Sotirios A. Tsaftaris, Aggelos K. Katsaggelos,
Thrasyvoulos N. Pappas, *and* Eleftherios T. Papoutsakis

DNA Computing from a Signal Processing Viewpoint

We are witnessing a considerable interaction between biology and signal processing. Bioinformatics research has been benefiting tremendously from the increasing application of digital signal processing (DSP) techniques in solving bioinformatics problems, using, for example, wavelets and Fourier transforms to study deoxyribonucleic acid (DNA) sequences, analyzing microarray images, and applying control theory to regulatory networks [7]. Conversely, actual chemistry has been recently used to solve computational problems, giving rise to DNA computing or more general biocomputing. The main focus of this article is to review the most significant results in DNA computing from a signal-processing point of view and to identify areas of interaction between DNA computing and DSP. In an accompanying article, we will provide a brief description of an application of DNA computing in DSP and point to future research directions.

Adleman's pioneering work [1], [2] set the stage for the new field of biocomputing research. His main idea was to use actual chemistry to solve problems that are either unsolvable by conventional computers or require an enormous amount of computation. Adleman's long-term vision was to use DNA computation to design a general-purpose computer.

There are several reasons why computing with DNA may offer advantages over electronic computing. These include memory capacity, massive parallelism, and power requirements [2]. Regarding memory capacity, consider that 1 g of DNA, when dried, occupies a volume of approximately 1 cm^2, while it can store as much information as approximately 1 trillion compact discs [2]. The massive parallelism is illustrated by Adleman's experiment, which was carried out in 1/50th of a teaspoon of solution, and approximately 10^{14} paths were simultaneously concatenated in about 1 s. Not even the world's fastest supercomputer built by the NEC Corp. has this ability. [The Earth Simulator, built by NEC for Japan's Earth Simulator Center, has the ability of 40 TFLOPS (http://www.nec.co.jp/press/en/ 0203/ 0801.html).] Finally, as far as energy efficiency is concerned, in principle 1 J is sufficient for approximately $2 \cdot 10^{19}$ ligation operations (to be explained later), while existing supercomputers operate in the significantly smaller range of 10^9 operations/J [2]. Although the fidelity of biooperations is low [16], these practical incentives and the fascination of being able to perform computations with biological means have inspired many researchers to pursue the challenging topic of DNA computing.

This article introduces the field of DNA computing to the signal processing community, although various signal processing techniques have been used in DNA computing; for example, in designing code words for mapping binary data to DNA data and improving the error rates in DNA interactions. The early steps in the field are summarized, and the most significant contributions are presented. In a subsequent article, we will focus on the use of DNA computing for solving DSP problems.

DNA Essentials

The DNA Molecule, History, and Terminology

A double helix of DNA is made from two single strands of DNA, each of which is a chain of nucleotides [37]. A nucleotide is an organic molecule made up of three basic parts: a phosphate group, a five-carbon sugar group, and a nitrogenous side group, which is more commonly called a base. Four different nucleotides occur in DNA: adenine (A), guanine (G), thymine (T), and cytosine (C). Nucleotides can be joined together in a linear chain to form a single strand of DNA.

A short single strand of DNA consisting of up to 100 or so nucleotides is called an oligonucleotide or oligo. It has a backbone of alternating sugar and phosphate groups with one of the four bases bound to each sugar group. The backbone gives an oligonucleotide a polarity; i.e., it has two distinct ends, the 5′ end and the 3′ end.

1053-5888/04/$20.00©2004IEEE

The chemical structure of the bases allows for the unique pairing between A-T (double hydrogen bond) and G-C (triple hydrogen bond). Each base in DNA has its unique Watson-Crick complement, which is formed by replacing every A with a T and vice versa, and every G with a C and vice versa. Every oligonucleotide has a complementary sequence with opposite polarity; for example, the complementary sequence of 5′-ATG-3′ is 3′-TAC-5′.

If two complementary sequences meet in a solution under appropriate conditions (temperature, pH, sequence length), they will attract each other and form a double-stranded structure. This process is called hybridization. Through hydrogen bonds and Van der Waals forces, these pairings are the basis for the exquisite molecular recognition that allows DNA to act as an information-carrying molecule. There are two types of hybridization: 1) specific hybridization, which refers to cases where the two single strands are perfectly complementary at every position and the double-stranded molecule that is formed is perfect; and 2) nonspecific hybridization, for which the sequence may not be completely complementary, and, thus, it may contain mismatched base pairs, which will appear as "bubbles" or "wobbles."

Tools for Manipulating DNA Molecules

In this section, useful tools for manipulating DNA molecules are presented with DNA computing applications in mind. Useful introductory material for DNA manipulation for DNA computing can be found in [20]. For a more extensive background in DNA, interested readers should consider molecular biology texts, such as [37].

DNA annealing is the process of DNA hybridization when performed in a cell. From a DNA computing perspective, DNA hybridization can provide a mechanism for binding together single stranded molecules. DNA melting, or DNA denaturation, if performed in vitro, is the opposite of DNA annealing. When the temperature is raised, the double-stranded sequence breaks into (melts) two single-stranded parts. While prediction of DNA annealing is very hard, there has been a lot of work on predicting DNA melting through thermodynamic studies (see, for example, [21] and [29]).

Polymerase chain reaction (PCR) is used to amplify a target that contains predefined sequences by running a cycle of annealing-melting-extension operations. Both target and the predefined sequences (primers) are introduced in a solution containing appropriate concentrations of salt with DNA poly -merase (an enzyme that duplicates DNA) and monomers (A,T,G, and C). PCR can give a yes/no answer to whether a given target is present in a solution.

Gel electrophoresis is a technique for separating molecules in a gel medium by applying an electrical field. The frictional force of the gel material acts as a "molecular sieve," separating the molecules relatively to their size and shape.

Affinity purification is a process that permits single-stranded DNA molecules containing a given subsequence to be filtered out from a heterogeneous pool of other DNA molecules. Strands complementary to the subsequence are attached to magnetic beads. The heterogeneous solution is passed over the beads and strands containing the subsequence anneal to the complementary sequence and are retained, while strands not containing it pass through.

Ligation is the process of joining together double stranded DNA with compatible sticky ends with the use of DNA ligase. A double-stranded DNA molecule can either have blunt ends or it can have single–stranded overhanging ends (called sticky ends) at one or both of its extremities. The enzyme DNA ligase joins together, or ligates, the end of a DNA molecule to another molecule.

One class of enzymes, called restriction endonucleases, recognize a specific short sequence of DNA, known as a restriction site and cut any double-stranded DNA at that location. Using enzymes called exonucleases, either double-stranded or single-stranded DNA molecules may be selectively degraded from the ends in.

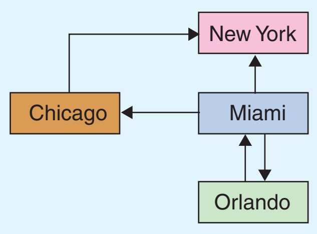
The Field of DNA Computing

The field of DNA computing started with the pioneering work of Adleman in the late 1990s [1], [2]. He demonstrated his ideas by solving a specific combinatorial problem, the Hamiltonian path problem, by applying principles of combinatorial chemistry and DNA chemistry. The principles of combinatorial chemistry had been demonstrated in the pioneering work of Brenner in the 1990s [6]. We provide a more elaborate analysis of Adleman's experiment next.

Adleman's Experiment

As stated above, the capability of encoding information in a DNA sequence and manipulating DNA strands in vitro was used in [2] to solve a seven-node instance of the directed Hamiltonian path problem, a known NP-complete problem (a problem not solvable in deterministic polynomial time). In brief, the essence of this problem is for a given graph with a starting node *v-in* and ending node *v-out* to find a valid path by visiting all other nodes only once.

The following (nondeterministic) algorithm provides a solution to the problem:



(a)

City	DNA Name	Complement
Orlando	AAACCC	TTTGGG
Miami	ATAGTC	TATCAG
Chicago	GAGCGT	CTCGCA
New York	TCGACT	AGCTGA

Flight	DNA Flight
Orlando–Miami	CCCATA
Miami–Chicago	GTCGAG
Miami–New York	GTCTCG
Miami–Orlando	GTCAAA
Chicago–New York	CGTTCG

(b)

▲ *1.Although in Adleman's experiment a seven-node instance was used for illustration reasons in (a) a four-node instance is used, adapted from [2]. The nodes (vertices) are represented as cities connected by nonstop flights (edges). The objective is to determine a route starting in Orlando and ending in New York passing through all the cities exactly once. To achieve a DNA-based solution, every city is given a DNA name (GAGCGG for Chicago) using unique first (GAG) and last (CGG) names (last names are depicted in a smaller font) as illustrated in (b). DNA Flights can then be determined by concatenating the last name of the departing citsy with the first name of the arriving city as seen at the bottom of (b). For this example the DNA solution for the Hamiltonian path is the 18-base sequence CCCATAGTCGAGCGTTCG. In the actual experiment the complements of the city DNA names were used.*

- Step 1) Generate random paths through the graph.
- Step 2) Keep only those paths that begin with *v-in* and end with *v-out*.
- Step 3) If the graph has N vertices, then keep only those paths that go through exactly N vertices.
- Step 4) Keep only those paths that enter all of the vertices of the graph at least once.
- Step 5) If any paths remain, say yes; otherwise say no.

Utilizing the tools for manipulating DNA discussed in the previous section, next we discuss the laboratory procedure followed by Adleman to implement the above algorithm. To implement Step 1, each vertex of the graph was encoded into a random 20-nucleotide strand of DNA that was synthesized. Then, for each (oriented) edge of the graph, a DNA sequence was synthesized consisting of the second half of the sequence encoding the source vertex and the first half of the sequence encoding the target vertex (see Figure 1). By mixing together single strands encoding the edges and complements of single strands encoding the vertices, DNA sequences corresponding to compatible edges were linked together. Indeed, by construction, a complement of a vertex strand would bind to both a strand encoding an edge entering the vertex and a strand encoding an edge exiting the vertex. A subsequent ligation reaction resulted in the formation of DNA molecules encoding random paths through the graph. To implement Step 2, the product of Step 1 was amplified by PCR using as primers the complements of the words coding the start and the end node. Thus, only those molecules encoding paths that include *v-in* and *v-out* were amplified. A valid candidate has to pass through each vertex and therefore has to have a certain length. Implementing Step 3, gel electrophoresis was used to retain only molecules encoding paths of the desired length. Step 4 was accomplished using repetitive applications of affinity purification. In each application, the strands that were retained contained as a subsequence the encoding for the first vertex, second vertex, and so on, until only those paths that pass through all vertices remained. To implement Step 5, the presence of a molecule encoding a Hamiltonian path was checked by amplifying the result of Step 4 by PCR. The molecule was then sequenced to determine the sequence that encodes the actual solution to the problem.

Although the actual computation took a fraction of a second, it took seven days in the lab to perform the above laboratory procedure and read the results. Laboratory automation and Lab-on-a-Chip products clearly indicate that laboratory procedures will become more efficient and less time consuming, bringing DNA computing closer to reality in the not-too-distant future.

At the beginning, the work of Adleman stirred the interest of many researchers but led to some confusion about the capabilities and the performance of DNA computing. Initial work considered the solution of large combinatorial search problems. The scalability of the approach became immediately an important issue, and numerous publications addressed it and suggested improvements of the procedure [9].

Applications of DNA Computing

A significant body of research, both theoretical and experimental, followed Adleman's work. While there has been a variety of DNA-based solutions to various problems, such as NP-complete problems, it is not the purpose of this article to cover the broad range of research in the field but to provide an overview of interesting DNA computing applications with a signal processing perspective in mind. For further study, interested readers are directed to review texts such as, but not limited to, [10], [20], [24], [30], and references therein.

Molecular Arithmetic and Circuitry

Bancroft and his group [13] demonstrated the execution of the first single-bit addition operations in recombinant DNA, but their protocol did not support subsequent arithmetic operations. Rubin and his group [28] provided an experimental demonstration of reversible arithmetic operations that allowed further operations using the output of arithmetic operations as inputs. Finally, various research groups have demonstrated molecular circuits that behave like transistors [4] and adders [31], [35].

Autonomous Molecular Computing

For some DNA computing approaches, the biomolecular computation required a considerable number of laboratory procedures that usually required human intervention. In some cases, the actual laboratory procedure was an inseparable part of the algorithm and was not used just to provide the output results. For example, in Adleman's work the laboratory procedures were part of the algorithmic steps. Autonomous computing is a term commonly used to describe methods that execute multiple steps autonomously. Some trends towards this direction include:

Autonomous computation using restriction enzymes and ligase. Shapiro et al. [5] demonstrated a simple autonomous computation using restriction enzymes and ligase applied to double-stranded DNA to execute state transitions of a small finite automaton. As a proof of concept, they provided experimental procedures of a finite automaton computing the parity of a sequence of bits encoded in DNA sequences.

Self-assembled nanostructures. With this approach, a computation is executed by the self-assembly of DNA nanostructures (tiles) from component DNA single strands [26]. Currently, this approach is receiving much attention and is considered to be the next generation of DNA computing. For a good source of information, see [23].

Deoxyribozyme-based molecular automation. Recently, the group of Stojanovic and Stefanovic built a DNA computer to play tic-tac-toe [32]. Their device, MAYA, is a deoxyribozyme-based molecular automaton and represents the first time artificial DNA molecules have been assembled into circuits that can make complex decisions. MAYA is comprised of nine tubes that contain synthetic DNA enzymes responsible for guiding its moves. The enzymes are designed to release fluorescent molecules only when specific DNA fragments are present or absent. Combinations of these enzymes make up the circuits in MAYA that enable the analysis of complex arrays of inputs to play tic-tac-toe. The user inputs his/her move in all tubes and the computer responds by releasing a fluorescent molecule in the tube that corresponds to each move.

Cellular computation. A novel approach towards automation is to utilize microorganisms, such as bacteria, and modify, by re-engineering, the regulatory feedback systems used in cellular metabolism to program behavior that represents computation. Some work has already been demonstrated (see, for example, [15]) but unfortunately such a task is extremely difficult with the current knowledge, and the procedures can easily destroy the cell instead of extending its behavior.

"Killer Apps" for DNA Computing

It has been demonstrated from a theoretical point of view that DNA computing is universal (all purpose). For example, DNA computing with self-assembly [38] has been proven to allow universal computation. It is believed, though, that certain applications of DNA computing, called "killer-apps," will have a tremendous impact on other fields and will give ground for commercial use.

Genome research and biotechnology are believed to be the areas that will benefit the most from DNA computing. Adopting techniques, which were developed for code word design for DNA computation, new ways for designing index tags for genomic databases can be invented (see for example, [25]). New biotechnology techniques that arose from DNA computing research have already been demonstrated in [18] to assist in faster sequencing of genomes.

Clelland et al. [8] demonstrated for the first time the application of DNA computing in security and intelligence, proposing a DNA-based steganography scheme of embedding secret DNA messages in a human genome that were extracted with the use of PCR.

Nanorobotics and self-assembly are examples of application of DNA computing in nanotechnology. DNA-assisted self-assembly of materials has been used extensively in nanotechnology, as reviewed in [30]. It has evolved even in building molecular circuits using self-assembly of DNA strands attached to nanoparticles [19], a technology that can lead to faster diagnostic tools and laboratory methods. Yurke et al. [36] presented the first molecular motor by developing molecular tweezers using three specially designed DNA strands, which can be programmed to open and close.

Significance of Code Word Design

As we have already seen, a critical step in Adleman's experiment was the assignment of DNA sequences to the available cities and routes. It was clear from the beginning that in order for DNA computations to be useful, efficient, and reliable, sophisticated DNA code word design techniques should be developed. Code word design is the key to success for any DNA-computing application and it is tightly dependent on the application and the laboratory procedures used.

In principle, the successful DNA encoding requires finding DNA code words that can carry information useful for computation in a reliable manner. A four-base (quaternary) representation provides a large flexibility in choosing a code word design scheme. Ideally, a four-base word would have been adequate to encode all possible 256 levels of an 8-b digital sample or pixel value. However, a number of constraints need to be imposed in order to reduce the error rate of biomolecular applications, making such a straightforward mapping nonfeasible.

Constraints in Code Word Design

In order to describe these constraints, the appropriate notation is first introduced. We denote a DNA sequence of length l by x_l with values from the alphabet {A, T, G, C} always in a $5'$ to $3'$ direction, by x_l^C, the Watson-Crick complement of the sequence, and by $x_l[i]$, the base in position i from left to right.

Reading a sequence from right to left, the reverse of a sequence can be formed, denoted by x_l^R. Let us consider for example the sequence of length 6 in direction $5'$ to $3'$, $x_6 = 5'$-ACAGTA-$3'$. In this case $x_6^C = 3'$-TGTCAT-$5'$ and $x_6^R = 3'$-ATGACA-$5'$.

The Hamming distance between code words is defined as the number of base differences between the two words. For example, the sequences $x_6 =$ ATAGCT and $w_6 =$ ATTGTT have a Hamming distance equal to 2. We denote the Hamming distance of two code words x_l and w_l of the same length l by $d_H(x_l, w_l)$.

The following constraints on code words have been proposed and used extensively. All constraints refer to code words of the same length l. The constraints are divided in two groups: self-constraints and group constraints.

Self constraints depend only on the code word under examination and are:

- *Consecutive bases constraint.* In some applications, consecutive occurrences (also known as runs) of the same base increase the number of annealing errors. A constraint is imposed on the maximum number of consecutive occurrences of a base in a code word, $R_B(x_l)$, where $B \in \{A, T, G, C\}$.
- *Self-complementarity constraint.* A code word must not be self-complementary; that is, when it folds it should not anneal to itself.
- *The GC content constraint.* The ratio of the sum of occurrences of G and C bases in a code word over the length of the code word must lie in a certain range to assure similar thermodynamic characteristics between code words.

Group constraints depend on the code word and the rest of the code words and are:

- *The Hamming distance constraint.* To limit unwanted hybridizations between code words, all possible distinct pairs of code words x_l, w_l, the Hamming distance must be greater than some predefined threshold E_H; i.e., $d_H(x_l, w_l) \geq E_H$.
- *The Reverse complement constraint.* To limit hybridization between a code word and the reverse of another, all possible distinct pairs of code words x_l, w_l, $d_H(x_l^R, w_l^C) \geq E_{RC}$ must be true, where E_{RC} is some predefined threshold.
- *The Frame-shift constraint.* If we denote the concatenation of two code words x_l and w_l by $x_l w_l$, then no other code word $z_l \neq \{x_l, w_l\}$ can be found in the concatenation. For example, for $x_6 = 5'$-ACAGTA-$3'$ and $w_6 = 5'$-ACCTGA-$3'$, the concatenated sequence is equal to $x_6 w_6 = 5'$-ACAGTAACCT GA-$3'$. Then, for example, code word $z_6 = 5'$-AGTAAC-$3'$ could not be a valid code word since it can be found in the concatenation; that is, $x_6 w_6 = 5'$-ACz_6CTGA-$3'$. In other words, no code word must result from the suffix of one code word and the prefix of another.
- *Illegal codes constraint.* Specific substrings must not occur in any code word or concatenation of code words. This is necessary when, for example, restriction enzymes need to be used and the site must be recognizable and specified.
- *Melting temperature T_M constraint.* The melting temperature T_M of a duplex is defined as the temperature at which half of the strands are in the double-stranded state. For perfect and nonperfect duplexes T_M can be estimated under some constraints based on a nearest-neighbor model, as described in [21] and [29]. T_M is a critical parameter for designing PCR experiments. According to this constraint, all duplexes formed by any code word and its complement must have similar melting temperatures and in some small range. In addition, T_M of the duplex formed by a code word and the complement of another one should be very small. This allows for the control of the hybridization errors by controlling the temperature of the PCR reaction.

Previous Work in Code Word Design

Depending on the application, the code words are designed to satisfy some or all of the constraints listed above. For the design of small-length code words a brute force search may be adequate for obtaining a solution. However, as the code word length increases the search becomes very complicated, and, therefore, more sophisticated optimization methods are required. For example, in [6] a greedy algorithm to find possible code words that satisfy the Hamming constraint is utilized. A modified Hamming distance, the H-Measure metric, was introduced in [12] to simultaneously model the Hamming distance, frame shift, reverse complement, and self-complementarity constraints. The biological computing group at the University of Wisconsin [11] and [34] derived certain upper bounds on the maximum size of a code set and suggested the use of dynamic programming and stochastic search to reduce the complexity of the problem.

Other researchers followed a "biological" approach using genetic algorithms (see, for example, [3] and [27]) or evolutionary techniques [9]. In [33] a simulated annealing technique satisfying certain sequence fitness criteria was employed. In [14] a code word design scheme inspired by nature is presented in which rules are defined that produce good code words implemented in rounds of programmed mutagenesis in which specific sequences of DNA are allowed to enter a given strand.

At this point we should note that the design of code words for self-assembly applications differs because it has to account for the three-dimensional behavior of the strands and the structure formed upon their annealing. For this reason molecular folding prediction tools are used [39]. In many cases it is desirable to allow programmability, to create tiles that have protruding ends (sticky ends) and only interact with another specific tile that has compatible ends. In [17] a mathematical construction for creating such specific strands is provided.

Conclusions

Adleman's work established the foundations for biocomputing research. In this article, we provided an analysis of Adleman's experiment and a review of DNA computing applications from a signal-processing point of view. In addition, we emphasized certain key parts of DNA computing, such as code word design, to which the application of signal-processing theory and techniques can offer significant advantages.

The goal of this article is to introduce to the signal-processing community a new unexplored area of research. As we will see in an accompanying article, DNA-based DSP can offer some significant advantages over traditional DSP.

Acknowledgments

Mr. Tsaftaris would like to thank the Alexander S. Onassis Public Benefit Foundation for their continuing financial support via their scholarship program.

References

[1] L. Adleman, "Molecular computation of solutions to combinatorial problems," *Science*, vol. 266, no. 5187, pp. 1021–1024, 1994.

[2] L. Adleman, "Computing with DNA," *Sci. Amer.*, vol. 279, no. 2, pp. 54–62, 1998.

[3] M. Arita, A. Nishikawa, M. Hagiya, K. Komiya, H. Gouzu, and K. Sakamoto, "Improving sequence design for DNA computing," in *Proc. Genetic and Evolutionary Computation Conf. 2000*, 2000, pp. 875–882.

[4] E. Ben-Jacob, Z. Hermon, and S. Caspi, "DNA transistor and quantum bit element: Realization of nano-biomolecular logical devices," *Phys. Lett. A*, vol. 263, no. 3, pp. 199–202, 1999.

[5] Y. Benenson, T. Paz-Elizur, R. Adar, E. Keinan, Z. Livneh, and E. Shapiro, "Programmable and autonomous computing machine made of biomolecules," *Nature*, vol. 414, no. 6862, pp. 430–434, 2001.

[6] S. Brenner and R. Lerner, "Encoded combinatorial chemistry," *Proc. Natl. Acad. Sci.*, vol. 89, no. 12, pp. 5381–5383, 1992.

[7] J. Chen, H. Li, K. Sun, and B. Kim, "How will bio-informatics impact signal processing?" *IEEE Signal Processing Mag.*, vol. 20, no. 6, pp. 16–26, Nov. 2003.

[8] C.T. Clelland, V. Risca, and C. Bancroft, "Hiding messages in DNA microdots," *Nature*, vol. 399, no. 6736, pp. 533–534, 1999.

[9] R. Deaton, M. Garzon, R.C. Murphy, J.A. Rose, D.R. Franceschetti, and S.E. Stevens, Jr., "Reliability and efficiency of a DNA-based computation," *Phys. Rev. Lett.*, vol. 80, no. 2, pp. 417–420, Jan. 1998.

[10] R. Deaton, M. Garzon, J.A. Rose, D.R. Franceschetti, and S.E. Stevens, Jr., "DNA computing: A review," *Fundamenta Informaticae*, vol. 35, no. 1–4, pp. 231–245, 1998.

[11] A.G. Frutos, Q. Liu, A.J. Thiel, A.M.W. Sanner, A.E. Condon, L.M. Smith, and R.M. Corn, "Demonstration of a word design strategy for DNA computing on surfaces," *Nucleic Acids Res.*, vol. 25, no. 23, pp. 4748–4757, 1997.

[12] M. Garzon, R. Deaton, P.D. Neathery, R.C. Murphy, S.E. Stevens, and R. Franceschetti, "A new metric for DNA computing," in *Proc. Genetic Programming 1997*, pp. 479–490.

[13] F. Guarnieri, M. Fliss, and C. Bancroft, "Making DNA add," *Science*, vol. 273, no. 5272, pp. 220–223, 1996.

[14] A.J. Hartemink, D.K. Gifford, and J. Khodor, "Automated constraint-based nucleotide sequence selection for DNA computation," *BioSystems*, vol. 52, no. 1–3, pp. 227–235, 1999.

[15] S. Ji, "The cell as the smallest DNA-based molecular computer," *BioSystems*, vol. 52, no. 2, 1–3pp. 123–133, 1999.

[16] L. Kari, "DNA computing—The arrival of biological mathematics," *The Mathematical Intelligencer*, vol. 19, no. 2, pp. 9–22, 1997.

[17] L. Kari, S. Konstantinidis, E. Losseva, and G. Wozniak, "Sticky-free and overhang-free DNA languages," *Acta Informatica*, vol. 40, no. 2, pp. 119–157, 2003.

[18] B. Mishra, "Comparing genomes," *Computing Sci. Eng.*, vol. 4, no. 1, pp. 42–49, 2002.

[19] S.-J. Park, T.A. Taton, and C.A. Mirkin, "Array-based electrical detection of DNA using nanoparticle probes," *Science*, vol. 295, no. 5559, pp. 1503–1506, 2002.

[20] G. Paun, G. Rozenberg, and A. Salomaa, *DNA Computing. New Computing Paradigms.* New York: Springer-Verlag, 1998.

[21] N. Peyret, P.A. Seneviratne, H.T. Allawi, and J. SantaLucia, Jr., "Nearest-neighbor thermodynamics and NMR of DNA sequences with internal A-A, C-C, G-G, and T-T mismatches," *Biochemistry*, vol. 38, no. 12, pp. 3468–3477, 1999.

[22] J.H. Reif and T.H. LaBean, "Computationally inspired biotechnologies: Improved

DNA synthesis and associative search using error-correcting codes and vector-quantization," *Revised Papers from the 6th Int. Workshop on DNA-Based Computers: DNA Computing, Lecture Notes in Computer Science*, vol. 2054. Berlin: Springer-Verlag, 2001, pp. 145–172.

[23] J.H. Reif, "DNA lattices: A programmable method for molecular scale patterning and computation," *Comput. Scientific Eng. (Special Issue on Bio-Computation)*, no. 1, pp. 32–41, Feb. 2002.

[24] J.H. Reif, "The emergence of the discipline of biomolecular computation in the US," *New Gener. Comput.*, vol. 20, no. 3, pp. 217–236, 2002.

[25] J.H. Reif, T.H. LaBean, M. Pirrung, V. Rana, B. Guo, K. Kingsford, and G. Wickham, "Experimental construction of very large scale DNA databases with associative search capability," in *Proc. 7th Int. Meeting DNA Based Computers,* Tampa, FL, 2001, pp. 231–247.

[26] S. Roweis, E. Winfree, R. Burgoyne, N. Chelyapov, M. Goodman, P. Rothemund, and L. Adleman, "A sticker based architecture for DNA computation," in *Proc. 2nd Annu. Workshop DNA Computing*, Princeton, NJ, 1999, pp. 1–29.

[27] A.J. Ruben, S.J. Freeland, and L. Landweber, "PUNCH: An evolutionary algorithm for optimizing bit set selection," in *Revised Papers from the 7th International Workshop on DNA-Based Computers: DNA Computing, Lecture Notes in Computer Science*. London: Springer-Verlag, 2001, pp. 150–160.

[28] J.P. Klein, T.H. Leete and H. Rubin, "A biomolecular implementation of logically reversible computation with minimal energy dissipation," *BioSyst.*, vol. 52, issue 1–3, pp. 15–23, 1999.

[29] J. SantaLucia, Jr., "A unified view of polymer, dumbbell, and oligonucleotide DNA nearest—Neighbor thermodynamics," *Proc. Nat. Acad. Sci.*,USA, vol. 95, no. 4, pp. 1460–1465, 1998.

[30] N.C. Seeman, "DNA in a material world," *Nature*, vol. 421, no. 6921, pp. 427–431, 2003.

[31] M.N. Stojanovic and D. Stefanovic, "Deoxyribozyme-based half-adder," *J. Amer. Chem. Soc.*, vol. 125, no. 22, pp. 6673–6676, 2003.

[32] M.N. Stojanovic and D. Stefanovic, "A deoxyribozyme-based molecular automaton," *Nature Biotechnology*, vol. 21, no. 9, pp. 1069–1074, 2003.

[33] F. Tanaka, M. Nakatsugawa, M. Yamamoto, T. Shiba, and A. Ohuchi, "Developing support system for sequence design in DNA computing," in in *Revised Papers from the 7th International Workshop on DNA-Based Computers: DNA Computing, Lecture Notes in Computer Science*. London: Springer-Verlag, 2001, pp. 129–137.

[34] D. Tulpan, H. Hoos, and A. Condon, "Stochastic local search algorithms for DNA word design," in in *Revised Papers from the 8th International Workshop on DNA-Based Computers: DNA Computing, Lecture Notes in Computer Science*. London: Springer-Verlag, 2003, pp. 229–241.

[35] B. Yurke, A.P. Miller, and S.L. Cheng, "DNA implementation of addition in which the input strands are separate from the operator strands," *BioSystems*, vol. 52, no. 1–3, pp. 165–174, 1999.

[36] B. Yurke, A.J. Turberfield, A.P. Mills, Jr., F.C. Simmel, and J.L. Neumann, "A DNA-fuelled molecular machine made of DNA," *Nature*, vol. 406, no. 6796, pp. 605–608, 2000.

[37] J.D. Watson, T.A. Baker, S.P. Bell, A. Gann, M. Levine and R. Losick, *Molecular Biology of the Gene,* 5th ed. San Francisco, CA: Pearson/Benjamin Cummings, 2004.

[38] E. Winfree, X. Yang, and N.C. Seeman, "Universal computation via self-assembly of DNA: Some theory and experiments," in *2nd Annu. DIMACS Meeting DNA Based Computers*, 1996, pp. 191–213.

[39] M. Zuker, "Mfold web server for nucleic acid folding and hybridization prediction," *Nucleic Acids Res.*, vol. 31, no. 13, pp. 3406–3415, 2003.

Sotirios A. Tsaftaris, *Aggelos K. Katsaggelos*, and *Thrasyvoulos N. Pappas* are with the Department of Electrical and Computer Engineering, Northwestern University. *Eleftherios T. Papoutsakis* is with the Department of Chemical Engineering, Northwestern University.

dsp history

continued from page 99

References

[1] B.P. Bogert, M.J.R. Healy, and J.W. Tukey, "The quefrency alanysis of time series for echoes: Cepstrum, pseudo-autocovariance, cross-cepstrum, and saphe cracking," in *Time Series Analysis*, M. Rosenblatt, Ed., 1963, ch. 15,pp. 209–243.

[2] A.V. Oppenheim, "Superposition in a class of nonlinear systems," Ph.D. dissertation, MIT, May, 1964.

[3] J.W. Cooley and J.W. Tukey, "An algorithm for the machine computation of complex Fourier series," *Math. Computation*, vol. 19, pp. 297–301, Apr. 1965.

[4] A.M. Noll, "Short-time spectrum and 'cepstrum' techniques for vocal-pitch detection," *J. Acoust. Soc. Amer.*, vol. 36, no. 2, pp. 296–302, Feb. 1964.

[5] A.M. Noll, "Cepstrum pitch determination," *J. Acoust. Soc. Amer.*, vol. 41, no. 2, pp. 293–309, Feb. 1967.

[6] R.W. Schafer, "Echo removal by discrete generalized linear filtering," Ph.D. dissertation, MIT, Jan. 1968.

[7] A.V. Oppenheim, R.W. Schafer, and T.G. Stockham, Jr., "Nonlinear filtering of multiplied and convolved signals," *Proc. IEEE*, vol. 56, no. 8, pp. 1264–1291, Aug. 1968.

[8] A.V. Oppenheim and R.W. Schafer, *Digital Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1975.

[9] A.V. Oppenheim and T.G. Stockham Jr., "Signal compression and expansion system," U.S. Patent 3 518 578, June 1970.

[10] T.G. Stockham, Jr., "Image processing in the context of a visual model," *Proc. IEEE*, vol. 60, pp. 828–842, July 1972.

[11] T.G. Stockham, Jr., T.M. Cannon, and R.B. Ingebretsen, "Blind deconvolution through digital signal processing," *Proc. IEEE*, vol. 63, pp. 678–692, Apr. 1975.

[12] S.B. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-28, no. 4, pp. 357–366, Aug. 1980.

[13] T.J. Ulrych, "Application of homomorphic deconvolution to seismology," *Geophysics*, vol. 36, no. 4, pp. 650–660, Aug. 1971.

[14] P.L. Stoffa, P. Buhl, and G.M. Bryan, "The application of homomorphic deconvolution to shallow-water marine seismology–Part I: Models; Part II: Real data," *Geophysics*, vol. 39, pp. 401–426, Aug. 1974.

[15] K. Steiglitz and B. Dickinson, "Computation of the complex cepstrum by factorization of the Z-transform," in *Proc. Int. Conf. Acoust., Speech and Signal Processing*, 1977, pp. 723–726.

[16] G.A. Sitton, C.S. Burrus, J.W. Fox, S. Treitel, "Factoring very high-degree polynomials," *IEEE Signal Processing Mag.*, vol. 20, no. 6, pp. 27–42, Nov. 2003.

For an extensive bibliography, see http://www.rle.mit.edu/dspg/pub_journal.html.