $x, y \in \mathcal{C}$, including $x = y$. It is useful as an intermediate step in constructing codes with the reverse-complement constraint. A natural idea is to start with a code that is fixed by the reverse permutation R, which exchanges column i and column $n + 1 - i$ for $1 \leqslant i \leqslant n$.

This idea is generalized by the following simple lemma:

Lemma 1. *Let $\mathcal{C}'$ be a code of length n such that*:

- *$n = 2k$ is even and $\mathcal{C}'$ has a fixed-point free involution in its permutation group* (*i.e., a permutation of the form $(a_1, a_2) \cdots (a_{2k-1}, a_{2k})$ which leaves no column unchanged*); *or*
- *$n = 2k+1$ is odd and C' has a one-point-fixed involution in its permutation group* (*i.e., a permutation of the form $(a_1, a_2) \cdots (a_{2k-1}, a_{2k})$ which leaves one column unchanged*).

Then $\mathcal{C}'$ is permutation-equivalent to a code $\mathcal{C}$ that has the reverse permutation R in its permutation group.

Proof. Suppose $\mathcal{C}'$ is a code of even length $n = 2k$ with a fixed-point free involution $(a_1, a_2) \cdots (a_{2k-1}, a_{2k})$. Consider the permutation p that sends column a_{2i-1} to column i and column a_{2i} to column $2n + 1 - i$, for $1 \leqslant i \leqslant k$. The argument for odd n is similar. □

The *linear reverse construction* is then defined as follows:

Let $\mathcal{C}'$ be a code over K with minimum distance d which possesses in its permutation group a fixed-point free involution (for n even) or a one-point-fixed involution (for n odd). Then by Lemma 1, $\mathcal{C}'$ is permutation-equivalent to a code $\mathcal{C}$ that is fixed by R. Now since R is an involution, $\mathcal{C}$ can be written as a disjoint union $\mathcal{C} = \mathcal{C}_0 \cup \mathcal{C}_1 \cup \mathcal{C}_2$, where $\mathcal{C}_0$ is the set of codewords in $\mathcal{C}$ that are unchanged by R, and $\mathcal{C}_1$ and $\mathcal{C}_2$ are two sets that are interchanged by R. A set satisfying the reverse constraint together with the Hamming constraint d is obtained by taking $\mathcal{C}_1$ or $\mathcal{C}_2$. Note that of course the sets $\mathcal{C}_1$ and $\mathcal{C}_2$ are not unique.

In the case of linear codes it is easy to compute the subcode $\mathcal{C}_0$. We associate to any reverse permutation R of length n a code $\mathcal{C}_{\mathrm{R}}$ defined by the following generator matrix G_{R}:

– for n even

$$G_{\mathrm{R}} = \begin{pmatrix} 1 & 0 & 0 \cdots & \cdots 0 & 0 & 1 \\ 0 & 1 & 0 \cdots & \cdots 0 & 1 & 0 \\ & \cdots & \cdots & \cdots & \cdots & \\ 0 & \cdots 0 & 1 & 1 & 0 \cdots & 0 \end{pmatrix}$$

– for n odd

$$G_{\mathrm{R}} = \begin{pmatrix} 1 & 0 & 0 & \cdots & & \cdots & 0 & 0 & 1 \\ 0 & 1 & 0 & \cdots & & \cdots & 0 & 1 & 0 \\ & \cdots & & \cdots & & \cdots & & \cdots & \\ 0 & \cdots & 0 & 1 & 0 & 1 & 0 & \cdots & 0 \\ 0 & \cdots & \cdots & 0 & 1 & 0 & \cdots & \cdots & 0 \end{pmatrix}$$

Now obviously $|\mathcal{C}_1| = \frac{|\mathcal{C}|-|\mathcal{C}_0|}{2}$, and $|\mathcal{C}_0|$ can be computed by the following proposition:

Proposition 1. *If $\mathcal{C}$ is a code that is fixed by the reverse permutation R, then the subcode $\mathcal{C}_0$ of $\mathcal{C}$ consisting of the codewords that are unchanged by R is obtained as the intersection of $\mathcal{C}$ and the code $\mathcal{C}_{\mathrm{R}}$.*

Proof. A codeword $c = (c_1, \ldots, c_n)$ is unchanged by R if and only if $c_i = c_{n+1-i}$ for $1 \leqslant i \leqslant n$, which is equivalent to having $c \in \mathcal{C}_{\mathrm{R}}$. □

Note that although we mainly use this construction for linear codes over $GF(4)$, the construction can be generalized to additive codes over $GF(4)$ simply by considering the matrix $G_{\mathrm{R}} + \omega G_{\mathrm{R}}$ rather than G_{R}.

3.3. Reverse-complement constraint

The reverse-complement constraint is that $H(x^{\mathrm{RC}}, y) \geqslant d$ for all $x, y \in \mathcal{C}$, including $x = y$. Again the map $x \to x^{\mathrm{RC}}$ is an involution, although it is not a linear map. To construct codes satisfying the reverse-complement constraint, it can be useful to begin with codes over K that contains a special codeword we denote by 1^K, which is the all-one word for $K = GF(4)$, and is the all-two word for $K = Z_4$.

Starting from a code $\mathcal{C}'$ that contains 1^K and a fixed-point free or a one-point-fixed involution, one may construct a code C that is equivalent to $\mathcal{C}'$ and is of the form $\mathcal{C} = \mathcal{C}_0 \cup \mathcal{C}_1 \cup \mathcal{C}_2$, where $\mathcal{C}_0$ is fixed pointwise under the reverse-complement operation and $\mathcal{C}_1$ and $\mathcal{C}_2$ are interchanged by this operation.

The condition that a code contains 1^K may be avoided in some cases. As in [22], we let $A_4^{\mathrm{R}}(n, d)$ denote the maximum cardinality of a DNA code of length n that satisfies the Hamming distance and reverse constraints for a given d, and let $A_4^{\mathrm{RC}}(n, d)$ denote the maximum cardinality of a DNA code of length n that satisfies the Hamming distance and reverse-complement constraints for a given d. In [22] the following close relationship is given for even n:

$$A_4^{\mathrm{RC}}(n, d) = A_4^{\mathrm{R}}(n, d).$$

This result is obtained through a construction in which the first $n/2$ coordinates of each codeword are replaced by their complements. Unfortunately, for odd n this construction may decrease the minimum distance by one, giving the inequality $A_4^{\mathrm{RC}}(n, d) \geqslant A_4^{\mathrm{R}}(n, d-1)$ in [22]. Here we prove an inequality for odd n that is often tighter:

Proposition 2. *For odd n,*

$$A_4^{\mathrm{RC}}(n, d) \geqslant \frac{A_4^{\mathrm{R}}(n, d)}{2}.$$

Proof. Suppose that $\mathcal{C}$ is a DNA code of size $A_4^{\mathrm{R}}(n, d)$ satisfying the Hamming distance and reverse constraints for a given d, with $n = 2k + 1$ odd. Let $x \to x^*$ denote the map that complements the first $k = \lfloor \frac{n}{2} \rfloor$ coordinates of a codeword x. Clearly, $H(x^*, y^*) = H(x, y)$ for all $x, y \in \mathcal{C}$. We also have $H((x^*)^{\mathrm{RC}}, y^*) = H(x^{\mathrm{R}}, y) + 1$ if $x_{k+1} = y_{k+1}$; $H((x^*)^{\mathrm{RC}}, y^*) = H(x^{\mathrm{R}}, y) - 1$ if $x_{k+1} = \hat{y}_{k+1}$; and $H((x^*)^{\mathrm{RC}}, y^*) = H(x^{\mathrm{R}}, y)$ otherwise. Now consider the subcodes $\mathcal{C}_A, \mathcal{C}_C, \mathcal{C}_G$ and $\mathcal{C}_T$ of $\mathcal{C}$ consisting of codewords in which the $(k+1)$st coordinate is A, C, G and T, respectively. Clearly, $|\mathcal{C}_A| + |\mathcal{C}_C| + |\mathcal{C}_G| + |\mathcal{C}_T| = |\mathcal{C}|$ and one (say $\mathcal{C}'$) of the two codes $\mathcal{C}_A \cup \mathcal{C}_C$ or $\mathcal{C}_G \cup \mathcal{C}_T$ has at least $A_4^{\mathrm{R}}(n, d)/2$ codewords. Note that we never have $x_{k+1} = \hat{y}_{k+1}$ for $x, y \in \mathcal{C}'$, so $H((x^*)^{\mathrm{RC}}, y^*) \geqslant H(x^{\mathrm{R}}, y)$ for $x, y \in \mathcal{C}'$. Therefore $\mathcal{C}'^* = \{c^* | c \in \mathcal{C}'\}$ satisfies the Hamming distance and reverse-complement constraints for the given d, so $A_4^{\mathrm{RC}}(n, d) \geqslant |\mathcal{C}'^*| \geqslant A_4^{\mathrm{R}}(n, d)/2$. □

Remark 1. For odd n, if the code C is linear and the middle column is not always A (or 0), then $|\mathcal{C}_A| = |\mathcal{C}_C| = |\mathcal{C}_G| = |\mathcal{C}_T| = |\mathcal{C}|/4$; equivalent results hold for additive codes.

Remark 2. When starting from a linear (or additive) code, the construction described above in which one takes the complement of the first half of the coordinates does not in general preserve the linearity (or additivity) of the code; nevertheless, it is still possible to apply the Linear Reverse Construction and the method of the previous subsection to compute the size of $|\mathcal{C}_0|$.

3.4. Fixed GC-content constraint

The *GC*-content constraint is that each codeword $x \in \mathcal{C}$ has the same *GC*-weight. Starting from a linear code, the question is how to compute the *GC*-weight enumerator. It can of course be obtained by specializing the complete weight enumerator, but this turns out to be quickly time consuming, since finding the complete weight enumerator may in itself take a long time. We propose in the following a simple way to compute the *GC*-weight enumerator of a code over $GF(4)$ (additive or linear).

Theorem 1. *Let $\mathcal{C}$ be a linear code over $GF(4)$. Then*

$$GCW_{\mathcal{C}}(x, y) = \frac{1}{|\mathcal{C}^{\perp}|} BW_{\mathcal{C}^{\perp}}(2(x + y), 2(x - y)).$$

Proof. $GCW_{\mathcal{C}}(x, y) = CWE_{\mathcal{C}}(x, x, y, y) = \frac{1}{|\mathcal{C}^{\perp}|} CWE_{\mathcal{C}^{\perp}}(2(x + y), 2(x - y), 0, 0) = \frac{1}{|\mathcal{C}^{\perp}|} BW_{\mathcal{C}^{\perp}}(2(x + y), 2(x - y))$. □

Proposition 3. *Let $\mathcal{C}$ be a linear code over $GF(4)$. Then the binary subcode of $\mathcal{C}$ is obtained as the intersection of the two binary codes $\mathcal{C}_1^{\perp}$ and $\mathcal{C}_2^{\perp}$, where $\mathcal{C}_1$ and $\mathcal{C}_2$ are generated by the binary matrices H_1 and H_2 satisfying $H_1 + \omega H_2 = H$, for H a generator matrix of $\mathcal{C}^{\perp}$.*

Proof. Let c be an element of the binary subcode of $\mathcal{C}$ and let H be a generator matrix of $\mathcal{C}^{\perp}$. Then H can be uniquely written as $H = H_1 + \omega H_2$ with H_1 and H_2 binary matrices. Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be the binary codes generated by H_1 and H_2. Then c is in the binary subcode of $\mathcal{C}$ if and only if $H_1 x^t = H_2 x^t = 0$, i.e., if and only if $c \in \mathcal{C}_1^{\perp} \cap \mathcal{C}_2^{\perp}$. □

Now since one is interested in finding as many codewords as possible with the same *GC*-weight, one may prefer even *GC*-weight enumerators, because these contain on average twice as many codewords for each even *GC*-weight than non-even *GC*-weight enumerators. A simple way to construct such codes is given in the following proposition.

Proposition 4. *Let $\mathcal{C}$ be a code over $GF(4)$. If the all-one vector belongs to $\mathcal{C}^{\perp}$, then the GC-weight enumerator of $\mathcal{C}$ is even* (*i.e., has only even weights*).

Proof. By Theorem 1, the *GC*-weight enumerator of $\mathcal{C}$ is $\frac{1}{|\mathcal{C}^{\perp}|} BW_{\mathcal{C}^{\perp}}(2(x + y), 2(x - y))$. Let $W = \sum_{i=0}^{n} A_i x^{n-i} y^i$ be the binary weight enumerator $BW_{\mathcal{C}^{\perp}}$. Let $P_i(x, y) = (x + y)^{n-i}(x - y)^i + (x + y)^i (x - y)^{n-i}$. Then $P_i(x, y) = P_i(x, -y)$, and $P_i(x, y)$ is even in y. Now if the all-one vector is in $\mathcal{C}^{\perp}$, $A_i = A_{n-i}$ and the result follows. □

The technique for turning a code satisfying the reverse constraint into a code satisfying the reverse-complement constraint generalizes to codes with fixed *GC*-content. Let $A_4^{GC,\mathrm{RC}}(n, d, w)$ denote the maximum size of a DNA code of length *n* with constant *GC*-content w that satisfies the Hamming distance and *RC* constraints for a given *d*, and let $A_4^{GC,\mathrm{R}}(n, d, w)$ denote the maximum size of a DNA code of length *n* with constant *GC*-content w that satisfies the Hamming distance and reverse constraints for a given *d*. It is proved in [17] that $A_4^{GC,\mathrm{RC}}(n, d, w) = A_4^{GC,\mathrm{R}}(n, d, w)$ for even *n*, and $A_4^{GC,\mathrm{RC}}(n, d, w) \geqslant A_4^{GC,\mathrm{R}}(n, d-1, w)$ for odd *n*. It is also straightforward to adapt the proof of Proposition 2 to show the following.

Proposition 5. *For odd n,*

$$A_4^{GC,\mathrm{RC}}(n, d, w) \geqslant \frac{A_4^{GC,\mathrm{R}}(n, d, w)}{2}.$$

4. Constructions

Constructions for DNA codes with constant *GC*-content (with and without the *RC* constraint) are given in [17,20,26,27]. But the ‘template-map’ constructions in [20] (for $d \approx n/2$) are suboptimal for $n > 8$ (see [17]), and the lexicographic constructions in [17] become impractical for *n* around 20 (or less for small *d*). Stochastic local search methods as in [26,27] are appealing in that they have access to more of the space of possible codes than lexicographic constructions, and are suitable for larger *n*, particularly when *d* is large. For smaller *d*, the size of codes increases, and both lexicographic and stochastic search methods suffer from doing many pairwise distance comparisons between candidate codewords. By using algebraic constructions for codes we can avoid the explicit computation of distances between pairs of codewords.

Remark. We have in many cases improved the previously published lower bounds on $A_4^{GC}(n, d, w)$ and $A_4^{GC,\mathrm{RC}}(n, d, w)$ for $n \leqslant 12$ by beginning with a lexicographic code as in [17], then enlarging the code using a variant of simulated annealing [12] that uses hybrid neighborhoods similar to those used in [26]. These new bounds are included in the tables in Section 5.

Our interest in constructing DNA codes by starting with linear codes is that there is a well-developed theory of linear codes, and the parameters of the associated DNA codes can be easily computed. This approach is scalable to any reasonable length *n* (say less than 80), and one can use any of the known constructions for linear codes over the different alphabets *K*. The quaternary Hamming codes and shortened or truncated derived codes are interesting, as are the various codes related to the quadratic or duadic codes over the different alphabets.

Note that the Hamming distance between two codewords is unchanged by multiplication of columns by an invertible element of *K*, but this may alter the permutation group and the *GC*-weight enumerator.

4.1. Constructions for DNA codes with fixed GC-content

Suppose one is to construct a DNA code with fixed *GC*-content and minimum distance *d*. The first step is to search for a linear (or additive) code $\mathcal{C}$ which has a large (if not the best possible) number of words with Hamming weight *d*. Lower bounds for such codes can be found for any four elements alphabet in [5] for small parameters, and for $GF(4)$ in [9]. Note also that for small parameters over $GF(4)$ many codes are given in [19].

Now starting from $\mathcal{C}$, since the minimum distance is not changed by column-multiplication with an invertible element, searching for codes that have the all-one vector in their dual seems to be a good choice since it leads to a code with only even *GC*-weights. This is always possible under certain assumptions:

Proposition 6. *Let $\mathcal{C}$ be a linear (or additive) code over $GF(4)$ of length n. Suppose $\mathcal{C}^{\perp}$ has a vector $c = (c_1, \ldots, c_n)$ of weight n. Then $\mathcal{C}$ is equivalent to a code that has the all-one vector in its dual.*

Proof. Let $c = (c_1, \ldots, c_n)$ be a vector of $\mathcal{C}^{\perp}$ of weight *n* and let *G* be a generator matrix of *C*. Vectors *y* of the dual of $\mathcal{C}$ are characterized by $Gy^t = 0$. By multiplying column *i* of $\mathcal{C}$ by c_i, one obtains an equivalent code $\mathcal{C}'$ that has the all-one vector in its dual and therefore has an even *GC*-weight enumerator. □

Note that different codewords *c* in $\mathcal{C}^{\perp}$ may lead to different equivalent codes, and repeating the operation several times may lead to different *GC*-weight enumerators.

Example 1. Consider the [12, 6, 6] extended quadratic residue codes of length 12 over $GF(4)$. This code contains 1848 codewords of *GC*-weight 6, to be compared with the previously best know result 736 of [17].

Example 2. Consider the [21, 18, 3] Hamming code $H_{3,4}$ over $GF(4)$. The dual code has only words of non-null weights 12 or 16. When considering shortened or truncated codes of $H_{3,4}$ it is therefore interesting to truncate or shorten columns depending on words of the dual, so that the dual of the shortened or truncated code has words of weight *n* in its dual.

Remark 1. The number of DNA words of length *n* and *GC*-content w is $\binom{n}{w} 2^n$, which for fixed *n* is largest when $w = \lfloor \frac{n}{2} \rfloor$ (and also for $w = \lceil \frac{n}{2} \rceil$ when *n* is odd). But for some *n* and *d*, the largest code we found of length *n*, minimum Hamming *d*, and constant *GC*-content w was for $w < \lfloor \frac{n}{2} \rfloor$ (see also Section 5). For even *n*, the largest codes with constant *GC*-content were often, but not always, derived from linear codes with the all-one vector in their duals. For example consider $n = 10$ and $d = 4$, in that case one considers a [10, 6, 4] code. There are then 11 possibilities for the *GC*-weight: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}, now if one modifies the code such that the all-one word is in the dual, the *GC*-weight of any codeword has to be even so that there are only 6 possibilities for the *GC*-weight: {0, 2, 4, 6, 8, 10}, since the number of codewords does not change, it means that on the average there are twice as much codewords for each possibility in the case of even *GC*-weight. Practically one finds

1008 codewords with *GC*-weight 5 for a $[10, 6, 4]$ code and 1680 codewords of *GC*-weight 4 if one modifies the code.

Remark 2. More accurate models of melting temperatures take into account the nucleotides at neighboring positions of a codeword, not just the overall *GC*-content or a codeword (see e.g. [24]). From a linear code with minimum weight d, one may test all the codewords having a small range of *GC*-contents with a more accurate model of melting temperature, and keep those that fall within the desired tolerance. One may similarly filter out those codewords having undesirable predicted secondary structure.

4.2. Constructions for DNA codes with fixed GC-content with RC constraint

Suppose now that we also wish to add the *RC* constraint.

For n even, we saw that the R constraint was sufficient, so to optimize our construction we try as many codes $\mathcal{C}$ as possible of minimum weight d with the all-one vector in their dual, obtained by the method of the previous section. For each code, we compute the permutation group of the code and search for a fixed-point free involution so that we may apply Lemma 1. We then keep the code that is unchanged by R having the most codewords with a fixed *GC*-content, and we apply the linear reverse construction of Section 2. Note that if the all-one vector is in $\mathcal{C} \cap \mathcal{C}^{\perp}$, the construction also works.

For n odd, there are two possibilities for dealing with the *RC* constraint before applying the linear reverse construction:

– starting with a code with good parameters, searching for equivalent codes with a one-point-fixed involution in their permutation groups, and then applying Proposition 2,
– starting with a code with good parameters, and constructing equivalent codes that contain the all-one vector and a one-point-fixed involution in their permutation groups.

Note that one can construct equivalent codes that contain the all-one vector in the following way: suppose $\mathcal{C}$ has a codeword $c = (c_1, \ldots, c_n)$ of weight n; then multiplying the column i of $\mathcal{C}$ by c_i^2 gives an equivalent code that contains the all-one vector. One has then only to try different codes in order to find one that also has a one-point-fixed involution in its permutation group.

One may also wonder whether it is possible to have the all-one vector both in the code and its dual. Unfortunately this is not possible for odd n, since in this case the all-one vector is not orthogonal to itself.

Example. Consider the $[18, 9, 8]$ quaternary cyclic code of [19]. Puncturing this code in two columns leads to a $[16, 9, 8]$ code C_{16} (say). The dual of this code contains words of weight 16, which can be used to construct codes equivalent to C_{16} whose duals contain the all-one vector. Different choices of different codewords of weight 16 lead to different codes which have all the all-one vector in their dual. Then for each code with the all-one vector in its dual one searches for a code with a fixed-point free involution in its permutation group. A certain number of trials (around 100 for this code) leads to a code which has a fixed-point free involution in its permutation group and 6600 words of *GC*-weight 8, from which one extracts 3264 words by the linear reverse construction. Eventually, one obtains a DNA code with 3264 words which satisfies the *RC* constraint with $d = 8$ and which has constant *GC*-content 8.

Remark. The fact that it is not possible to have the all-one vector in both a code and its dual for odd lengths explains the fact that our results are relatively less interesting for odd lengths.

4.3. Parameters for infinite families with fixed GC-content

4.3.1. Binary construction

A trivial construction consists of considering a linear binary code as a code over $GF(4)$.

The following two facts are true:

(1) If $\mathcal{C}$ is a binary $[n, k, d]$ code, then $\mathcal{C}$ considered as a code over $GF(4)$ is also a $[n, k, d]$ code over $GF(4)$.

(2) The binary subcode of the dual of $\mathcal{C}$ as a code over $GF(4)$ is the binary dual of $\mathcal{C} : \mathcal{C}^{\perp}$.

For instance the extended Hamming code for $n = 8$ and $d = 4$ leads to the best known result with *GC*-content 4 (cf. Tables in the next section).

Infinite families of linear binary codes with known weight enumerators like the Hamming codes or the Reed–Muller codes of order 2 [21] therefore lead to infinite families with known *GC*-content. This construction may also be used for binary codes whose automorphism groups are known to contain fixed-point free involutions.

4.3.2. Quaternary Hamming code

Let H_r be the $[\frac{4^r-1}{3}, \frac{4^r-1}{3} - r, 3]$ quaternary Hamming code of order *r*.

Proposition 7. *When $r \equiv 1$ or $2 \pmod 3$, then the GC-weight enumerator of H_r is*

$$2^{(4^r-1)/3-2r}(x+y)^n.$$

Proof. When $r \equiv 1$ or $2 \pmod 3$, H_r is cyclic (and is constacyclic otherwise [19] and its dual H_r is the $\left[\frac{4^r-1}{3}, r, 4^{r-1}\right]$ Simplex code S_r, which is also cyclic. Now suppose S_r contains a binary word. Since S_r is cyclic and its dual is generated by only one irreducible polynomial, this would mean that S_r has a binary generator matrix, which is not possible since the Sphere-Packing Bound would not be satisfied. □

4.3.3. Z_4 Kerdock codes

The infinite family of binary Kerdock codes $K(r+1)$ of length 2^{r+1} for odd *r* are the Gray images of the Z_4-linear Kerdock codes $K_4(r)$ of length 2^r [14]. This leads to the following construction.

Proposition 8. *For r odd, there exists a DNA code of length 2^r with Hamming weight $2^{r-1} + 2^{r-2} - 2^{(r-3)/2}$, $2^r(2^r - 1)$ codewords and GC-weight 2^{r-1}.*

Proof. From [14], the codewords of weights $2^r - 2^{(r-1)/2}$ of the Kerdock code $K(r+1)$ are the binary images of the set of codewords S_r of $K_4(r)$ having 2^{r-1} coordinates with value 1 or 3, having $2^{r-2} - 2^{(r-3)/2}$ coordinates with value 2, and having Hamming weight $2^{r-1} + 2^{r-2} - 2^{(r-3)/2}$.

Let c be a codeword of S_r. Then $2c$ is in $K_4(r)$ and $c + 2c = 3c$ remains in $K_4(r)$. Hence keeping only one vector of the two vectors c and $3c$, one derives for any odd r a DNA code of length 2^r, minimum distance $2^{r-1} + 2^{r-2} - 2^{(r-3)/2}$ and fixed *GC*-content 2^{r-1} with $2^r(2^r - 1)$ codewords. □

For $r = 3, n = 8, d = 5$, the set corresponds to the best DNA code known (cf. next section).

5. Tables

In the following tables we give lower bounds for $A_4^{GC}(n, d, w)$ (Table 1) and $A_4^{GC,\mathrm{RC}}$ (n, d, w) (Table 2) for $d \leqslant n \leqslant 20$. The constant *GC*-content w is usually taken to be $\lfloor n/2 \rfloor$, but in some cases linear constructions give larger codes for some other value of w (usually even) in the range from $\lfloor n/2 \rfloor - 1$ and $\lfloor n/2 \rfloor + 1$. As the goal is to find the largest code with *any* fixed *GC*-content, we have included these codes in the tables. (Note that $A_4^{GC}(n, d, w) = A_4^{GC}(n, d, n - w)$ and $A_4^{GC,\mathrm{RC}}(n, d, w) = A_4^{GC,\mathrm{RC}}(n, d, n - w)$.)

Bounds for $d = 2$ are not given in the tables. In [17] it is shown that $A_4^{GC}(n, 2, w) = \binom{n}{w} 2^{n-1}$ for all n, and that $A_4^{GC,\mathrm{RC}}(n, 2, w) = \binom{n}{w} 2^{n-2}$ for even n. It is also shown in [17] that $A_4^{GC,\mathrm{R}}(n, d, w) \geqslant A_2^{\mathrm{R}}(n, d, w) \cdot A_2(n, d)$ for all n, and essentially the same argument can be used to show the following:

Proposition 9. *For all n,*

$$A_4^{GC,\mathrm{RC}}(n, d, w) \geqslant A_2^{\mathrm{R}}(n, d, w) \cdot A_2(n, d).$$

Then since $A_2^{\mathrm{R}}(n, d, w) = \left[\binom{n}{w} - \binom{\lfloor n/2 \rfloor}{\lfloor w/2 \rfloor}\right]/2$, this gives $\left[\binom{n}{w} - \binom{\lfloor n/2 \rfloor}{\lfloor w/2 \rfloor}\right] 2^{n-2} \leqslant A_4^{GC,\mathrm{RC}}(n, 2, w) \leqslant \binom{n}{w} 2^{n-2}$ for all n. This lower bound can be improved on; for example we constructed codes of size 74, 1090, 15,918 and 231,424 for $n = 5, 7, 9$ and 11, respectively.

Constructions derived from linear codes are especially interesting for d up to roughly $n/2$; for higher d, except for very special codes we found larger codes using non-linear constructions (a combination of lexicographic constructions and stochastic search).

Linear constructions follow the method described in the previous section. We started from the best know quaternary codes of [9]; of special interest were all of the good cyclic codes, particular those described in [19]. Note that the extended quaternary [12, 6, 6], [14, 7, 6], [20, 10, 8] and [30, 15, 12] quadratic residue codes together with the duadic [18, 9, 8] code and the BCH codes of length 15 and 17 lead to many other good codes through the usual shortening and truncating constructions [15]. Since these codes have in general a big automorphism group, they can be used to construct DNA codes satisfying the *GC* constraint alone, or satisfying both the *GC* and *RC* constraints.

Table 1

(a) *Lower bounds for* $A_4^{GC}(n, d, w)$ *with* $n \leqslant 20$, $d \leqslant 9$

$n\backslash d$	3	4	5	6	7	8	9
4	$12.^{l}$	$4.^{p}$	—	—	—	—	—
5	30^{m}	$10.^{l}$	$3.^{p}$	—	—	—	—
6	112^{m}	$40.^{l}$	8^{l}	$4.^{p}$	—	—	—
7	274^{m}	72^{m}	22^{l}	$7.^{l}$	$3.^{p}$	—	—
8	1056^{m}	224^{t}	56^{c}	24^{s}	$5.^{l}$	$4.^{p}$	—
9	3012^{m}	555^{m}	133^{m}	40^{l}	16^{m}	$5.^{l}$	$3.^{p}$
10	10128^{m}	1680^{c}	420^{c}	116^{c}	32^{l}	$16.^{l}$	$5.^{l}$
11	32352^{m}	7392^{c}	1848^{c}	462^{c}	72^{l}	32^{l}	$10.^{l}$
12	118272^{c}	29568^{c}	2994^{m}	1848^{c}	179^{m}	68^{l}	23^{m}
13	473088^{c}	109824^{c}	8614^{m}	1921^{m}	440^{l}	134^{m}	44^{m}
14	1537536^{c}	384384^{c}	27456^{l}	6076^{c}	1534^{c}	404^{c}	112^{m}
15	6589440^{c}	1647360^{c}	96096^{c}	25740^{c}	6470^{c}	1575^{c}	225^{m}
16	26357760^{c}	6589440^{c}	411840^{c}	111360^{c}	25880^{c}	6680^{c}	532^{m}
17	105431040^{c}	26357760^{c}	1555840^{c}	390080^{c}	48620^{c}	24310^{c}	1272^{l}
18	210862080^{c}	26357760^{c}	5601024^{c}	1400704^{c}	87516^{c}	87516^{c}	3192^{l}
19	756760576^{c}	94595072^{c}	22404096^{c}	5922048^{c}	370128^{c}	92378^{c}	6924^{m}
20	3027042304^{c}	378380288^{c}	94595072^{c}	23688192^{c}	1478048^{c}	369120^{c}	23100^{c}

(b) *Lower bounds for* $A_4^{GC}(n, d, w)$ *with* $n \leqslant 20$, $10 \leqslant d \leqslant 20$

$n\backslash d$	10	11	12	13	14	15	16	17	18	19	20
10	$4.^{p}$	—	—	—	—	—	—	—	—	—	—
11	$4.^{l}$	$3.^{p}$	—	—	—	—	—	—	—	—	—
12	$9.^{m}$	$4.^{p}$	$4.^{p}$	—	—	—	—	—	—	—	—
13	20^{l}	$8.^{m}$	$4.^{l}$	$3.^{p}$	—	—	—	—	—	—	—
14	38^{l}	16^{m}	$8.^{l}$	$4.^{p}$	$4.^{p}$	—	—	—	—	—	—
15	107^{c}	30^{c}	13^{m}	$6.^{m}$	$4.^{m}$	$3.^{p}$	—	—	—	—	—
16	177^{l}	117^{c}	60^{c}	$12.^{m}$	$5.^{m}$	$4.^{p}$	$4.^{p}$	—	—	—	—
17	380^{l}	132^{l}	123^{c}	22^{m}	$9.^{m}$	$5.^{m}$	$4.^{m}$	$3.^{p}$	—	—	—
18	920^{l}	216^{m}	123^{c}	38^{m}	18^{m}	$9.^{m}$	$5.^{m}$	$4.^{p}$	$4.^{p}$	—	—
19	1326^{m}	431^{m}	163^{m}	71^{m}	33^{m}	15^{m}	$8.^{m}$	$5.^{m}$	$4.^{m}$	$3.^{p}$	—
20	5882^{c}	1461^{c}	401^{c}	130^{m}	58^{m}	31^{c}	13^{m}	$8.^{m}$	$5.^{m}$	$4.^{p}$	$4.^{p}$

Note, moreover that $A_4^{GC,\mathrm{RC}}(n, d, w) \leqslant A_4^{GC,\mathrm{RC}}(n, d-1, w)$, and that $A_4^{GC,\mathrm{RC}}(n, d, w) \leqslant A_4^{GC,\mathrm{RC}}(n+1, d, w)$ (for even n simply add a column of A's in the center coordinate; for odd n, insert a column immediately after the center column which is T whenever the center column is T and is A otherwise).

Remark. The techniques described in this paper can also be used to improve many of the lower bounds for $A_4^{\mathrm{RC}}(n, d)$ (with unrestricted GC-content) given in [22,27].

The following notation is used in the tables:

– 'c' means coding construction as described above;
– 'p' means construction from Proposition 1 of [17];
– 'l' means lexicographic construction as in [17];

Table 2

(a) *Lower bounds for* $A_4^{GC,\mathrm{RC}}(n,d,w)$ *with* $n \leqslant 20$, $d \leqslant 9$

$n\backslash d$	3	4	5	6	7	8	9
4	$6.^l$	$2.^p$	—	—	—	—	—
5	15^l	3^l	$1.^p$	—	—	—	—
6	43^m	16^l	$4.^l$	$2.^p$	—	—	—
7	133^m	35^m	11^m	2^l	$1.^p$	—	—
8	528^m	112^c	27^m	12^s	$2.^p$	$2.^p$	—
9	1354^m	273^m	65^m	19^m	8^m	$2.^l$	$1.^p$
10	4542^m	840^c	170^m	54^c	15^l	$8.^l$	$2.^p$
11	14405^m	2457^m	463^m	113^m	35^m	12^m	$5.^m$
12	58976^c	14624^c	1369^m	924^c	81^l	27^l	11^m
13	167263^m	27376^c	3954^l	924^c	200^m	59^m	21^m
14	430080^c	192192^c	11878^c	2963^c	749^c	180^c	46^c
15	1646240^c	411821^c	25670^c	6430^l	1600^c	343^l	102^l
16	13174400^c	3293600^c	55376^c	55376^c	12864^c	3264^c	230^l
17	26355520^c	6587200^c	97450^c	97450^c	12864^c	6060^c	549^l
18	44808192^c	11202048^c	698592^c	698592^c	41784^c	10496^c	1403^l
19	47102080^c	23647760^c	698592^c	698592^c	46838^m	11319^c	3462^m
20	756760576^c	189189536^c	11806240^c	11806240^c	184756^c	184756^c	11452^c

(b) *Lower bounds for* $A_4^{GC,\mathrm{RC}}(n,d,w)$ *with* $n \leqslant 20$, $10 \leqslant d \leqslant n$

$n\backslash d$	10	11	12	13	14	15	16	17	18	19	20
10	$2.^p$	—	—	—	—	—	—	—	—	—	—
11	$2.^m$	$1.^p$	—	—	—	—	—	—	—	—	—
12	4^l	$2.^p$	$2.^p$	—	—	—	—	—	—	—	—
13	9^m	4.	$2.^m$	$1.^p$	—	—	—	—	—	—	—
14	15^m	7^m	$4.^m$	$2.^p$	$2.^p$	—	—	—	—	—	—
15	35^l	18^m	6^m	$3.^m$	$2.^m$	$1.^p$	—	—	—	—	—
16	74^l	52^c	24^c	5^m	$2.^p$	$2.^p$	$2.^p$	—	—	—	—
17	164^l	56^l	30^c	11^m	$4.^m$	$2.^m$	$2.^m$	$1.^p$	—	—	—
18	387^l	104^m	43^m	19^m	9^m	$4.^m$	$2.^p$	$2.^p$	$2.^p$	—	—
19	909^m	215^m	80^m	35^m	16^m	7^m	$4.^m$	$2.^m$	$2.^m$	$1.^p$	—
20	2868^c	766^c	179^c	64^m	29^m	14^m	6^m	$4.^m$	$2.^p$	$2.^p$	$2.^p$

- ‘*t*’ means template-map construction from [20]
- ‘*s*’ means stochastic local search from [26];
- ‘*m*’ means miscellaneous new construction (usually using simulated annealing, sometimes with a lexicographic code as a seed);
- ‘.’ means the lower bound is optimal since it equals the Johnson-type upper bound given in [17];

Note that sometimes different types of constructions give codes of the same size; for example in Table 1(a), codes of size 224 for $(n,d,w) = (8,4,4)$ can be found using a template-map construction [20], a lexicographic construction [17], and a coding construction. To avoid using multiple subscripts, in the tables we give preference to *p* over *t* over

c over l over s over m, corresponding roughly to a preference for simpler or more structured constructions. In the tables, all of the lower bounds for odd n and for $n > 12$ (odd or even) are new, except those with the superscript p. For $n \in \{4, 6, 8, 10, 12\}$, all of the lower bounds with superscripts m are new, and those with superscript c are new except for $(n, d, w) = (8, 5, 4)$ in Table 1(a) and $(n, d, w) = (8, 4, 4)$ in Table 2(a). The tables are also available on the web at `http://csua.berkeley.edu/~ok/dnacodes.html` and we welcome updates.

5.1. Remarks on running times

For codes of length n and GC-context w, the running time for the lexicographic constructions we used scales roughly like $|\mathcal{C}|n\binom{n}{w}2^n$, where $|\mathcal{C}|$ is the size of the resulting code, which increases as d decreases. (Bounds on the size of the resulting code can be computed in advance using the methods in [17].) We used both random codes and lexicographic codes with random offsets as seeds for the stochastic search algorithms, which we ran until we were bored (generally because the codes had stopped improving for a while). We did not keep track of all of the run times, but they mostly ranged from a few minutes to a few days depending on n, d and w. (For example, in Table 2(b) the code of size 528 satisfying the RC and GC-content constraints for $(n, d, w) = (8, 3, 4)$ took 4 h (CPU-time) to construct on a 2 GHz Pentium 4 computer; three minutes total were spent constructing 200 lexicographic codes using different random offsets—the largest of these codes had size 383, and the remaining time was spent improving this code using stochastic search.) We did not attempt either lexicographic or stochastic constructions when d was much smaller than n (roughly $d < n - 12$).

All of the computations for linear codes were done with the Magma system [6], and the running time was usually a few seconds for small n up to a few minutes for $n = 20$.

References

[1] L.M. Adleman, Molecular computation of solutions to combinatorial problems, Science 266 (1994) 1021–1024.

[2] M. Arita, S. Kobayashi, DNA sequence design using templates, New Generation Comput. 20 (2002) 263–278.

[3] A. Ben-Dor, R.M. Karp, B. Schwikowski, Z. Yakhini, Universal DNA tag systems: a combinatorial design scheme, J. Comput. Biol. 7 (2000) 503–519.

[4] H. Bi, J. Chen, R. Deaton, M.H. Garzon, H. Rubin, D.H. Wood, In vitro selection of non-crosshybridizing oligonucleotides for computation, Natural Comput. 2 (2003) 417–426.

[5] G.T. Bogdanova, A.E. Brouwer, S.N. Kapralov, P.R.J. Östergård, Error correcting codes over an alphabet with four elements, Designs Codes Cryptography 23 (2001) 333–342.

[6] W. Bosma, J. Cannon, Handbook of Magma Functions, Sydney, 1995.

[7] A. Brenneman, A. Condon, Strand design for bio-molecular computation, Theoret. Comput. Sci. 287 (2002) 39–58.

[8] S. Brenner, R.A. Lerner, Encoded combinatorial chemistry, Proc. Natl. Acad. Sci. USA 89 (1992) 5381–5383.

[9] A.E. Brouwer, Bounds on the size of linear codes, in: V.S. Pless, W.C. Huffman (Eds.), Handbook of Coding Theory, North-Holland, Amsterdam, 1998, pp. 295–461.

[10] E.M. Coven, N. Jonoska, DNA hybridization, shifts of finite type, and tiling of the integers, in: C. Mart:'in-Vide, V. Mitrana (Eds.), Grammars and Automata for String Processing, Taylor & Francis, London, 2003, pp. 369–380.

[11] R. Deaton, M. Garzon, R.C. Murphy, J.A. Rose, D.R. Franceschetti, S.E. Stevens Jr., Genetic search of reliable encodings for DNA-based computation, in: J.R. Koza, D.E. Goldberg, D.B. Fogel, R.L. Riolo (Eds.), Late Breaking Papers at the First Annual Conference on Genetic Programming, Stanford Bookstore GP-96B, 1996, pp. 9–15.

[12] A.A. El Gamal, L.A. Hemachandra, I. Shperling, V.K. Wei, Using simulated annealing to design good codes, IEEE Trans. Inform. Theory 33 (1987) 116–123.

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

[14] A.R. Hammons Jr., P.V. Kumar, A.R. Calderbank, N.J.A. Sloane, P. Solé, The Z_4-linearity of Kerdock, Preparata, Goethals and related codes, IEEE Trans. Inform. Theory 40 (1994) 301–319.

[15] W.C. Huffman, V. Pless, Fundamentals of Coding Theory, Cambridge University Press, Cambridge, 2003.

[16] L. Kari, S. Konstantinidis, E. Losseva, G. Wozniak, Sticky-free and overhang-free DNA languages, Acta Inform. 40 (2003) 119–157.

[17] O.D. King, Bounds for DNA codes with constant GC-content, Electron. J. Combin. 10 (2003) R33 13pp.

[18] S. Kobayashi, T. Kondo, M. Arita, On template method for DNA sequence design, in: M. Hagiya, A. Ohuchi (Eds.), DNA Computing: Eighth Internat. Workshop on DNA-Based Computers, Lecture Notes in Computer Science, Vol. 2568, Springer, Berlin, 2002, pp. 205–214.

[19] F. Kschischang, S. Pasupathy, Some ternary and quaternary codes and associated sphere packings, IEEE Trans. Inform. Theory 38 (1992) 227–246.

[20] M. Li, H.J. Lee, A.E. Condon, R.M. Corn, DNA word design strategy for creating sets of non-interacting oligonucleotides for DNA microarrays, Langmuir 18 (2002) 805–812.

[21] F.J. MacWilliams, N.J.A. Sloane, The Theory of Error Correcting Codes, North-Holland, Amsterdam, 1977.

[22] A. Marathe, A.E. Condon, R.M. Corn, On combinatorial DNA word design, J. Comput. Biol. 8 (2001) 201–220.

[23] V. Rykov, A. J. Macula, D. Torney, P. White, DNA sequences and quaternary cyclic codes, IEEE Isit, 2001, Washington, pp. 248–248.

[24] J. SantaLucia Jr., A unified view of polymer, dumbbell, and oligonucleotide DNA nearest-neighbor thermodynamics, Proc. Natl. Acad. Sci. USA 95 (1998) 1460–1465.

[25] D. Shoemaker, D.A. Lashkari, D. Morris, M. Mittmann, R.W. Davis, Quantitative phenotypic analysis of yeast deletion mutants using a highly parallel molecular bar-coding strategy, Nature Genetics 16 (1996) 450 –456.

[26] D.C. Tulpan, H.H. Hoos. Hybrid randomised neighbourhoods improve stochastic local search for DNA code design, in: Y. Xiang, B. Chaib-draa (Eds.), Advances in Artificial Intelligence: 16th Conf. of the Canadian Society for Computational Studies of Intelligence, Lecture Notes in Computer Science, Vol. 2671, Springer, Berlin, 2003, pp. 418–433.

[27] D.C. Tulpan, H.H. Hoos, A.E. Condon, Stochastic local search algorithms for DNA word design, in: M. Hagiya, A. Ohuchi (Eds.), DNA Computing: Eighth International Workshop on DNA-Based Computers, Lecture Notes in Computer Science, Vol. 2568, Springer, Berlin, 2002, pp. 229–241.