AFRL-IF-RS-TR-2003-57
Final Technical Report
March 2003

DNA TAG-ANTITAGS (TAT) CODES

State University of New York @ Geneseo

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2003-57 has been reviewed and is approved for publication.

APPROVED:

TOM RENZ
Project Engineer

FOR THE DIRECTOR:

JAMES A. COLLINS, Acting Chief
Information Technology Division
Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503

1. AGENCY USE ONLY (Leave blank)

2. REPORT DATE
MARCH 2003

3. REPORT TYPE AND DATES COVERED
Final Sep 02 – Feb 03

4. TITLE AND SUBTITLE
DNA TAG-ANTITAGS (TAT) CODES

5. FUNDING NUMBERS
C - F30602-02-2-0205
PE - 61102F
PR - DNAT
TA - AT
WU - 02

6. AUTHOR(S)
Anthony J. Macula

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
State University of New York Geneseo
The Research Foundation
1 College Circle
Geneseo New York 14454

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory/IFTC
26 Electronic Parkway
Rome New York 13441-4514

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
AFRL-IF-RS-TR-2003-57

11. SUPPLEMENTARY NOTES
AFRL Project Engineer: Tom Renz/IFTC/(315) 330-3423

12a. DISTRIBUTION / AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 Words)***

This research addresses the initial stages of the development of an enabling technology for DNA computing and other biological assay applications. This work combines mathematics, computer science and chemistry. It is focused on the construction of a biomolecular architecture designed to employ new algorithmic paradigms based on the massively parallel computational power of DNA hybridization. The ultimate intent is to develop a computing basis to eventually overcome the exponential time complexity of many discrete math problems so that they can be solved in linear real time. Many of these computationally hard (NP) problems are critical to logistics, scheduling and security. In particular, we made an initial application of biomolecular computing methods to data mining. Data mining has important applications to information security, assurance and superiority.

In this research, we developed methods of generating large collections of single stranded DNA sequences called a DNA (n,d)code. DNA(n,d) codes serve as universal components for biomolecular computing. DNA(n,d) codes are closed under reverse-complementation. The strands in a DNA(n,d) code have such binding specificity that a code strand will only hybridize with its reverse-complement and will not cross hybridize with any other code strand in the DNA(n,d) code. Such collections of strands are crucial to the success of DNA computing.

14. SUBJECT TERMS
Biomolecular Computing, DNA Sequencing, Information Optimization

15. NUMBER OF PAGES
23

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT
UNCLASSIFIED

18. SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

19. SECURITY CLASSIFICATION OF ABSTRACT
UNCLASSIFIED

20. LIMITATION OF ABSTRACT
UL

Table of Contents

1. Summary 1
2. Introduction 3
3. Methods, Assumptions, Procedures 6
   3.1 Insertion-Deletion Codes 6
   3.2 DNA(n,d) Codes 8
4. Results, Discussion 9
   4.1 DNA(n,d) Code Generation Programs 9
5. Conclusions 11
   5.1 DNA Computing with DNA(n,d) Codes 11
   5.2 DNA Computing and Data Mining 14
6. References 16

List of Figures

FIGURE 1 2
FIGURE 2 Intended Duplex 4
FIGURE 3 Unintended Duplex 5
FIGURE 4 10
FIGURE 5 Encoding of the Sequence 1000101 12
FIGURE 6 Correct DNA Bit String 13
FIGURE 7 Incorrect DNA Bit String 13
FIGURE 8 DNA Bit String Interactions 14
FIGURE 9 14
FIGURE 10 Patterns in Data are Being Transformed Into Molecules 16

List of Tables

TABLE 1 An example of a DNA(8,3) partition 9
TABLE 2 Using the uniform distribution method, we could only generate DNA(15,5) code of size 14 11

1. Summary

This research addresses the initial stages of the development of an enabling technology for DNA computing and other biological assay applications. This work combines mathematics, computer science and chemistry. It is focused on the construction of a biomolecular architecture designed to employ new algorithmic paradigms based on the massively parallel computational power of DNA hybridization. The ultimate intent is to develop a computing basis to eventually overcome the exponential time complexity of many discrete math problems so that they can be solved in linear real time. Many of these computationally hard (NP) problems are critical to logistics, scheduling and security. In particular, we made an initial application of biomolecular computing methods to the identification of important patterns in data, i.e., data mining. Data mining has important applications to information security, assurance and superiority.

In this research, we developed methods of generating large collections of single stranded DNA sequences called a DNA(n,d) code. DNA(n,d) codes serve as universal components for biomolecular computing. DNA(n,d) codes are closed under reverse-complementation. The strands in a DNA(n,d) code have such binding specificity that a code strand will only hybridize with its reverse-complement and will not cross hybridize with any other code strand in the DNA(n,d) code. Such collections of strands are crucial to the success of Adleman [1] style DNA computing [4], [15], [24]. The collections also have important applications in many other biological assays. Some of the other applications are single nucleotide polymorphism (SNP) genotyping [9], gene expression profiling [5], DNA chip development [11], [22], [37], and self-assembly [41].

In this proposal, we think of the strand design problem as a mathematical coding theory problem and we use the insertion-deletion metric as our constraint. This approach has been previously suggested [2], [33]. It was initially implemented in [9] with excellent binding specificity. We have dramatically improved the initial research in [9].

Two codewords **x** and **y** are at insertion-deletion distance at least d+1 if and only if their longest common subsequence has length at most n-d-1. This means that if up to d deletions (of sequence entries) are made in any codeword **x**, the resulting (and shorter) q-ary sequence could not have been obtained by deleting up to d entries in any other codeword **y** with $\mathbf{y} \neq \mathbf{x}$. Since two 3'-5' DNA sequences **x**, **y** of length n can form s bonds in a duplex only if **x** and the complement of **y** have a common subsequence of length s the insertion-deletion metric is the only metric that can model this cross-hybridization bonding constraint

We wrote programs that generate very good random and pseudo-random DNA(n,d) ($ID_q(n,d)$) codes. On a 2.3 ghtz Pentium PC, we generated a DNA(20,5) code of size 3038. This is a ten-fold increase other previously published constructions. The reason for this improvement stems from the Markov chain approach we used to generate candidate code words.

Given two n-sequences **x**, **y**, all of our programs use a "folklore" dynamic programming algorithm to find the longest common subsequence ($\mathrm{lcs}(\mathbf{x},\mathbf{y})$) between **x** and **y**. Our very first program used a (reverse-)complement cyclic code as an initial set from which to find a DNA(n,d) code as a subcode. However, we eventually realized that cyclic codes have too much symmetry to generate large-size DNA subcodes. The next step was based on generating uniformly distributed independent random n-sequences, the volumes of the spheres of a fixed radius centered at many of these n-sequences is too large. We did not get the desired performance here. Our most recent programs generate candidate n-sequences **x** in the following way. The value of x_1 is selected from {a,c,g,t} with uniform probability. Then, the remaining entries are generated by a stationary Markov chain given by transition matrix M_k with parameter k. Then, the remaining entries are generated by a stationary Markov chain given by transition matrix M_k with parameter k.

$$M_k = \begin{array}{c} \\ a \\ c \\ g \\ t \end{array} \begin{array}{c} \begin{array}{cccc} a & c & g & t \end{array} \\ \begin{pmatrix} \frac{k-3}{k} & k^{-1} & k^{-1} & k^{-1} \\ k^{-1} & \frac{k-3}{k} & k^{-1} & k^{-1} \\ k^{-1} & k^{-1} & \frac{k-3}{k} & k^{-1} \\ k^{-1} & k^{-1} & k^{-1} & \frac{k-3}{k} \end{pmatrix} \end{array}$$

FIGURE 1

These sequences have low volume spheres of the desired radius. However, the higher the k, the fewer the number of sequences generated. Our programs take a dynamic heuristic approach. These programs start a high value of k and then check all values of the permitted length of the longest common subsequence. This continues for a set number of cycles over which no new codeword is added. The next value of k is set by finding the next highest value of k for which a codeword can be added to the growing code. Here dichotomy is used. This heuristic has worked very well and is much better than the uniform codeword generation method. For example, by using our Markov chain heuristic, we can generate DNA(15,5) code of size 104. Using the uniform distribution method, we could only generate DNA(15,5) code of size 14.

Our DNA(n,d) codes can be applied to biomolecular computing as described in [6]. Note that because of the properties of our DNA(n,d) code, we need only half as many distinct strands as were used in [6]. We show that a DNA(n,d) code can be used to encode *and* filter. In [6], a strand either encodes *or* filters exclusively. We have applied this approach to the data mining problem of the identification of maximal frequent sets. In short, using a DNA(n,d) code as universal components, an assembly process creates single stranded DNA molecules called DNA bit strings that store and retrieve information. For a given database, a library of DNA bit strings is past through (algorithmically constructed) filters. Then the DNA bit strings that remain represent maximal frequent sets in the database.

2. Introduction

In [1], [6], [17], [26] it has been shown that the hybridization that occurs between a DNA strand and its Watson-Crick complement can be used to perform mathematical computation. The promise of DNA computing is that the massive parallelism of DNA hybridization reactions can be exploited to overcome the time complexity (via a silicon based computer) of an important class discrete mathematical problems so that they can be solved in real time. However, to achieve the full potential of DNA computing, many technological hurdles need to be overcome. This work addresses this issue.

In this research large collections of single stranded DNA sequences called a DNA(n,d) code are developed. DNA(n,d) codes are closed under reverse-complementation. The strands in a DNA(n,d) code have such binding specificity that a code strand will only hybridize with its reverse-complement and will not cross hybridize with any other code strand in the DNA(n,d) code. Such collections of strands are crucial to the success of Adleman [1] style DNA computing [4], [15], [24]. The collections also have important applications in many other biological assays [3], [4], [5], [8], [9], [11], [22], [37], [41].

Single strands of DNA are modeled by directed sequences of letters from the alphabet {A, C, G, T} where A, T and C, G are called *complementary pairs.* Two oppositely directed DNA sequences are capable of coalescing into a duplex. Because an A (C) in one strand can only bind to a T (G) in the oppositely directed strand, the greatest energy of duplex formation is obtained when the two sequences are *reverse-complements* (a.k.a. *Watson-Crick complements*) of one another. This annealing process is referred to as *DNA hybridization*. For example, given the strand 3'GTATTGAT5' (directed 3' to 5'), the oppositely directed (5' to 3') strand 5'ATCAATAC3' is the reverse complement. <u>Henceforth, the terms complement and reverse-complement are synonymous.</u> See FIGURE 2. Since molecules can turn over in solution, our pictures are intended to capture this. Because we are accustom to working with the bottom strand of a duplex, the numbering of our sequences is 3'-5' rather than the more customary 5'-3'.

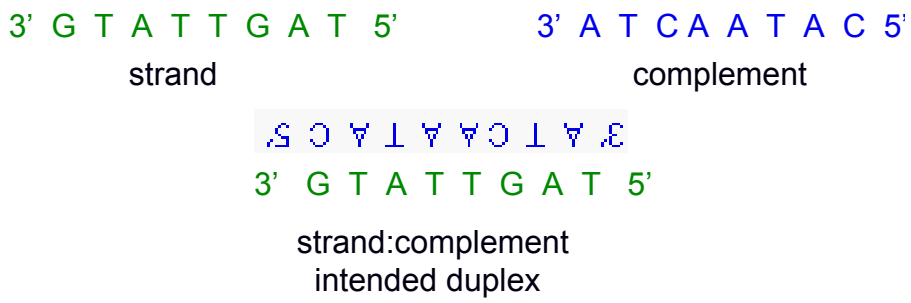

FIGURE 2

Hybridization assays offer the possibility of *simultaneously* processing trillions of bits of information. In DNA hybridization assays for biomolecular computing, DNA strands can be used for multiple purposes. They can be used to store, write, read and retrieve information. Hybridization assays with DNA strands are also used to separate, manipulate, identify and address molecules in many other important experiments beyond biomolecular computing [3], [4], [5], [8], [9], [11], [22], [37], [41].

In DNA computing hybridization assays, each strand in the assay must hybridize much more strongly to *its* complement strand than to any other strand or any other complement strand. In such assays, DNA strands are synthesized and "labeled" or "fixed" DNA probes are allowed to hybridize with the synthesized DNA strands in a controlled and algorithmic way. The resulting hybridized and "labeled" or "fixed" DNA molecules contain information (i.e., solutions to problems) that can in turn be "read" by further hybridization reactions or other means. An example of this type of paradigm is the sticker method [33].

The advantage of this method is that it uses universal components that can be mass produced. We call the collection of universal components a DNA(n,d) code. See Definition 2. The main problem with this basic method is that unintended cross-hybridization is a main source of errors.

Thus the problem is to avoid the formation of unintended duplexes in a DNA(n,d) code. In FIGURE 3, we give some examples of how this could happen. In FIGURE 3, the pairs (R, L) and (r, l) are complements. The only intended duplexes are R:L and r:l. Base pairing can (possibly) occur between the red bases in the unintended duplexes listed in FIGURE 3. The other unintended duplexes not shown in FIGURE 3 are R:l, R:R, L:L, l:L and l:l.

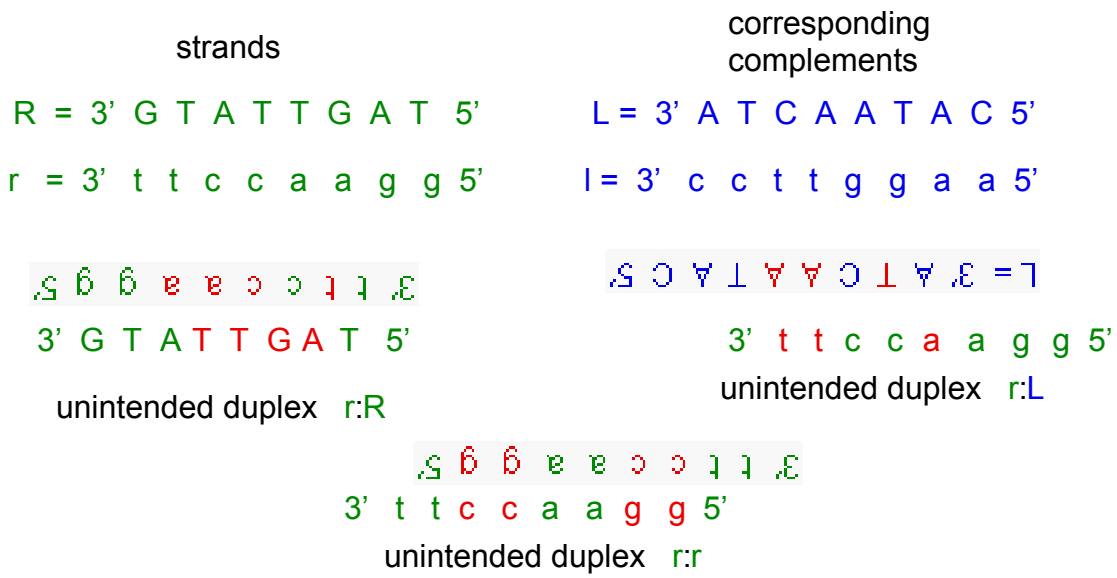

FIGURE 3

To deal with the problem of unintended duplexes, each intended duplex should be much more energetically stable than any possible unintended duplex.

In this way, if the hybridization assays are conducted at a temperature above a certain threshold, then only intended duplexes can form. The main open question is how to best mathematically model this strand design problem.

The design problem for a DNA(n,d) code presents a trade-off. In order to maximize the amount of information that can be stored or processed in parallel, it is desirable to have as many strands as possible. On the other hand, if too many strands are used, similar strands will entail cross-hybridization, reducing the accuracy of the assay. Thus the DNA(n,d) code must be constructed to adhere to some constraints

This research improved known constructions of DNA(n,d) codes and demonstrated how to use them as universal components in DNA based computing.

3. Methods, Assumptions, Procedures

3.1 Insertion-Deletion Codes

In this work we think of the strand design problem as a mathematical coding theory problem and we use the insertion-deletion metric as our constraint.

While other "metric" constraints have been applied to this problem [7], the insertion-deletion metric is the only one that models constraining the absolute maximum number of possible cross-hybridized (i.e., bad) base pairings. Also the insertion-deletion is the only metric that can be adapted to constraining the absolute maximum number of possible cross-hybridized (i.e., bad) hydrogen bounds.

All lower-case Roman variables represent non-negative integers. [n] denotes the set $\{1, 2, ..., n\}$. A q-ary n-sequence $\mathbf{x} = (x_i)$ is sequence of length n with entries $x_i \in \{0,...,q-1\}$ for each $1 \le i \le n$. For applications to our strand design problem we have q = 4.

Definition 1. Given two q-ary n-sequences **x** and **y**, $\mathrm{lcs}(\mathbf{x},\mathbf{y})$ is the maximum length of all subsequences common to both **x** and **y**. *The insertion-deletion* (or *Levenshtien*) metric is denoted by $\rho(\mathbf{x},\mathbf{y})$ where $\rho(\mathbf{x},\mathbf{y}) = n - \mathrm{lcs}(\mathbf{x},\mathbf{y})$. A collection C of q-ary n-sequences is called an $\mathrm{ID}_q(n,d)$code, if for any $\mathbf{x},\mathbf{y} \in C$, we have that $\rho(\mathbf{x},\mathbf{y}) \ge d+1$. Let $\mathrm{id}_q(n,d)$be the maximum size of an $\mathrm{ID}_q(n,d)$ code.

Two codewords **x** and **y** are at insertion-deletion distance at least d+1 if and only if their longest common subsequence has length at most n-d-1. This means that if up to d deletions (of sequence entries) are made in any codeword **x**, the resulting (and shorter) q-ary sequence could not have been obtained by deleting up to d entries in any other codeword **y** with $\mathbf{y} \ne \mathbf{x}$. For this reason $\mathrm{ID}_q(n,d)$ codes can be thought of as *d-deletion correcting codes*[1]. See Example 1.

Example 1. Consider the 4-ary sequences **x**, **y** of length 8 with **x** = 31011301 and **y** = 22113300. Then $\rho(\mathbf{x},\mathbf{y}) = 4$ because $\mathrm{lcs}(\mathbf{x},\mathbf{y})$ = 4. A common subsequence of length four is indicated by red entries.

[1] Actually such a code can correct any combination of up to d deletions and\or insertions. Hence the name insertion-deletion metric.

Note that by starting with **x** and deleting the blue entries, we have the common sequence 1130. Then by inserting the entries 2, 2, 3, 0 in the appropriate places, we obtain **y**.

There are a couple of basic formulas that form the basis for the theory of ID(n,d) codes. Let **x** be a q-ary n-sequence. Let $D_t(\mathbf{x})$ be the set of all q-ary (n-t)-sequences that can be obtained from **x** by t deletions. Let $I_t(\mathbf{x})$ be the set of all q-ary (n+t)-sequences that can be obtained from **x** by t insertions. Let $V_t(\mathbf{x})$ be the set of all codewords with insertion-deletion distance at most t from **x**. $V_t(\mathbf{x})$ is the insertion-deletion sphere of radius t and centered at **x**. Let r(**x**) be the number of runs of **x**. A run is a maximum interval of **x** that consists of the same symbol, e.g., 111211000033 has five runs. From [20], [23], [24], we have that: $\sum_{i=0}^{t}\binom{r(\mathbf{x})-t}{i} \le |D_t(\mathbf{x})| \le \binom{r(\mathbf{x})+t-1}{t}$ and $|I_t(\mathbf{x})| = \sum_{i=0}^{t}\binom{n+t}{i}(q-1)^i$.

These equations give information about bounds for $ID_q(n,d)$ code. Suppose we have an $ID_q(n,d)$ code C. Then for $\mathbf{x},\mathbf{y} \in C$, there is no common subsequence of length n-d. Hence $D_d(\mathbf{x})$ and $D_d(\mathbf{y})$ are disjoint. Therefore $\sum_{\mathbf{x}\in C} D_d(\mathbf{x}) \le q^{n-d}$. Known upper bounds are essentially derived from this observation. Asymptotic lower bounds have been obtained by standard random coding methods based on an ensemble generated by the uniform distribution for q-ary n-sequences [24]. What makes $ID_q(n,d)$ (and DNA(n,d) codes below) difficult to analyze can be observed by the following. Since $\mathrm{lcs}(\mathbf{x},\mathbf{y}) \ge n-t$ if and only if **y** can be obtained from **x** by t deletions followed by t insertions, it follows that $V_t(\mathbf{x}) = \bigcup_{\mathbf{z}\in D_t(\mathbf{x})} I_t(\mathbf{z})$.

This indicates the fact that there are spheres of the same radius but different volumes. From the above observation about insertion-deletion spheres, good codes should have codewords **x** with $V_t(\mathbf{x})$ of smaller volume. This requires codewords **x** with fewer runs. One way to create such codewords is to use a stationary Markov Chain.

In comparison to the plethora of codes for the Hamming metric, there are a few known constructions of non-random $ID_q(n,d)$ codes and almost all are for $ID_q(n,1)$ codes. See [19], [25], [36], [40], [42]. One non-random method that we have discovered (and have applied in some of our programs described in Section 4) can be explained as follows. Given an $ID_q(n,d)$ code C of size N an $ID_q(nk, kd+k-1)$ code C(k) of size N can be obtained by replacing every n-sequence in C with the nk-sequence achieved by repeating each entry k times. For example if k = 3, then 0112033 would be replaced by 000111111222000333333.

The reason for choosing the insertion-deletion metric for strand design is that two 3'-5' DNA sequences **x**, **y** of length n can form s bonds in a duplex only if **x** and the complement of **y** have a common subsequence of length s. No other metric constraint has this property.

This is exhibited in Figure 3. The unintended duplex r:R can form four base pairings because l has the subsequence ttga and R has the exact same subsequence TTGA. Recall that l is the complement of r. Note these sequences are not necessarily contiguous. For example, the subsequence ttga of l is not contiguous. With this in mind, we have the following definition.

3.2 DNA(n,d) Codes

Definition 2 [15] A DNA(n,d) code is a collection of DNA strands (3'-5') of length n that satisfy the following constraints:

1. The complement of every strand in the collection is also in the collection (i.e., the code is closed under complementation.)
2. No strand is equal to its complement.
3. The longest common subsequence between any two strands in the collection is at most n-d-1.

Ideally, strands from an DNA(n,d) code form n bonds with their complement strands in the formation of intended hybridized duplexes, while at most n-d-1 bonds occur in any unintended cross-hybridized duplexes. Thus if d is large enough and the reactions are carried out at a temperature that exceeds the n-d-1 bonding threshold, but is below the n bonding threshold, then cross-hybridization will be essentially eliminated.

If we set 0 = A, 1 = T, 2 = C, and 3 = G, then DNA(n,d) codes are a subclass of $\mathrm{ID}_4(n,d)$ codes because an $\mathrm{ID}_4(n,d)$ only satisfies condition 3 of a DNA(n,d) code. The reasons for conditions 1 and 2 can be observed in FIGURES 6-9, and are discussed Section 5. Using this numeric-letter identification, the reverse complement of a 4-ary sequence is defined exactly as the Watson-Crick complement. Using this definition, an $\mathrm{ID}_4(n,d)$ that also satisfies condition 1 of a DNA(n,d) is called a RC(n, d) code.

Asymptotic lower bounds have been established for RC(n,d) codes have been obtained. Let rc(n,d) be the maximum size of a RC(n,d) code. Clearly rc(n,d) ≤ $\mathrm{id}_4(n,d)$. Then as $n \to \infty$ we have

$$\mathrm{rc}(n,d) \geq (d!)^2 \left(\frac{q}{(q-1)^2}\right)^d \frac{q^n}{n^{2d}} (1+o(1)). \quad (1)$$

As was the case for $ID_q(n,d)$ codes, this lower bound by was achieved by standard random coding arguments based on an ensemble of codewords generated by using the uniform distribution of the q-ary symbols.

Let C be a DNA(n,d) of size 2N. We can partition C into two halves, each half free of the complement of any other strand in the given half. Let $R(C) = \{r_1,...,r_N\}$ denote one of these halves and let $L(C) = \{l_1,...,l_N\}$ (where r_i is the complement of l_i) denote the other half. An example of a DNA(8,3) partition in this way is given below in TABLE 1. Note, for any **x**, **y** in this DNA(8,3) we have $lcs(\mathbf{x},\mathbf{y}) = 4$. In TABLE 1 all sequences are given 3'-5'.

In applications the subsets R(C) and L(C) have complementary functions. For example, the strands in R(C) can function as molecular tags or sites to write on a molecule while the strands in L(C) can function as probes, extractors or site blockers. In Section 5, we describe how the strands in a DNA(n,d) code can be used to construct a biomolecular computing architecture. In that architecture, each R(C) and L(C) each have two distinct functions at different points in the procedure. At one point of the procedure, r_i is used to write "1" and l_i is used to write "0." At another point, r_i is used to read "0" and l_i is used to read "1." In FIGURES 6-9, we also indicate why the insertion-deletion distance needs to be obtained for codewords between and inside each of the halves.

R(C)		L(C)	
r_1 = aattttaa	r_7 = atgcgttg	l_1 = ttaaaatt	l_7 = caacgcat
r_2 = taaccccg	r_8 = cccccccc	l_2 = cggggtta	l_8 = gggggggg
r_3 = ttccaagg	r_9 = aaaaaaaa	l_3 = ccttggaa	l_9 = tttttttt
r_4 = ggccaatt	r_{10} = ggtttccc	l_4 = aattggcc	l_{10} = gggaaacc
r_5 = gctacggg	r_{11} = ccccttttt	l_5 = cccgtagc	l_{11} = aaaagggg
r_6 = gtattgat	r_{12} = ttttgggg	l_6 = atcaatac	l_{12} = ccccttttt

TABLE 1

4. Results, Discussion

4.1 DNA(n,d) Code Generation Programs

We have programs that generate very good random and pseudo-random DNA(n,d) ($ID_q(n,d)$) codes. One reason that we believe we can construct better bounds for $ID_q(n,d)$ and DNA(n,d) codes with a Markov chain approach stems from the great improvement that we achieved by using a Markov

chain to generate codewords for our randomly constructed codes. In [9], a random ID(20,5) of size 1024 was generated by a using a uniform distribution. This code was pruned to a DNA(20,5) code of size 16 by experimental and computational methods.

On a 2.3 ghtz Pentium PC, we generated a DNA(20,5) code of size 3038!

Given two n-sequences **x**, **y**, all of our programs use a "folklore" dynamic programming algorithm to find the $\mathrm{lcs}(\mathbf{x},\mathbf{y})$. This is essentially described in [10]. The complexity of this subroutine is $O(n^2)$. In [10], an improvement of the "folklore" algorithm is given and we plan to incorporate this in our future programs.

Our very first program used a (reverse-)complement cyclic code as an initial set from which to find a DNA(n,d) code as a subcode. However, we eventually realized that cyclic codes have too much symmetry to generate large-size DNA subcodes.

The next step was based on generating uniformly distributed independent random n-sequences, but from the discussion in Section 4, one can understand that the volumes of the spheres of a fixed radius centered at many of these n-sequences is too large. We did not get the desired performance here.

Our most recent programs generate candidate n-sequences **x** in the following way. The value of x_1 is selected from {a,c,g,t} with uniform probability. Then, the remaining entries are generated by a stationary Markov chain given by transition matrix M_k with parameter k.

$$M_k = \begin{array}{c} \\ a \\ c \\ g \\ t \end{array} \begin{array}{c} \begin{array}{cccc} a & c & g & t \end{array} \\ \begin{pmatrix} \frac{k-3}{k} & k^{-1} & k^{-1} & k^{-1} \\ k^{-1} & \frac{k-3}{k} & k^{-1} & k^{-1} \\ k^{-1} & k^{-1} & \frac{k-3}{k} & k^{-1} \\ k^{-1} & k^{-1} & k^{-1} & \frac{k-3}{k} \end{pmatrix} \end{array}$$

FIGURE 4

The average number[2] of runs in the codewords of this ensemble is $\frac{3n}{k}$. Thus higher values of k give sequences with fewer runs. These sequences have low volume spheres of the desired radius.

---

2 $\frac{(q-1)k}{n}$ for $q \neq 4$.

However, the higher the k, the fewer the number of sequences generated. At present, our programs take a dynamic heuristic approach. These programs start a high value of k and then check all values of the permitted length of the longest common subsequence. This continues for a set number of cycles over which no new codeword is added. The next value of k is set by finding the next highest value of k for which a codeword can be added to the growing code. Here dichotomy is used. This heuristic has worked very well and is much better than the uniform codeword generation method. For example, by using our Markov chain heuristic, we can generate DNA(15,5) code of size 104. This code is exhibited in TABLE 2 (only R(C) is given, L(C) follows.) Using the uniform distribution method, we could only generate DNA(15,5) code of size 14.

R(C)

cccccccccccccc	aaccccggattttta	cctttaaggatttcc	tttgggccccagcct
tttttttttttttt	ttggggaaaaacccc	cttttcccgtaactc	aagtaaggtagcagg
aaaaaaggggggggc	ttttggaaaaatttt	ggtttcccttcggtt	gccgtgggctggaac
gggcccttaaaaaaa	ggggaaaaaggttgg	ttccaaaattaaacc	tctgcaaacaagcag
ttttttccccggggg	ggaatttggggttcc	aaagggggctttacg	gtcctttgtcgcctg
ggggggaaatttttt	ttcgggggcccggg	aaacaactttgggca	tgcctcccgcgattg
aaaaaaaaccctttt	ccaaaaaccccgaaa	ttagtttgaagcttg	acaatcgtatcccga
tttacccccccaaaa	cataattttggccgg	gggtggattcaagca	attactggctggcat
aaaggggcccccccc	acccctttaacctgg	cccattcggccaaca	agaattccatacctt
cccccctttggggcc	acaaaaaaggtcaat	acaggccagtccggg	ttggtcgtctttcac
tttaaaaaaaaaggg	cgggccagggtttaa	ttccaaggggtccaa	gacgacccgataggt
aaagggccggaaaat	cccccaaaaggccct	ggactttagtcaatt	tttgatgggactacg
aatcccccccagggt	gttccgttttaaaat	cgtcggttaggcccc	gagcggtcggtactt

TABLE 2

5. Conclusions

5.1 DNA Computing with DNA(n,d) Codes

To give an example of how DNA(n,d) codes can be applied to biomolecular computing, we discuss algorithm and architecture in [6]. In [6] a total of 80 distinct strands (40 library encoding, 40 filtering) were used to solve a 20 variable SAT problem. We show that a DNA(n,d) of size can be used to

encode *and* filter. In [6], a strand either encodes *or* filters exclusively. Note, there are other architectures that can be constructed using DNA(n,d) codes ,e.g., the variants of the sticker method [33].

Note that because of the (assumed) properties of our DNA(n,d) code, we need only half as many distinct strands as were used in [6].

A DNA bit string of length N is a DNA molecule (single long strand) that consists of N distinct non-overlapping substrands $X_1, X_2, ..., X_N$ and N-1 identical DNA molecules S that are located between any two consecutive X_i, X_{i+1} in the DNA bit string[3]. Suppose we have a DNA(n,d) code C of size 2N partitioned into R(C) and L(C). A Lipton encoding [26] can be used to construct a DNA library of 2^N distinct DNA bit strings $X_1SX_2S...X_{n-1}SX_n$ were $X_i = r_i$ or $X_i = l_i$ for $r_i \in R(C)$, $l_i \in L(C)$. If we think of $r_i = 1$ and $l_i = 0$, then we have a library of DNA molecules that encode all binary sequences of length N. Using a subcode $\{r_1, ..., r_7\} \cup \{l_1, ..., l_7\}$ of the above DNA(8,3) code in TABLE 1, the encoding of the sequence 1000101 is given in FIGURE 5.

3' aattttaa S cggggtta S ccttggaa S aattggcc S gctacggg S atcaatac S atgcgttg 5'

FIGURE 5

As indicated above, we identify DNA bit strings and binary sequences. For $I \subseteq [N]$ and $(e_i)_{i \in I}$ a binary sequence, let K be the following a subset of binary N-sequences defined as $K = \{(b_i) : b_i = e_i \text{ for some } i \in I\}$. K is the set of all binary sequences that satisfy the disjunctive clause K' over N Boolean terms, each of which is a variable x_i (if $e_i = 1$) or its negation $\sim x_i$ (if $e_i = 0$.) The main "computing" idea in [6] is an iteration of the following: Given a subset T of DNA bit strings and a set K defined above, the subset $T \cap K$ can be extracted from the set T by hybridization. See Example 2.

Example 2. Suppose T is the set of all 2^7 DNA bits stings formed by using our DNA(8,3) subcode of size 14 given above. Suppose $K_1 = \{(b_i) : b_1 = 1 \text{ or } b_3 = 0 \text{ or } b_4 = 1 \text{ or } b_5 = 0\}$ and $K_2 = \{(b_i) : b_2 = 1 \text{ or } b_3 = 1 \text{ or } b_7 = 0\}$. If we use the DNA bit string representation, then $K_1 = \{(b_i) : b_1 = r_1 \text{ or } b_3 = l_3 \text{ or } b_4 = r_4 \text{ or } b_5 = l_5\}$ and $K_2 = \{(b_i) : b_2 = r_2 \text{ or } b_3 = r_3 \text{ or } b_7 = l_7\}$.

[3] To improve performance a strands of synthetic bases (e.g., iso-G) could be used as separator sequences.

Two corresponding "filters" F_1 and F_2 are constructed. F_1 consists of the *probe strands* l_1, r_3, l_4, r_5 affixed to a gel. Note that these are the complement stands to those that appear in K_1. Thus F_1 could be called the *complement filter* of K_1. Similarly, F_2 consists of the strands l_2, l_3, r_7 affixed to a gel. When T is passed through F_1, only the strands in K_1 hybridize with the probes affixed in F_1 and remain in the gel. The strands that pass through the filter F_1 are discarded. The strands that remain in the F_1 gel are exactly $T \cap K_1$. These strands can be "washed" from the filter F_1 and recovered. Then these recovered strands are passed through filter F_2. Only the strands in $T \cap K_1$ and in K_2 hybridize with probes in F_2. What passes through is discarded. The strands that remain in the F_2 gel are exactly $T \cap K_1 \cap K_2$. Thus the strands $T \cap K_1 \cap K_2$ in the F_2 gel are all binary sequences that satisfy the conjunction $K_1 \wedge K_2$ of the clauses K_1 and K_2.

Given the above descriptions, the general SAT problem can be thought of as: Given disjunctive clauses $K_1, K_2, \ldots, K_p$, then is $\bigcap_{i=1}^{p} K_i \neq \varnothing$? By constructing the corresponding complement filters $F_1, F_2, \ldots, F_p$ and iterating the above process, the answer is "yes" if and only if there are any strands in F_p.

All of the above analysis is contingent on avoiding all of the possible cross-hybridization situations that a DNA(n,d) code intends to avoid. We now give some examples of potential cross-hybridizations. For the filter to work, we need correct reads. See FIGURE 6.

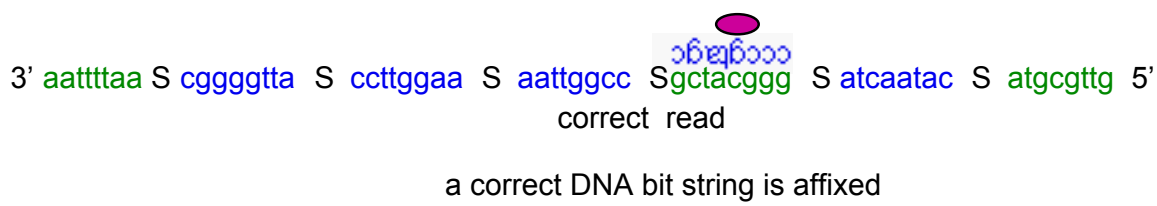

FIGURE 6

Incorrect reads are avoided by ensuring that codewords inside of L(C) (R(C)) have the proper insertion-deletion distance. In FIGURE 7, only four base pairings can form in a "bad read." Here c_5 is

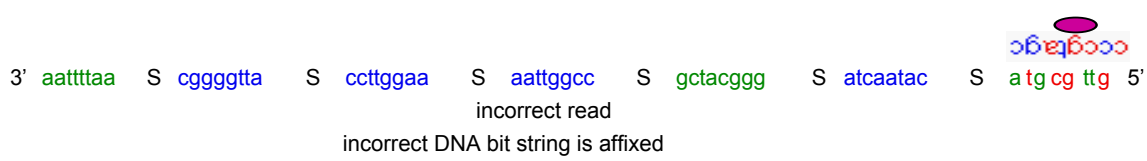

FIGURE 7

"incorrectly reading" r_7. Four bonds can form because $\mathrm{lcs}(r_5, r_7) = 4$. The common subsequence between r_5= gctacgg and r_7=atgcgttg is tcgg.

Inter-DNA bit string interactions are prevented by ensuring that the insertion-deletion distance inside of R(C) (L(C)) or between R(C) and L(C) is sufficient. In top pair in FIGURE 8, only four bonds can form between the two strands at the indicated positions because $\mathrm{lcs}(l_2, r_5) = 4$. The common subsequence between l_2 = cggggtta and r_5= gctacggg" in Figure 8 is gggg. Similarly ensuring the proper insertion-deletion distance prevents intra-DNA (hairpin) interactions. See FIGURE 9.

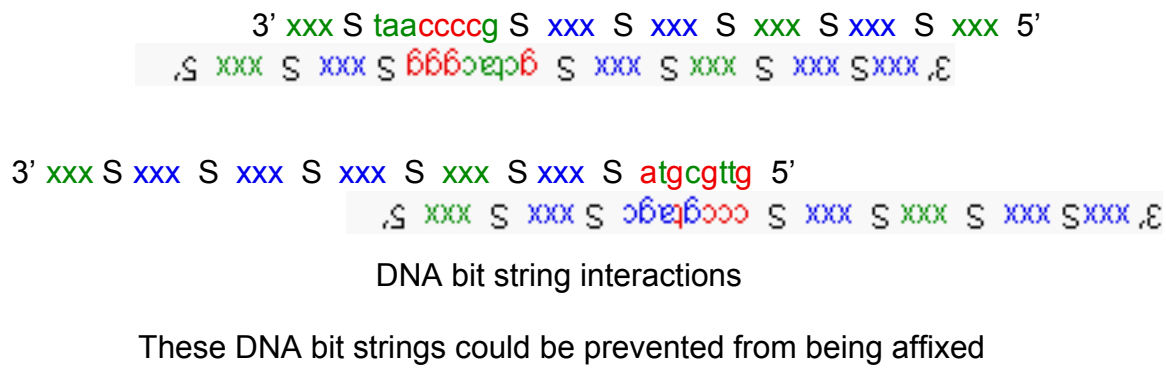

FIGURE 8

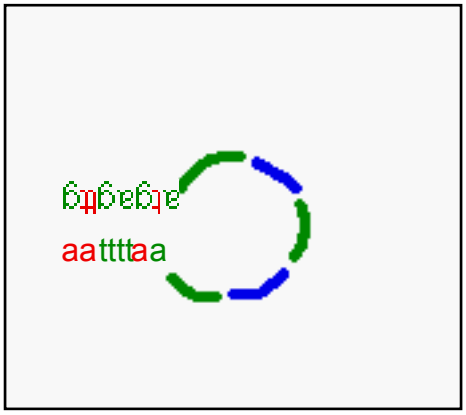

FIGURE 9

5.2 DNA Computing and Data Mining

We now discuss a problem that is of particular interest to us. Let $[2^N]$ denote the power set of [N]. Suppose we have DNA(n,d) code C of size 2N partitioned into R(C) and L(C).

Problem 1. Let $P_1, P_2, ..., P_m$ be fixed subsets of [N].

a. Find all $S \subset [N]$ with $S \not\subset P_i$ for all i with $1 \le i \le m$.
b. Find all $T \subset [N]$ with $P_i \not\subset T$ for all i with $1 \le i \le m$.

Both of these problems are related and are simplified forms of the general SAT problem. They can be solved by the method described above. (These are simplifications because no negations appear in the clauses.)

There is one important difference. In the SAT problem, only one solution needs to be found. Here all solutions are required.

Let (b_i) be a binary n-sequence. As above, let $K_i = \{(b_j) : b_j = 1 \text{ for some } j \notin P_i\}$. Clearly all $S \not\subset P_i$ for all i with $1 \le i \le m$ is the set of all S with incidence vector in $\bigcap_{i=1}^{m} K_i$. In the DNA bit string representation, $K_i = \{(b_j) : b_j = r_j \text{ for some } j \notin P_i\}$. The associated filter F_i consists of $\{l_j : j \notin P_i\}$. If a set S of DNA bit strings of length N is passed through F_i, then only the bit strings in K_i remain in the gel of F_i. Starting with all possible DNA bit strings and iterating the filter process outlined above m times, we arrive at F_m. F_m contains all the DNA bit string representations of the solutions to Problem 1a. Problem 1b can transformed into Problem 1a because $P_i \not\subset T$ if and only if $[N] - T \not\subset [N] - P_i$.

The most straightforward application of the above problem is in the identification of *independent sets* in a graph (or hypergraph). In Problem 1b, if one takes all the edges of a simple graph G as the collection $\{P_i\}$, then the set of all T is the collection of independent sets in G.

Problem 1a can be applied to the identification of maximal frequent sets in a data base [12], [18], [27], [28], [29], [30]. The reason that this is of interest to us is because the identification of the maximal frequent sets is the main computational bottleneck in the data mining of association rules.

The relationship between data mining and Problem 1 is this. If the sets $\{P_i\}$ are selected properly, the subsets S will be candidates for maximal frequent sets. See [12], [18], [27], [28], [29], [30] and Section 12 here. The application of DNA computing is to apply to Problem 1 an algorithm like that described in Section 7.

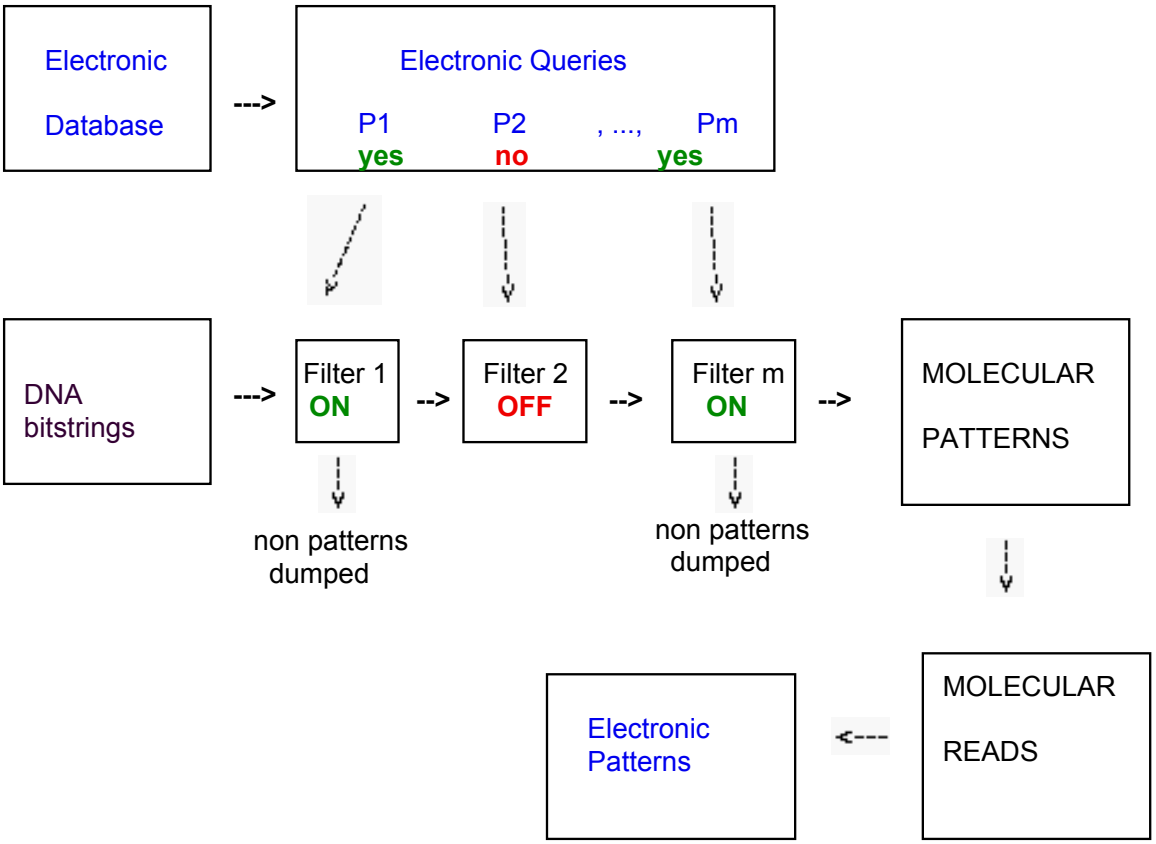

FIGURE 10

The relationship between data mining and Problem 1 is this. If the sets $\{P_i\}$ are selected properly the subsets S will be candidates for maximal frequent sets. See [12], [18], [27], [28], [29], [30]. The application of DNA computing is to apply an algorithm like that described in Section 7 to Problem 1.

Then the resulting collection of DNA library strands that remain in the final filter will code for maximal frequent sets. What is envisioned is that patterns in data are being transformed into molecules. See FIGURE 10. If we can read the molecules, then that we have found the patterns..

6. References

1. Adleman, L., Molecular computation of solutions to combinatorial problems, Science} 266, 1021-1024, (1994).

2. Baum, E. DNA sequences useful for computation, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 44, 235-242, (1999).

3. BenDor, A., Karp, R., Schwikowski, B., and Yakhani, Z., Universal DNA tag systems: A combinatorial scheme, J. Comp. Biol., 7, 503-519, (2000).

4. Brenner, S. Methods for sorting polynucleotides using oligonucleotide tags, U.S. Patent No. 5,604,097, 1997

5. Brenner, S. et al., Gene expression analysis by massively parallel signature sequencing (MPSS) on microbead arrarys, Nat. Biotechnol., 18, 630-634 (2000).

6. Braich, R., Chelyapov, N., Johnson, C., Rothemund, P.W.K., Adleman, L. *Solution of a 20-Variable 3-SAT Problem on a DNA Computer*. Sciencexpress, 1-15, (2002).

7. Brenneman, A. and Condon, A. Strand Design for biomolecular computation, Theoretical Computer Science, 287, 39-58, (2002).

8. Breslauer, K., R. Frank, H. Blocker, and L.A. Markey, Predicting Duplex DNA Stability from the Base Sequence, PNAS 83, 3746-3750, (1986).

9. Cai, H., P. White, D. Torney, A. Deshpande, Z. Wang, B. Marrone, and J. Nolan, Flow Cytometry-Based Minisequencing: A New Platform for High Throughput Single Nucleotide Polymorphism Scoring, Genomics, 66, 135-143, (2000).

10. Crochemore, M., Iliopoulos, C., Pinzon, Y., and Reid, J., A fast and practical bit-vector algorithm for the longest common subsequence, Information Processing Letters, 80, 279-285, (2001).

11. Delgado, A, DNA chips as look up tables for rule based systems, Computing and Control Engineering Journal, 113-119, June (2002).

12. Dyachkov, A., A. Macula, P. Vilenkin and D. Torney On families of subsets where the intersection of l subsets are not covered by the union of s others, , J. Combinatorial Th. Ser. A, to appear

13. D'yachkov, A., and D. Torney, On Similarity Codes, IEEE Trans. on Information Theory}, 46, 1558-1564, (2000).

14. D'yachkov, A., D. Torney, P. Vilenkin, and P. White, On a Class of Codes for Insertion-Deletion Metric, 2002 IEEE International Symposium on Information Theory, Lausanne, Switzerland, (2002).

15. D'yachkov, A., D. Torney, P. Vilenkin, and P. White, Reverse-Complement Similarity Codes, IEEE Trans. on Information Theory submitted.

16. Erdos, P.L., D. Torney, and P. Sziklai, A Finite Word Poset, Elec. J. of Combinatorics, 8, (2001).

17. Faulhammer, D., Cukras, A., Lipton, R., and Landweber, L., Molecular computation, RNA solutions to chess problems, PNAS, 97, no.4, 1385-1389, (2000).

18. Flodman , P., A. Macula, A. Spence and D. Torney , A new data mining technique for the analysis of simulated genetic data, Proceedings of Genetic Analysis Workshop 12, Wiley-Liss, (2001).

19. G. Tenengolts, Nonbinary codes correcting single deletion or insertion, IEEE Trans. Inform. Theory, 30, 766-769, (1984).

20. Hirschberg, D., Bound on the number of string subsequences, in Proc. 10th Symp. on Combinatorial Pattern Matching, Warwick UK , Lecture Notes in Computer Science, Springer-Verlag, Berlin, (1999).

21. Hollman, H., A relation between Levenshtein-type distances and insertion and deletion correcting capabilities of codes, IEEE Trans. on Information Theory, 39 1424-1427, (1993).

22. Kaderali, L, Selecting Target Specific Probes for DNA Arrays, Master's Thesis, Informatics, U. Koln, (2001).

23. Levenshtein, V., Binary Codes Capable of Correcting Deletions, Insertions, and Reversals, Soviet Phys.--Doklady, 10 707-710, (1966).

24. Levenshtein, V., Bounds for Deletion-Insertion Correcting Codes, 2002 IEEE International Symposium on Information Theory, Lausanne, Switzerland, (2002).

25. Levenshtein, V., Elements of Coding Theory, in the book: Discrete Mathematics and Mathematical Problems of Cybernetics, Moscow, Nauka, pp.207-305, (1974) (in Russian).

26. Lipton, R., DNA solution of hard computational Problems, Science, 268, 542-545,

27. Macula, A., A Combinatorial Profiling Model for Intrusion Detection and Analysis, IEEE Conference on Information Assurance and Security, West Point, NY, 47-52, (2000)

28. Macula, A., and L. Popyack, A group testing method for finding patterns in data, with L. Popyack, Proc. SIAM 2002 Conference on Data Mining and Discrete Mathematics, to appear

29. Macula, A., and Rykov, Two-stage group testing for complexes using almost disjunct matrices, Discrete Applied Math, to appear

30. Macula, A., D. Torney and P. Vilenkin, Two stage group testing for complexes in the presence of errors, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 55, p 145-158, (2000).

31. Marathe, A., A. Condon, and R. Corn, On combinatorial word design, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 54, 75-89, (2000).

32. Needleman, S., and C.D. Wunsch, A General Method Applicable to the Search for Similarities in the Amino-Acid Sequences of Two Proteins, J. Mol. Biol. 48, 443-453 (1970).

33. Roweis, S., Winfree, E., Burgoyne, R., Chelyapov, N.V., Goodman, M.F., Rothemund, P.W.K., Adleman, L.M. *A Sticker Based Model for DNA Computation.* DIMACS Series in Discrete Mathematics and Theoretical Computer Science, *44*, 1-29, (1999).

34. Rykov, V., A. Macula, C. Korzelius, D. Englehart, D. Torney, and P. White, DNA Sequences Constructed on the Basis of Quaternary Cyclic Codes, Proceedings of the 4th World Multiconference on Systematics, Cybernetics, and Informatics, SCI 2000/ISAS2000, Orlando, Florida, July 2000.

35. Sakamoto, K., H, Gouzu, K. Komiya, D. Kiga, S. Yokoyama, T. Yokomori, and M. Hagiya, Molecular Computation by DNA Hairpin Formation, Science, 288, 1223-1226, (2000).

36. Sloane, N., On Single-Deletion-Correcting Codes, IEEE Trans. Inform. Theory, (2002).

37. Solas, D., C. Pease, E. Sullivan, M. Cronin, C. Holmes, and S. Fodor, Oligonucleotide arrays for rapid DNA sequence analysis, PNAS, 91, 5022-5026, (1994).

38. Smith, T. and M. Waterman, Identification of Common Molecular Subsequences, J. Mol. Biol, 147, 195-197 (1981).

39. Ullman, J., On the capabilities of codes to correct synchronization errors, IEEE IT , 13, 95-105 (1967).

40. Varusamov, R., and G. Tenengol'ts, One asymmetrical error correction codes, Avtomatika I Telemekhanika, 26, 288-292, (1965) (in Russian).

41. Winfree, E., F. Lui, L. Wenzler, N. Seeman, Design and self-assembly of 2D DNA cystrals, Nature 394, 539-544, (1998).

42. Yin, J., Directed Designs and Perfect Deletion-Correcting Codes, Designs, Codes, and Cryptography, 99-110, (2000).