Chapter III

DNA Computing and Errors:
A Computer Science Perspective

Lila Kari, The University of Western Ontario, Canada

Elena Losseva, The University of Western Ontario, Canada

Petr Sosík,
Silesian University, Czech Republic and
The University of Western Ontario, Canada

ABSTRACT

This chapter looks at the question of managing errors that arise in DNA-based computation. Due to the inaccuracy of biochemical reactions, the experimental implementation of a DNA computation may lead to incorrectly calculated results. This chapter explores different methods that can assist in the reduction of such occurrences. The solutions to the problem of erroneous biocomputations are presented from the perspective of computer science techniques. Three main aspects of dealing with errors are covered: software simulations, algorithmic approaches, and theoretical methods. The objective of this survey is to explain how these tools can reduce errors associated with DNA computing.

INTRODUCTION

Biomolecular computing is a field that studies biologically based computational paradigms that serve as alternatives to the traditional electronic ones. Biomolecular computing includes DNA computing (Adleman, 1994; Head, 1987), RNA computing (Faulhammer et al., 2000), peptide computing (Balan et al., 2002), and membrane computing (Păun, 2000). The main idea behind DNA computing is that data can be encoded in DNA strands and molecular biology tools can be used to perform arithmetic and logic operations.

Nearly a decade has passed since the field of DNA computing premiered on the scientific stage as the possible computational paradigm of the future. The idea attracted research from a wide spectrum of mathematical and natural sciences. However, the inherently complex nature of biological processes tempered the advancement of the field, suggesting the development of biocomputing will trail a path that is different from that of electronic computing half a century ago. It is becoming increasingly more apparent that most plausible implementations of biocomputing are likely to produce some unexpected and erroneous results. From chemical reactions in vitro that occasionally have unpredicted output, to unforeseen problems in vivo, it seems that many errors are not only inevitable but also an integral part of the biological processes. The purpose of this chapter is to provide a survey of the tools that computer scientists offer for dealing with the imminent problem of managing errors in DNA computing.

The battle for reliability of biomolecular computation and reduction of errors can be fought on several fronts. Research is conducted to find better ways to encode information in DNA, to develop more efficient algorithms, and to improve laboratory techniques, among other results. This survey does not cover the wide scope of research in chemistry, biology, physics, or engineering that contributes to dealing with errors in biomolecular computing. Instead, this exposition explores the tools that computer science offers us in managing the errors that arise in DNA computing processes.

A single strand of DNA (deoxyribonucleic acid) is a molecule made of a sequence of nucleotides, also called bases. Four types of nucleotides are present in DNA, called adenine, guanine, cytosine, and thymine. These are abbreviated as *A*, *G*, *C*, and *T* respectively. A single strand of DNA is held together by covalent bonds that keep the bases linearly attached to each other. In addition, it is possible for hydrogen bonds to form between the *A* and *T* bases, as well as between *C* and *G* bases of two different strands. This property is referred to as the *complementarity* of nucleotides — that is, *A* and *T* are said to be complementary, and so are the *C* and *G* bases. Bonds between

complementary nucleotides are called *base-pair* bonds. Unlike beads on a string, a sequence of nucleotides is distinct from its reverse. This property of a DNA strand is called the *polarity* of a strand, and it imposes a distinction on the two ends of the DNA molecule. The two ends of the strand are called the *3'-end* and the *5'-end*. Whenever the nucleotides of two sequences are complementary and the strands have opposite polarities (orientation in space), the strands will anneal (hybridize) to form a double helix. Sequences with this property are also called "Watson-Crick complementary" in honour of the two scientists who discovered the structure of DNA. See Watson et al. (1987) for further information on molecular biology.

Hybridization is one of the fundamental mechanisms used in DNA-based computing. Using hybridization, along with other biochemical operations, potentially general-purpose computations can be carried out (Freund et al., 1999). However, many of the designed experiments fail to produce the anticipated computational answer. In this chapter, the discussion of errors in DNA computing carries a computational connotation and refers to events that lead to obtaining a computationally incorrect result. The source of errors leading to incorrect answers can be anything from an inappropriate choice of encodings of information into DNA strands to unsuitable experimental conditions. Here we examine reducing the effect of such errors on the correctness of the computed answer.

This chapter addresses the various aspects of managing errors in three main sections examining software simulation, algorithmic, and theoretical approaches. The software simulation tools described in the first section can accomplish such activities as testing computation protocols. This testing verifies protocol correctness before it is carried out in a laboratory experiment, thus detecting potential errors. Certain errors in DNA computation can be avoided by designing strands that prevent the formation of DNA secondary structures (intramolecular bonds). Algorithms for constructing DNA sequences with this property are mentioned in the second section of the chapter. Finally, the last section gives an overview of theoretical methods aimed at reducing errors caused by undesirable hybridization. This includes template-based sequence design of code words and a study of bond-free DNA languages, followed by a discussion of future trends and research directions in the area.

More precisely, the software section of this paper looks at three different programs: BIND, SCAN, and Edna. This is not a complete list of programs written for DNA computing purposes, but it provides an overview of the types of problems that can be successfully addressed with software. The BIND program's main focus is to estimate DNA hybridization temperatures. Hybrid-

ization is a reaction present in all DNA computing protocols. Understanding under what conditions hybridization occurs involves knowing the hybridization temperature of each reaction. Carrying out laboratory experiments at inappropriate temperatures is a common source of erroneous results. Prediction of hybridization temperatures by BIND helps to avoid such problems.

A short overview of thermodynamics of DNA hybridization is also included in this section. The SCAN program is designed to find DNA sequences for computation that meet a required set of constraints. These constraints include, for example, the property of strands that avoid formation of secondary structures. Another constraint is that computation rules, encoded in DNA sequences, should not interfere with each other. If these constraints are not met, the computation is likely to result in errors. Finally, the simulation software Edna can test DNA-based algorithms for possible errors. Edna simulates biochemical processes and reactions that can occur during a laboratory experiment. Testing laboratory protocols with Edna before the experimental implementation is conducted can avoid many errors.

Another avenue of research aimed at reducing errors in DNA computing looks at methods of reducing the possibility secondary structures of DNA strands. In particular, this problem arises when a number of short DNA strands attach together to form long strands. While the original strands may not sick to each other in undesirable ways, the resulting strand may form bulges or loops. The structure freeness problem for combinatorial sets asks whether, given a set of DNA words, a concatenation of an arbitrary number of words from this set will form a word that leads to secondary structures. Algorithms that answer this question are based on heuristic calculations of the free energy of a DNA strand.

An entirely different approach to reducing errors related to DNA computing is offered by theoretical computer science methods. The question of developing appropriate techniques for encoding data in DNA can be studied in both the formal language theory and the coding theory frameworks. The final section of the chapter first explains a template-based design of DNA sequences, followed by an overview of the properties of DNA languages.

SOFTWARE APPROACHES

BIND Simulator

Hybridization of DNA strands is utilized in virtually all proposals for DNA computation — both experimental and theoretical ones. Hybridization, otherwise called annealing, is the process that forms a double-stranded DNA helix

from two single-stranded DNA sequences, provided that certain conditions apply. Almost all models of computing with DNA rely on an accurate prediction of whether some DNA sequences will anneal. The success of a given model, therefore, directly depends on the correctness of this prediction.

While it is not easy to determine whether two arbitrary sequences will anneal, some general principles can be considered. We can see if two single strands of DNA anneal by checking if they are Watson-Crick complementary. However, the picture of DNA hybridization is much more complex than that. The length of the sequences makes a difference. If one sequence is longer than another and the double strand has an unhybridized segment on its end — called the sticky end — the stability of the helix is also affected. The concentration of strands in the solution, the temperature at which the reaction takes place, and numerous other factors also play a role. To complicate the situation further, it is also possible for nucleotides to form bonds with nucleotides other than their complements. This situation is called a *base-pairing mismatch*. To get the full picture, new software tools are needed to predict the likelihood of hybridization of two strands.

One example of such software is BIND (Hartemink & Gifford, 1997), which uses the Nearest Neighbour Model (NNM) of annealing to describe hybridization of strands. For two DNA sequences, we can say that there exists a temperature at which half of the strands in the solution are hybridized and half are not. This temperature is called the *melting temperature* for the given DNA double helix. Melting is the opposite process of hybridization; it separates a double strand into two single strands. The melting temperature is denoted by T_M. NNM investigates thermodynamics of DNA hybridization and provides a method for calculating T_M. We shall now explain the principles of this model and how it is used by the BIND software to determine T_M.

Throughout this paper we will use a convention to write $n_1n_2\ldots n_k/m_1m_2\ldots m_k$ to denote the DNA duplex (i.e., a double-helical segment) formed by two complementary strands $5'-n_1n_2\ldots n_k-3'$ and $3'-m_1m_2\ldots m_k-5'$, where n_i and m_j are individual nucleotides. When the duplex is formed with a self-complementary strand (i.e., $n_1n_2\ldots n_k=m_km_{k-1}\ldots m_1$), the duplex is written simply as $n_1n_2\ldots n_k$.

Originally introduced by Borer et al. (1974), NNM proposes that the most significant contribution to helix stability comes from the order of nucleotides in the helix. Helices with exactly the same base-pair composition can have sufficiently different melting temperatures (SantaLucia et al., 1996). The difference in melting temperatures is attributed to the ordering of base pairs. The term *stacking interactions* is used in reference to the processes affecting

the stability of base-pair bonds as a result of neighbouring base-pairs interactions. The order in which the base pairs are stacked is a primary factor influencing duplex stability, according to NNM.

The basic idea of the model is that for short, single-stranded complementary sequences, hybridization happens like the closing of a zipper. Base-pair bonds form one by one, gradually closing the DNA "zipper". The model views hybridization of two single strands as a sequence of smaller subreactions, each one corresponding to the formation of a single base-pair bond. With this in mind, the model uses characteristics of the smaller subreactions to estimate melting temperature and other properties of the entire hybridization reaction.

To calculate the melting temperature for a strand, BIND considers the thermodynamics of DNA hybridization. A single reaction of base-pair formation is the formation of hydrogen bonds between two complementary bases. Each reaction has a number of characteristics associated with it, including enthalpy, entropy, and Gibbs free energy. Thermodynamics is a study of the interconversions of various types of energy, and since these characteristics deal with changes in the energy of the system, they are called the *thermodynamic parameters* of the model.

We now explain these thermodynamic parameters. Enthalpy change, denoted by ΔH°, of a reaction is the amount of heat released (exothermic reaction) or absorbed (endothermic reaction) by the system. Entropy is a measure of randomness or disorder. Spontaneous changes can be accompanied by either an increase or a decrease of entropy in the system. Change in entropy is denoted by ΔS°, and hybridization of strands is a process increasing the order of the system; therefore, ΔS° of hybridization reactions has a negative value (corresponding to a decrease in disorder). Gibbs free energy (or simply, free energy) describes the potential of a reaction to occur spontaneously. Each chemical reaction that converts products into reactants also happens in the reverse direction simultaneously, but at a different rate. When the rates are equal, the system is in equilibrium.

A simple example of a system in equilibrium is a bucket of water at zero degrees with some ice in it. While nothing happens visibly, there are two reactions going on — some ice is melting and some water is freezing. DNA hybridization works in a similar manner. The melting temperature of a DNA helix is defined as the temperature at which half of the DNA strands in the solution are annealed and half are not. The system is at equilibrium. Some single strands are annealing, some double strands are melting, but the rates of reaction of both forward (annealing) and reverse (melting) reactions are the same. When a system is not at equilibrium, one direction of the reaction is spontaneous; it is

the one we observe. The change in Gibbs free energy, denoted by $\Delta G°$, determines whether the system is at equilibrium or not. $\Delta G°$ is the free energy of products minus the free energy of reactants. In our case, the products are the annealed double strands and the reactants are the single double strands. For the forward direction to occur spontaneously, $\Delta G°$ has to be negative. When $\Delta G°$ is zero, the system is at equilibrium. If $\Delta G°$ is positive, then the reverse reaction (melting) will occur spontaneously.

Gibbs free energy has a close correlation to melting temperature. The stronger the bond of a DNA duplex (double-stranded segment), the higher its melting temperature and the greater the change in free energy of the hybridization reaction. Another way to say this is that a duplex folds into a structure that has the lowest free energy.

So how does the BIND software use these thermodynamic parameters to calculate melting temperature? As already mentioned, NNM used by BIND views hybridization as the closing of a zipper. The formation of each new base-pair bond is examined as a separate reaction. The complete sequence of base-pair formation reactions together makes up the hybridization process. For example, hybridization of *ATG/TAC* is viewed as the initiation reaction forming *A/T*, then forming *T/A*, and finally *G/C*. The three mini-reactions together are equivalent to the hybridization reaction. In this model, the base-pair bond formation reaction depends only on the base pair formed immediately prior to it, but not on any of the future ones. That is, the free ends of the zipper do not participate in the formation of a single base-pair bond and neither does the other, closed end of the zipper. The only factor in a new base-pair bond formation reaction is the preceding base pair next to which the new pair is stacked. That immediately preceding base pair is called the *nearest neighbour* and is the source of the model's name.

$\Delta G°$ for the hybridization of the entire duplex is calculated as the sum of $\Delta G°$ of the mini-reactions plus some extra parameters accounting for initiation of the first pair. The other two parameters are calculated similarly. BIND calculates the melting temperature T_M as follows:

$$T_M = \frac{\Delta H°}{\Delta S° + R \ln([C_T]/4)}$$

In this equation, R is the Boltzmann's constant and C_T is strand concentration. If [Na^+] concentrations differ from 1M, there is an adjustment term, which is added to the equation.

The BIND program can help test any computational protocol that is based on site-specific annealing. For example, it has been used to verify a sticker-based model of DNA computation described by Roweis et al. (1996). The model design involved a long template sequence and five short DNA sequences that are supposed to anneal to the template at specific locations. These locations are exactly complementary to the sticker sequences. BIND successfully verified that the sticker sequences would not be able to incorrectly anneal at any other template location. The program predicted melting temperatures for the sticker sequences and produced a temperature range at which no erroneous binding could occur.

The SCAN Program

One of the common problems with designing DNA-based computational components is to select those sequences that are best suited for computation and yield the most reliable results. The SCAN program (Hartemink et al., 1999) assists in this task by scanning a vast space of possible designs and selecting those that meet a large set of constraints.

Consider the design of a unary counter, as proposed by Hartemink et al. (1999). The design is based on the technique of programmed mutagenesis, which is a systematic rewriting of a DNA sequence based on a set of rules. The rewriting is sequential, with rules devised in a way that allows all rules to be present in the solution. At a given step in the computation, only the valid rules can enter into a reaction. The unary counter consists of a sequence, called a template, of 12mer DNA strands. (A 12mer is a single strand of 12 nucleotides.) There are three types of participating 12mers, denoted *X*, *Y*, and *Z*, with *Z* representing number zero and *X* and *Y* representing number one. The initial template contains only the *Z* 12mers. At each consequent stage of the computation, one *Z* 12mer is replaced by either *X* or *Y*, incrementing the counter by one. This particular design employs two rules, encoded as sequences called *primers*. Each step of the computation involves the annealing of one of the primers to the current template and an extension of this primer to synthesize a new strand. After the DNA duplex is melted, this extended strand becomes the new template in the subsequent step of computation.

In this design, the first decision that has to be made is to select the *X*, *Y*, and *Z* sequences, as well as the primers. The fundamental property of programmed mutagenesis is that annealing of a primer to the template needs to include mismatches; otherwise, the original template would never be modified. Programmed mutagenesis is an example of a method that uses errors in biomolecular operations to facilitate the essential features of the computation.

The selection of a mismatch location plays an important role. If the bond between the primer and the template is too unstable, the extension of the primer will not be successful. The SCAN program considers many candidate sequences for the unary counter and selects the best ones. Another condition that must be met by the design is that the sequences used as rules of computation must be present in the solution simultaneously. The rules that are not actively used in a computational step should not be interfering. In particular, inactive rules should not be able to bind to the template sequence at any location. The sequences used in the design must have a low chance of forming secondary structures. This is necessary since a sequence that binds to itself at any point in the computation becomes unusable. Finally, this design of the unary counter is included as part of a plasmid, or a circular DNA molecule. That is, the DNA sequence encoding the unary counter is incorporated into the sequence of the plasmid. This plasmid needs to be chosen so that primers do not bind to it. The SCAN program tests all of these constraints. For the unary counter, SCAN examined over 7.5 billion design candidates and narrowed the choice down to nine candidates that satisfied all of the required constraints.

Edna

One of the most natural attempts to understand and improve the processes involved in DNA computing is to simulate the procedures and chemical reactions that take place in the laboratory. An actual laboratory experiment, whether successful or not, may take weeks or even months to conduct. At this stage of biomolecular computing, advancements in the field are made through trial and error processes. A large number of trials to learn from is therefore conducive to further progress. Having simulation software that can closely resemble laboratory procedures allows the opportunity to gain insight into wet lab experiments without spending the time required to carry out the experiments themselves.

One example of such software is Edna (Garzon & Oehmen, 2001), a simulation tool that uses a cluster of PCs and demonstrates the processes that could happen in test tubes. Edna can be used to determine if a particular choice of encoding strategy is appropriate, to test a proposed protocol and estimate its performance and reliability, and even to help assess the complexity of the protocols. Test tube operations are assigned a cost that takes into account many of the reaction conditions. The measure of complexity used by Edna is the sum of these costs added up over all operations in a protocol. Other features offered by the software allow the prediction of DNA melting temperature, taking into account various reaction conditions. One of the crucial properties

of Edna is that all molecular interactions are local and reflect the randomness inherent in biomolecular processes. The test tube reactions simulated by Edna can be carried out in the virtual test tubes under a variety of reaction conditions. Temperature, salt concentrations, and strand concentrations can be adjusted as necessary. Testing the scalability of a proposed protocol is another application of Edna.

In addition to the software described (BIND, SCAN, and Edna), a number of other software packages can aid in biomolecular computing. A DNA sequence compiler (Feldkamp et al., 2001a), a DNA sequence generator (Feldkamp et al., 2001b), and the NACST/Seq sequence design system (Kim et al., 2002) are some examples of these.

ALGORITHMIC METHODS

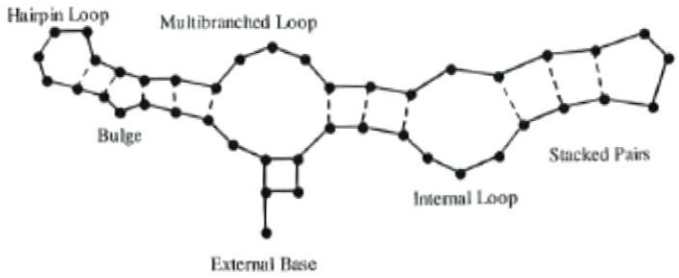
Structure-Free DNA Word Sets

It is possible that within a single-stranded DNA molecule some segments will bind to other segments of the same molecule, forming loops, bulges, and other shapes. This process involves the formation of base pairs within a single DNA strand and the folded structure is referred to as the *secondary structure* of DNA (see Figure 1). For a given molecule, there may be several possible secondary structures. Moreover, each formed structure is not necessarily stable; it may partially fold, then unfold, and, finally, refold into a different structure. Exactly how this happens depends on which segments of the molecule are Watson-Crick complementary and where these segments are located within the molecule.

Formation of secondary structures is an important factor to be considered in the designs of DNA-based computation. Suppose a strand is intended to be used for computation by interacting with other strands, and instead it folds into a secondary structure. It follows that this strand becomes useless for computation and introduces errors into the process. Therefore, a major effort in DNA computing is directed towards the study of how to predict and avoid secondary structures in DNA.

While there have been algorithms proposed for predicting the secondary structure of RNA and DNA, a new twist to the problem arises in the context of DNA computing. The structure freeness problem is proposed by Andronescu et al. (2003). Suppose that we have a set S_i containing DNA single strands, and they all have length l_i. We have t such sets — that is $1 \leq i \leq t$. Note that the lengths l_i need not be the same. Let S be the cross product of these sets, or

Figure 1. Secondary structure of a DNA molecule (dotted lines are base pairs, and solid lines form the backbone)



$S = S_1 \times S_2 \times \ldots \times S_t$. The structure freeness problem for combinatorial sets asks if the strands in the set S are free from secondary structure. The goal is to choose sets S_i in a way that ensures no secondary structure for strands from S.

One example where the structure freeness problem arose in DNA computation was in solving the satisfiability problem (Braich et al., 2001). The satisfiability problem asks, given Boolean expression, if there exist value assignments that make this formula true. The solution involved six word pairs, where each word is a 15mer. These words were used to construct new strands by choosing one word from each pair and concatenating these six words to form long strands of length 6 * 15 nucleotides. Since there are two words to choose from and six pairs in total, there are 2^6 resultant strands. This is a particular instance of the structure freeness problem as formulated earlier, and it can be solved using the algorithm by Andronescu et al. (2003).

The algorithm is based on calculating the free energy of the secondary structure and uses thermodynamic parameters somewhat similarly to the techniques used in predicting the melting temperature of a DNA duplex, as described earlier. DNA tends to fold into structures with the lowest free energy. The free energy of a loop can be experimentally established, which is done for common types of loops, such as hairpin loops, multibranched loops, internal loops, bulges, and other shapes encountered in secondary structures. The free energy of the secondary structure is calculated as a sum of the free energies of the component loops. In this model it is assumed that the free energy of a given loop does not depend on other loops. The strand is said to have secondary structure if the free energy corresponding to that strand is positive, and is said to be structure-free if the free energy is a negative value.

The algorithm for solving the structure freeness problem is based on dynamic programming techniques. It runs in $O(n^3)$ time and is a practical tool for testing if a proposed word set meets the required condition of avoiding secondary structures in computation.

Design of DNA Code Words

In surface-based strategies for DNA computing, DNA molecules are attached to a surface, such as a chemically modified gold film (Liu et al., 1998). A large quantity of unattached DNA molecules is placed in the same solution. Some of these molecules can be selected via hybridization to the complementary surface molecules and a subsequent removal of unhybridized molecules. Selected molecules in this case represent solutions to combinatorial problems.

In these types of experiments there is an assumption that a strand will bind to its perfect Watson-Crick complement. This is not always the case and can be the source of potential errors. Suppose there are molecules in the solution that differ by only one nucleotide. It is possible that the surface molecule complementary to one of them can just as easily bind to another molecule. The problem is that a mismatch in a single nucleotide location does not prevent hybridization from occurring. One way to avoid this situation is to ensure that any two molecules in the solution differ in more than one location. This property can be formalized in terms of the Hamming distance, denoted $H(w_1, w_2)$ and defined as the number of locations in which two given words w_1 and w_2 are distinct. In this definition, the two words must be of equal length. $H(w_1, w_2) \geq d$ means that word w_1 differs from word w_2 in at least d places. For a set of words S, the Hamming distance constraint (HD) requires that any two words w_1 and w_2 in the set S have $H(w_1, w_2) \geq d$.

The other type of error that can arise in surface-based computing is for the unattached molecules in the solution to bind to each other, instead of hybridizing to those on the surface. This occurs between molecules w_1 and w_2 that are Watson-Crick complementary. More specifically, w_1 is the reverse complement of w_2, which is defined as the reversed sequence of w_2, where each nucleotide is replaced by its complement. The reverse complement of w_2 is denoted by $wcc(w_2)$. The reverse complement Hamming distance constraint (RC) requires that for any two words w_1 and w_2 in the set S, we have $H(w_1, wcc(w_2)) \geq d$.

Another consideration in surface-based computing is that hybridization to different surface strands should occur simultaneously. This implies that respective melting temperatures should be comparable for all hybridization reactions

that are taking place. This is the third main constraint that the set of words under consideration needs to adhere to.

To address the design of DNA code words according to these three constraints, Tulpan et al. (2003) propose an algorithm based on a stochastic local search method. The melting temperature constraint is simplified to the constraint requiring that the number of *C* and *G* nucleotides is 50 percent. The algorithm produces a set of DNA words that satisfy one, two, or all three of the noted constraints, as required. The difficulty with nonstochastic methods for solving this problem is that there is no known polynomial-time algorithm for it. This problem, as many other optimization problems, is NP-complete. Stochastic methods, however, prove to be an effective alternative.

The algorithm takes as input the number of words needed to produce and the word length. It also takes as input the set of combinatorial constraints that must be satisfied. The first step of the algorithm is to produce a random set of k words. This set is then iteratively modified by decreasing the number of constraint violations at each step. More specifically, two words w_1 and w_2 are chosen from the set that violate at least one of the constraints. (If such words cannot be found, the algorithm terminates.) With a probability θ, called the noise parameter, one of these words is altered by randomly substituting one base. This substitution, of course, does not necessarily improve the set. Alternatively, with probability $1-\theta$, one of w_1 or w_2 is modified by substitution of one base in a way that maximally decreases the number of conflict violations. The algorithm terminates either when there are no more conflicts in the set of words or when the number of loop iterations has exceeded some maximum threshold.

One drawback of this algorithm is that it may stagnate towards the end in the sense that no improvements to the word set are made after a certain number of steps. This stagnation effect can be overcome by replacing a subset of the words from the set at the point of stagnation with randomly generated words and restarting the algorithm until it reaches the next stagnation. Empirical results prove this technique to be effective. The noise parameter θ is empirically determined to be optimal as 0.2, regardless of the problem instance.

THEORETICAL STUDIES

DNA Sequence Design

The design of DNA sequences optimal for computation can be done with the aid of software, as described in earlier sections. However, some refinement can be added to the process by appealing to theoretical methods. One

technique complementary to the algorithmic construction of DNA sequences uses template-based design (Arita & Kobayashi, 2002). The goal, once again, is to create a set S of words of length l over a four-letter alphabet $\Sigma = \{A, C, G, T\}$ that are as dissimilar as possible. The degree of similarity can be measured with the Hamming distance or some other suitable metric. It is also important that melting temperatures for different pairs of strands are in close proximity. For the purposes of this particular template-based design, it is assumed that similarity of melting temperatures can be achieved by standardizing the GC content of strands. That is, all strands are required to contain an equal proportion of G or C nucleotides with respect to the number of A and T nucleotides. This tactic is motivated by a so-called 2–4 rule that says that the melting temperature for a short (14 to 20 base pairs) DNA duplex is roughly equal to two times the number of A-T base pairs, plus four times the number of G-C base pairs.

The template method creates a set of DNA sequences in two separate stages. In the first stage, a template is chosen according to specific criteria. The template is a sequence over the binary alphabet, where the digit *1* indicates the location of either A or T nucleotide and *0* indicates the place of C or G. In the second stage of the design process, an error-correcting code over the binary alphabet is chosen with words of the same length as that of the template. Each code word is used in combination with the template to create a DNA sequence. The "1" locations in the code word mark places where either A or G will be selected in the future. The "0" locations indicate that the choice will be between T and C. Consider an example template 110100 and code word 101010. The choice of the template implies the sequence is $[AT]$ $[AT]$ $[GC]$ $[AT]$ $[GC]$ $[GC]$, where $[AT]$ means that either A or T is present in that location, and similarly for $[GC]$. The choice is made according to the template word: letters in the odd-numbered locations are picked to be A or G, and those in even locations are picked as T or C. The resulting DNA sequence is, hence, $ATGTGC$. This procedure is followed for each code word in order to obtain the resulting set of DNA sequences. Notice that all sequences will have an identical GC content, as determined by the template.

There are two separate tasks involved in this method. One is to find a suitable template, and the other is to find an error-correcting code of length l. The latter of these tasks is a well-studied problem in coding theory, so the remaining challenge is in finding a good template. What is a good template in this case? It is a binary string x of length l that sufficiently differs from its reverse x^R and from any overlap of its concatenation xx, with possibly reversed occurrences: xx^R, x^Rx, or x^Rx^R. More formally, the *mass* of x is defined to be:

$$mass(x) = min\ [\ H(x, x^R),\ H_M(x, xx),\ H_M(x, x^Rx^R),\ H_M(x, xx^R),\ H_M(x, x^Rx)\ ].$$

In this definition, H is the Hamming distance between two strings of equal lengths. In order to apply a similar measure for strings of different lengths, the measure $H_M(u,v)$ (though not a distance) is defined as the minimum Hamming distance between u and substrings of length $|u|$ in v. What we are looking for is the template x such that $mass(x) = d$. This will ensure that the sequences in the constructed set S differ from other sequences, their concatenations, and reverse complements in at least d locations.

Theoretical analysis methods suggested by Arita and Kobayashi (2002) make it feasible to use an exhaustive search to find an appropriate template, as the search space is significantly reduced. A list of suitable templates is made available by the authors. The number of sequences constructed with this method is limited to the size of the code used. However, it is possible to use multiple templates simultaneously, provided that they are significantly different. A limitation of the template-based sequence design is that it ignores the possibility of strands forming secondary structures.

Different methods of construction are described in Kari et al. (2003b). First, a stronger dissimilarity of sequences in a set K is required — each two subsequences w_1 and w_2 of a fixed length l must satisfy the RC constraint (see the previous section), meaning $H(w_1, wcc(w_2)) \geq d$. At this point, the set K is called $(\theta,H_{d,l})$-bond-free. The number d expresses the degree of dissimilarity — the number of mismatches per each l members of a sequence. Suppose that d mismatches (noncomplementary pairs of nucleotides) are enough to prevent a stable bond between any two (sub)sequences of length l. Then the $(\theta,H_{d,l})$-bond-freedom ensures that the strands in K can form neither (partial) duplexes, nor a stable secondary structure.

One of the most interesting construction methods based on $(\theta,H_{d,l})$-bond-freedom relies on the operation of *subword closure*. Consider a set S of single strands of a fixed length l. Its subword closure $S^{\otimes}$ is the set of all the possible strands with at least l nucleotides, such that each subsequence of $S^{\otimes}$ of length l is in S. The following important result holds: if S is $(\theta,H_{d,l})$-bond-free, then so is $S^{\otimes}$. Moreover, $S^{\otimes}$ is regular and its description (finite automaton or regular grammar) can be constructed from S in linear time. This description allows a rapid construction of $(\theta,H_{d,l})$-bond-free sets. Assume that we want to construct a $(\theta,H_{d,l})$-bond-free set of strands of a fixed length $k \geq l$. There is an algorithm that at each k steps produces one such strand. Moreover, this construction method also produces *maximal* $(\theta,H_{d,l})$-bond-free sets. Such a set is called

maximal if no further sequence can be added without violation of $(\theta, H_{d,i})$-bond freedom. This result can be strengthened as follows: if S is a maximal $(\theta, H_{d,i})$-bond-free set, then so is $S^{\mathcal{R}}$. Hence, with respect to the amount of produced strands (of a given length) that are free of all mutual (partial) bonds, this method is provably optimal.

DNA Languages and Their Properties

A set of single-stranded DNA sequences can be viewed as a formal language over a four-letter alphabet $\Delta = \{A, C, G, T\}$. For an introduction to formal language theory, the reader is referred to Hopcroft et al. (2001). Theoretical studies of language properties can shed light on some characteristics of molecular interactions and aid in the design of encoding strategies. Some representative studies include those by Kari et al. (2003) and Jonoska and Mahalingam (2002). The hybridization of two single strands is formalized by defining a function θ, which acts on a sequence over the Δ alphabet to produce the reverse complement of the original string. For example, θ (*ACCTGACT*) = *AGTCAGGT*, which corresponds to the fact that the two sequences *ACCTGACT* and *AGTCAGGT* would hybridize to form a DNA duplex. While this function attempts to formalize the hybridization process to facilitate theoretical analysis, it has many limitations. It does not take into account base-pair mismatches that can occur during hybridization. The length of the strand is also not taken into account. This function merely reflects the property that if $\theta(u) = v$, then u and v are Watson-Crick complements. The relative simplicity of this function provides several advantages. It allows us to reason about DNA strand interactions theoretically. Consider, for example, the stage of a programmed mutagenic unary counter in which a primer hybridizes to the template. If u is the template and w is the primer, then the template can be written as $u = x\theta(w)y$ for some nonempty strings x and y. A helpful feature of θ is that it happens to be an antimorphism, which is a type of function such that $\theta(uv) = \theta(v)\,\theta(u)$. This is a useful and well-understood property of functions. Another property of θ is that θ^2 is the identity function. A function for which this is true is called an *involution*. Therefore, θ is an antimorphic involution.

The question addressed in earlier parts of this chapter can be asked again from the theoretical perspective. How can we design DNA sequences that minimize undesirable hybridizations?

First, we need to introduce some notation. An *alphabet* is a finite, nonempty set of symbols. Let Σ be an arbitrary such alphabet. Then Σ^* denotes the set of all words over this alphabet, including the empty word λ. Σ^+ is the same as Σ^*, but without the empty word.

Suppose we want to avoid the type of hybridization shown in Figure 2. A language L is defined (Kari et al., 2003) to be θ-compliant if for all words w in L, and words x and y in Σ^*, we have that $w, x\theta(w)y \in L$ imply $xy = \lambda$. In this definition, Σ is an arbitrary alphabet and θ is taken to be any antimorphic involution. If Σ is taken to be the DNA alphabet Δ, then this property is essentially saying that if two DNA sequences could form a structure such as that shown in the figure, then the unhybridized ends of the longer strand (these are called sticky ends) are of length zero. That is, only the special variant of this structure could occur, as shown by Figure 3.

A language L is said to be θ-nonoverlapping if $L \cap \theta(L) = \emptyset$. A language that is both θ-compliant and θ-nonoverlapping is called strictly θ-compliant. In the context of the DNA alphabet, a strictly θ-compliant language over Δ avoids both of the structures in Figures 2 and 3.

Depending on which type of annealing needs to be avoided, a different language property can be defined. For example, a language is called θ-3'-overhang-free if the following condition holds: $\forall w \in \Sigma^*, x, y \in \Sigma^*$ when wx, $\theta(w)y \in L$ implies $xy = \lambda$.

When $\Sigma = \Delta$ and θ is the Watson-Crick antimorphic involution, a language is θ-3'-overhang-free when the structure shown in Figure 4 is avoided. In this DNA structure, a DNA duplex has two overhanging unhybridized ends. These sticky ends are the 3'-ends of the molecules.

Figure 2. A language L *is θ-compliant if it avoids this structure*

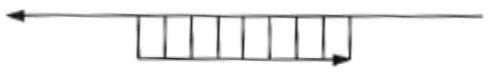
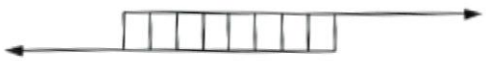


Figure 3. Pattern allowed in a θ-compliant language, but not in a strictly θ-compliant language

Figure 4. Pattern avoided in a θ-3'-overhang-free DNA language



Many other language classes have been similarly defined and their properties studied.

Recently, a more general property of languages, called the bond-free property, was introduced by Kari et al. (2003a). A class $\mathcal{P}$ of languages over Σ is called a *bond-free property of degree 2* if there exist binary word operations $\Diamond_{dn}$ and $\Diamond_{up}$ such that for an arbitrary $L \subseteq \Sigma^*$, $L \in \mathcal{P}$ holds if and only if $\forall w \in \Sigma^+, x, y \in \Sigma^*, (w \Diamond_{dn} x \cap L \neq \emptyset, w \Diamond_{up} y \cap \theta(L) \neq \emptyset)$ implies $xy = \lambda$. The "degree 2" in the name refers to the fact that binding interaction occurs between two DNA strands. To see how this property can generalize other properties, consider the θ-3'-overhang-free property as an example. Let $w \Diamond_{dn} x = \{wx\}$ and $w \Diamond_{up} y = \{yw\}$. Then the definition of bond-free property of degree 2 is instantiated as:

$$\forall w \in \Sigma^+, x, y \in \Sigma^*, \{wx\} \cap L \neq \emptyset, \{yw\} \cap \theta(L) \neq \emptyset \text{ imply } xy = \lambda.$$

This is the same as:

$$\forall w \in \Sigma^+, x, y \in \Sigma^*, wx \in L, yw \in \theta(L) \text{ imply } xy = \lambda.$$

If θ is antimorphic, then $yw \in \theta(L)$ if and only if $\theta(w)\,\theta(y) \in L$. Note that $xy = \lambda$ only if $x\,\theta(y) = \lambda$, which means that choosing $w \Diamond_{dn} x = \{wx\}$ and $w \Diamond_{up} y = \{yw\}$ in the definition of bond-free property of degree 2 reduces it to the definition of θ-3'-overhang-free property.

This definition of bond-free property of degree 2 covers many interactions between two DNA strands. Ten various DNA language properties, such as θ-compliance, θ-3'-overhang-freedom and others, were shown to be its special cases. Moreover, it offers some general solutions addressing DNA sequences without undesirable hybridizations.

Assume, for instance, that there is a certain set of DNA sequences. We want to test theoretically (without experiments) whether undesirable bonds of any mentioned type may occur between a pair of sequences in this set. If the set is finite or even *regular*, then the existence of an effective (quadratic time) testing algorithm was proven. In reality, of course, we always deal with finite sets of DNA sequences. The generalization to regular sets, however, might be important. These sets can be very concisely described by means of *regular grammars* (or *finite automata*). Hence, for large sets of DNA strands this concise description allows us to run the algorithms much faster. Also the maximality problem already mentioned in the previous section was addressed. For instance, consider any finite θ-compliant set of DNA sequences. There

exists an effective (cubic time) algorithm to decide whether new sequences can be added to this set without loss of θ-compliance.

Suppose we use codes (languages with the property that a catenation of words from a language has a unique factorization over this language) that have the language properties we have described. What may happen during the course of computation is that the properties initially present deteriorate over time. This leads to another area of study, which investigates how bio-operations such as cutting, pasting, splicing, contextual insertion, and deletion affect the various bond-free properties of DNA languages. Invariance under these bio-operations is studied by Kari et al. (2003). This paper also discusses how to add error-detecting capabilities to the codes with these properties. Other studies of coding properties of DNA languages are those by Head (2002) and Hussini et al. (2003). Bounds on the sizes of some other codes with desirable properties that can be constructed are explored by Marathe et al. (2001). Earlier results on code sizes can be found in the work of Baum (1998) and Garzon et al. (1997).

CONCLUSION AND FUTURE TRENDS

This chapter presents an overview of computer science methods that can help reduce errors associated with DNA computing. These methods include software that can find optimal encodings with properties conducive to the reduction of errors, algorithmic tactics to deal with errors, and recent studies of language properties desirable for codes in DNA computing.

It is important to remember that since this computational paradigm is biological in nature, the biochemical methods of dealing with errors are of paramount importance. Indeed, in nature the DNA processes occurring in living organisms only rarely lead to uncorrectable errors, such as carcinogenic mutations. It is remarkable, and almost miraculous, that DNA-related errors in nature are frequently either correctable or inconsequential. Hence, a big question that remains is to understand what error-detecting and error-correcting mechanisms are present in vivo. Since the theory of codes offers a wealth of knowledge in error detection and correction, it is only natural to attempt to apply it to DNA-based languages. More studies of these codes are likely to be carried out in the future.

It is hoped that the strategies described in this chapter can eventually reduce the gap between error rates in nature and those in computational designs carried out in vitro, or at least, explain why the gap is there.

REFERENCES

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

Andronescu, M., Dees, D., Slaybaugh, L., Zhao, Y., Cohen, B., Condon, A., & Skiena, S. (2003). Algorithms for testing that sets of DNA words concatenate without secondary structure. *Proceeding of the Eighth International Conference on DNA-Based Computers*, Hokkaido, Japan, June 2002. *Lecture Notes in Computer Science*, 2568, Berlin: Springer, 182-195.

Arita, M., & Kobayashi, S. (2002). DNA sequence design using templates. *New Generation Computing*, 20, 263-277.

Balan, M., Sakthi, Krithivasan, K., & Sivasubramanyam, Y. (2002). Peptide computing: Universality and computing. *Proceedings of the Seventh International Conference on DNA-Based Computers. Lecture Notes in Computer Science,* 2340, Berlin: Springer, 290-299.

Baum, E. B. (1998). DNA sequences useful for computation. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 44, 235-242.

Borer, P. N., Dengler, B., Tinoco, I., Jr., & Uhlenbeck, O. C. (1974). Stability of ribonucleic acid double-stranded helices. *Journal of Molecular Biology,* 86, 843-853.

Braich, R. S., Johnson, C., Rothemund, P. W. K., Hwang, D., Chelyapov, N., & Adleman, L. M. (2001). Solution of a satisfiability problem on a gel-based DNA computer. *Proceedings of the Sixth International Workshop on DNA-Based Computers,* Leiden, The Netherlands, June 2000. *Lecture Notes in Computer Science*, 2054, Berlin: Springer, 27-42.

Faulhammer, D., Cukras, A. R., Lipton, R. J., & Landweber, L. F. (2000). Molecular computation: RNA solutions to chess problems. *Proceedings of the National Academy of Sciences,* 97, 13690-13695.

Feldkamp, U., Banzhaf, W., & Rahue, H., (2001a). A DNA sequence compiler. Poster presented at the *Sixth International Workshop on DNA-Based Computers*, Leiden, The Netherlands, June 2000.

Feldkamp, U., Saghafi, S., Banzhaf, W., & Rahue, H. (2001b). DNA sequence generator: A program for the construction of DNA sequences. *Proceedings of the Seventh International Meeting on DNA-Based Computers*, Tampa, FL, June 2001. *Lecture Notes in Computer Science*, 2340, Berlin: Springer, 23-32.

Freund, R., Păun, Gh., Rozenberg, G., & Salomaa, A. (1999). Watson-Crick finite automata. *Discrete Mathematics and Theoretical Computer Science*, 48, 297-327.

Garzon, M., Deaton, R., Neatheary, P., Murphy, R.C., Franceschetti, D. R., & Stevens, S. E., Jr. (1997 June). On the encoding problem for DNA computing. *Proceedings of the Third Annual DIMACS Workshop on DNA-based Computers*, Philadelphia, PA.

Garzon, M. H., & Oehmen, C. (2001). Biomolecular computation in virtual test tubes. *Proceedings of the Seventh International Meeting on DNA-Based Computers*, Tampa, FL, June 2001. *Lecture Notes in Computer Science*, 2340, Berlin: Springer, 117-128.

Hartemink, J., & Gifford, D. K. (1997). Thermodynamic simulation of deoxyoligonucleotide hybridization for DNA computation. *Proceedings of the Third Annual DIMACS Workshop on DNA-Based Computers*, Philadelphia, PA.

Hartemink, J., Gifford, D. K., & Khodor, J. (1999). Automated constraint-based nucleotide sequence selection for DNA computation. *Biosystems, 52*(1-3), 227-235.

Head, T. (1987). Formal language theory and DNA: An analysis of the generative capacity of recombinant behaviors. *Bulletin of Mathematical Biology*, 49, 737-759.

Head, T. (2002). Relativised code concepts and multi-tube DNA dictionaries, submitted.

Hopcroft, J., Ullman, J., & Motwani, R. (2001). *Introduction to automata theory, languages, and computation* (2nd ed.), Boston: Addison-Wesley.

Hussini, S., Kari, L., & Konstantinidis, S. (2003). Coding properties of DNA languages. *Theoretical Computer Science*, 290, 107-118.

Jonoska, N. & Mahalingam K. (2002). Languages of DNA based code words. *Proceedings of the Ninth International Conference on the DNA-Based Computers*, Madison, WI.

Kari, L., Konstantinidis, S., & Sosík, P. (2003a). On properties of bond-free DNA languages, to appear in *Theoretical Computer Science*, track C.

Kari, L., Konstantinidis, S., & Sosík, P. (2003b). Bond-free DNA languages: Formalizations, maximality and construction methods, to appear in *Int. Journal of Foundations of Comp. Science*.

Kari, L., Konstantinidis, S., Losseva, E., & Wozniak, G. (2003). Sticky-free and overhang-free DNA languages. *Acta Informatica*, 40, 119-157.

Kim, D., Shin, S-Y., Lee, I-H., & Zhang, B-T. (2002). NACST/Seq: A sequence design system with multiobjective optimization. *Proceedings of the Eighth International Conference on DNA-Based Computers*, Hokkaido, Japan, June 2002. *Lecture Notes in Computer Science*, 2568, Berlin: Springer, 242-251.

Liu, Q., Guo, Z., Condon, A., Corn, R. M., Lagally, M. G., & Smith, L. M. (1998). A surface-based approach to DNA computation. *Journal of Computational Biology*, *5*(2), 255-267.

Marathe, A., Condon, A., & Corn, R. M. (2001). On combinatorial DNA word design. *Journal of Computational Biology*, *8*(3), 201-220.

Păun, G. (2000). Computing with membranes. *Journal of Computer and System Sciences*, *61*(1), 108-143.

Roweis, S., Winfree, E., Burgoyne, R., Chelyapov, N. V., Goodman, M.F., Rothemund, P. W. K., & Adleman, L. (1998). A sticker based model for DNA computation. *Journal of Computational Biology*, *5*(4), 615-629.

SantaLucia, J., Jr., Allawi, H. T., & Seneviratne, P. A. (1996). Improved nearest-neighbour parameters for predicting DNA duplex stability. *Biochemistry*, 35, 3555-3562.

Tulpan, D., Hoos, H., & Condon, A. (2003). Stochastic local search algorithms for DNA word design. *Proceedings of the Eighth International Workshop on DNA-Based Computers*, Hokkaido, Japan, June 2002. Lecture notes in *Computer Science*, 2568, Berlin: Springer, 229-241.

Watson, J., Hopkins, N., Roberts, J. W., Steitz, J.A., & Weiner, A. (1987). *Molecular biology of the gene* (4th ed.). Menlo Park, CA: Benjamin Cummings.