PERSPECTIVES

TIMELINE

The past, present and future of molecular computing

Adam J. Ruben and Laura F. Landweber

Ever since scientists discovered that conventional silicon-based computers have an upper limit in terms of speed, they have been searching for alternative media with which to solve computational problems. That search has led them, among other places, to DNA.

When most people think of a 'DNA computer', the first image that springs to mind is a personal computer-like interface with microcentrifuge tubes lined up inside the central processor and a keyboard that plugs directly into the molecule's 5′ end. Perhaps someday such a project will become reality. At the moment, however, 'DNA computing' is the slightly misleading title applied to experiments in which DNA molecules have computational roles. Often sequences of about 8–20 base pairs are used to represent bits, and numerous methods have been devised to manipulate and evaluate them.

DNA is a convenient choice, as it is both self-complementary, allowing single-stranded DNA to select its own Watson–Crick complement, and can easily be copied. Also, molecular biologists have already built a toolbox for manipulating DNA, including restriction enzyme digestion, ligation, sequencing, amplification and fluorescent labelling, giving DNA a head start over alternative computational media.

This unique combination of computer science and molecular biology has fascinated the world for nearly six years, perhaps because it finally links two popular disciplines that we have always hoped would be innately linked. Much of the human body is, in theory, a flood of binary operations. We could be staring into the abyss of a science as doomed as phrenology or mesmerism. But we may be at the forefront of a new and creative technology whose implications have not even been fully mapped out, let alone realized. Fifty years from now, molecular computing may conceivably be important in our lives — who, fifty years ago, could have predicted the personal computer? — and it all began with an experiment published by a computer scientist in *Science* in 1994.

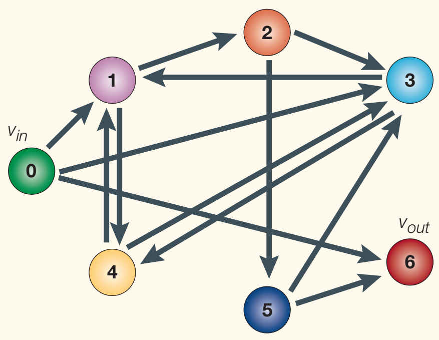

Figure 1 | **An example of a seven-vertex, 13-edge graph.** The 'salesman' starts his journey at vertex 0 (v_{in}) and ends at vertex 6 (v_{out}).

The Travelling Salesman Problem

In a seminal paper, Leonard Adleman[1] solved a simplified instance of a famous NP-complete computer problem called the Travelling Salesman Problem. ('NP' is the name given to the class of search problems for which correctness of solutions is easy to check, and 'NP-complete' problems are considered the hardest of the class because they require exponentially increasing amounts of time to solve.)

The problem asks whether, given a set of n cities ('vertices') with m paths ('edges') connecting them, a 'hamiltonian' path exists that starts at a given vertex v_{in}, passes through each vertex exactly once, and ends at vertex v_{out}. For fairly small values of n, today's computers easily solve this problem. However, when n becomes very large, the amount of time required to generate and check every possible solution increases exponentially, making very large calculations infeasible.

The version Adleman solved contained only seven vertices (FIG. 1), which is no remarkable feat in the world of computer science. He used a simple, brute-force algorithm: generate random paths through the graph, discard any that do not begin at v_{in} and end at v_{out}, discard any that do not enter exactly n vertices, and discard any that do not pass through each vertex at least once. But the use of DNA to solve this small computational problem marked the genesis of a new scientific field.

Adleman began by synthesizing a random 20-base-pair DNA oligonucleotide (20-mer) to represent each vertex, followed by another series of 20-mers to represent edges. The edge DNA had a certain built-in feature: in each 20-mer, the first ten nucleotides complemented the last ten of one vertex, and the last ten complemented the first ten of another vertex (FIG. 2). For example, a 20-base-pair edge connecting vertex one to vertex two would consist of the complements to vertex one's last ten base pairs and vertex two's first ten. That way, when the mixture of DNA is

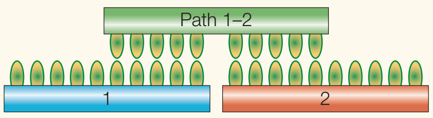

Figure 2 | **Edge DNA forming a splint bridging two vertices.** The last five nucleotides of vertex 1, and the first five of vertex 2, complement the nucleotides on vertex 1–2.

denatured and then allowed to anneal and ligate, the edge will form a splint to 'connect' both vertices, and T4 DNA ligase can join together all possible such combinations. It is the bulk-annealing step that allows this molecular approach to test a massive number of possibilities in parallel.

"Here's nature's toolbox … a bunch of little tools that are dirt cheap; you can buy a DNA strand for 100 femtocents."

This initial denaturing/hybridization/ligation step generated over 10^{13} strands of DNA. Among these, it was probable that at least one encoded the hamiltonian path. Next, all sequences that began at v_{in} and ended at v_{out} were selectively amplified by the polymerase chain reaction (PCR) using primers recognizing those sequences. Any path that did not pass through exactly seven points was then eliminated by gel-purifying only the 140-base-pair product (necessarily containing seven 20-mers). Finally, to eliminate solutions that did not pass through each vertex exactly once, the product from the previous step was affinity purified by denaturing the double-stranded paths and removing the biotinylated complementary strand on magnetic beads. The first affinity target was a single-stranded 20-mer complementary to the sequence of the second vertex, so only paths that contained the second 'city' were retained. This process was repeated for every vertex except the first and the last (as all paths were already bounded by v_{in} and v_{out} PCR primers).

The presence of a DNA band at the end indicated that, for the given graph, a hamiltonian path exists. Confirmation that a path exists and 'readout' of the correct solution required graduated PCR on the final product. This involved a series of six different PCR reactions, each using the v_{in} forward primer and a primer complementary to each of the other 20-mer vertices, which were then analysed in separate lanes on a gel. (So, for example, a readout showing bands of 40, *x*, 60, 80, 100 and 120 base pairs, where *x* represents the absence of a band in the second lane, would represent the path $0 \rightarrow 1$, $1 \rightarrow 3$, $3 \rightarrow 4$, $4 \rightarrow 5$, $5 \rightarrow 6$. This is not a hamiltonian path, as it misses the second vertex, and so would have been eliminated in the size-purification step.) Adleman's path, however, did indeed pass through each vertex exactly once. Although several of these methods have been modified and improved, graduated PCR remains a simple and rapid method for readout that bypasses DNA sequencing.

As this was the first experimental demonstration of DNA computing, Adleman reflected on some possible implications. This computation required about seven days of laboratory work, so for large *n* such a process could prove unwieldy, unless some sort of automation or alternative algorithm could be devised. However, a typical desktop computer can execute about 10^6 operations per second, and the fastest supercomputer can execute about 10^{12} — but Adleman's molecular computation, if each ligation counts as an operation, did over 10^{14}. Scaling up the ligation step could push the number over 10^{20} operations per second. Furthermore, it used an extremely small quantity of energy — 2×10^{19} operations (ligations) per joule, whereas the second law of thermodynamics dictates a theoretical maximum of 34×10^{19} operations per joule. Modern supercomputers only operate at 10^9 operations per joule[1]. Finally, one bit of information can be stored in a cubic nanometre of DNA, which is about 10^{12} times more efficient than existing storage media[1]. So molecular computers have the potential to be far faster and more efficient than any electronics we have developed. There is, of course, a possibility for error, although in this example the thoroughness of each step reduces the chance of survival of an incorrect path. However, in larger instances of the same algorithm, errors are more likely to propagate.

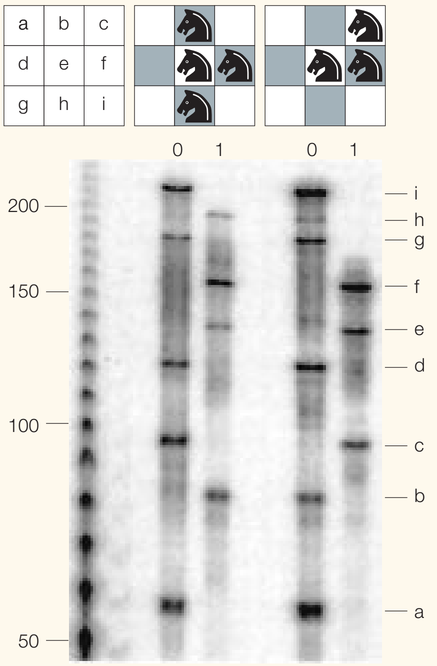

Figure 3 | **Solving the Knight Problem.** Multiplex linear PCR produces a 'bar-code' representing two configurations of knights on a 3 × 3 chessboard[3].

Six years later, DNA computing is still in its infancy. Researchers have developed several algorithms to solve classic problems, and a handful have been tested and work. Despite current progress, we are still a long way away from solving complex problems.

Computing on surfaces

There is an interesting and potentially useful approach now available to the field of DNA computing: computing on surfaces[2]. This involves affixing the solution DNA strands, correct and incorrect, to a solid medium, and — in a subtractive algorithm, for example — selectively destroying the incorrect ones.

Liu *et al.*[2] used this approach to solve a small instance of an NP-complete satisfiability (SAT) problem involving Boolean logic. They began with a logical statement in four variables divided into four connected sections, or 'clauses'. As each variable could be either 0 or 1, there existed $2^4 = 16$ possible solutions to the problem, four of which were correct.

A DNA strand corresponding to each possibility was synthesized. Each strand had the format 5′-FFFFvvvvvvvvFFFF-3′, where the eight bases labelled 'F' are fixed (for use in the PCR amplification step) and 'vvvvvvvv' is a unique octanucleotide corresponding to a different predetermined solution. The strands were bound to solid media (a maleimide-functionalized gold surface), and a fixed 21-mer spacer sequence was attached to the 5′ end in order to separate the solution sequence from the support. All DNA was single stranded.

The algorithm used at this point involved a cycle of mark–destroy–unmark operations. First the sequences encoding correct solutions were marked (by hybridization to their complements), then all unmarked (single-stranded) sequences were destroyed using *Escherichia coli* exonuclease I, and finally the remaining solutions were unmarked by removing their complements. This cycle was then repeated for each clause of the problem.

A distinct advantage of surface computing

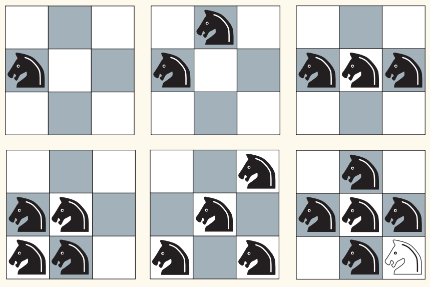
Figure 4 | **Five correct solutions to the Knight Problem and one incorrect solution.** In the last solution, the white knight in square i (see FIG. 3 for key) threatens the knights in squares b and d.

is the readout step. This involves a second surface, which is a DNA chip — an addressed array of surface-bound oligonucleotides. The remaining solution strands were amplified by PCR (using FFFF-region primers) and fluorescein labelled. They were then hybridized to a DNA chip on which each of the original 16 solution molecules were bound in a predetermined spot. The four solutions hybridized to their four corresponding spots, and their presence was then detected by a fluorescence intensity of 10 to 777 times the background intensity of the other 12 spots. So a lot of technology was needed to recover the four sequences out of 16, but again it proved the principle.

DNA chips allow more rapid throughput than direct sequencing, particularly in the crucial readout step. However, graduated PCR[1] has the advantage that it can not only determine which strand is present, but it also measures its length, which is necessary for certain problems. The benefits of using the chip method are also sizeable. For example, each 16-base (FFFFvvvvvvvvFFFF) oligonucleotide 'word' can be extended to several words in sequence. This sort of flexibility allows for easy scaling up, meaning that a much more complex, chip-based DNA computer could be designed[2]. Another advantage is the ease of using solid-state media such as DNA chips. This allows the DNA to be easily separated from solutions, and so no column separation or nucleic acid precipitation steps are necessary. The only components bound to the chip are the DNA and anything attached to it. This greatly streamlines complex, repetitive chemical processes and, perhaps most importantly for the field of DNA computing, opens a door towards automation.

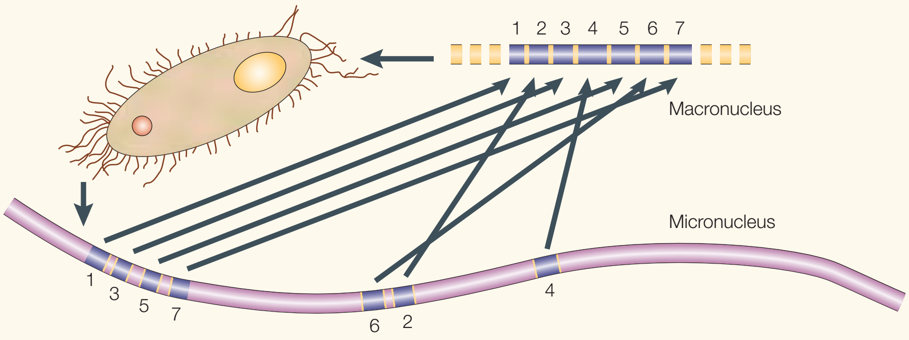

Figure 5 | **DNA computing *in vivo*.** Scrambled genes in some ciliates — microbial eukaryotes of the genus *Stylonychia* or *Oxytricha* — undergo massive rearrangement to form a functional gene in the macronucleus (top) that encodes a single protein product. Telomere repeats (yellow boxes) mark and protect the surviving ends. Dispersed coding segments 1–7 (blue boxes) become joined at their ends, analogous to the assembly of a seven-vertex path through a graph, such as the one in FIG. 1.

RNA computing

Earlier this year our laboratory demonstrated the first use of RNA to solve a computational problem[3]. Known as the Knight Problem, this nine-bit SAT problem (the largest molecular computation solved so far) asks, given a 3 × 3 chessboard, what configurations of knights may be placed on the board such that none threatens any others. (The knight can attack any piece placed two spaces away from it in one direction and one space in the other — that is, moving in an 'L' shape.) There are 94 correct solutions (out of a possible $2^9 = 512$), ranging from one solution with zero knights on the board to two solutions with five knights on the board.

We generated a ten-bit DNA data pool, with a tenth bit included in case one of the previous nine should compute less reliably (similar to having 'spare blocks' on computer disks). Each RNA strand in the data pool consisted of a series of ten 15-nucleotide bits (each of which can represent either 1 or 0, representing the presence or absence of a knight in this case) separated by nine 5-nucleotide spacers. The 5′ 'prefix' and 3′ 'suffix' sequences at the beginning and end permit PCR amplification and T7 RNA transcription of each library strand.

As Liu *et al.*[2] did, we separated the correct solutions from the data pool by 'destroying' the incorrect solutions. This time, however, the enzyme used was ribonuclease H (RNase H) because it digests RNA sequences hybridized to DNA, thus destroying specifically marked strands rather than unmarked ones. Furthermore, the use of RNase H in combination with any complementary DNA oligonucleotide is universally scalable, allowing specific digestion of many 'words' in parallel with a single enzyme. This flexibility in cleaving target sequences was an important incentive for choosing RNA. With DNA, computing can be done with the finite catalogue of restriction enzymes[4], but these require specific, often palindromic, sequences and the use of several enzymes — that may not be compatible — in one pot.

Our readout involved multiplex linear PCR, a process similar to Adleman's graduated PCR[1], except that we combined all the necessary primers in one reaction tube[5], producing a bar-code pattern of all the amplified products in two lanes on a polyacrylamide gel (FIG. 3)[3]. Readouts were taken of 43 randomly sampled strands remaining at the end of the algorithm, of which 42 offered correct solutions to the Knight Problem (FIG. 4). Attention therefore focused on the forty-third solution: If it was incorrect, why had it not been cleaved by RNase H? As it turned out, the RNA strand contained an adjacent point mutation and a deletion in bit nine, preventing its hybridization to the complementary DNA strand and hence digestion by RNase H. So accurate size purification of the data pool (with a resolution of 1–2 nucleotides) would effectively eliminate this most common source of error. However, the 2.3% error (incorrect placement of one knight out of 127 on 43 boards), although not bad in a normal molecular biology experiment, is still higher than that in a computer chip. Fortunately in this type of search problem one can easily check if the recovered solution is correct; hence it is robust to small amounts of error.

Pros and cons

Molecular computing may or may not be a wave of the future, paving the way for technological advances in chemistry, computer science and biology. At its present stage of development it has several challenges.

First, the materials used, whether DNA, RNA or proteins, are not reusable[1]. Whereas a silicon computer can operate indefinitely with electricity as its only input, a molecular computer would require periodic refuelling and cleaning. This illustrates a second important drawback to DNA computing — the molecular components used are generally specialized. A very different set of oligonucleotides is used to solve the Knight Problem[3] than is used to find a hamiltonian path[1].

A third notable downside is the error rate — typically, *in vivo*, with all the cellular machinery functioning properly, a mismatch error occurs only once every billion base pairs or so[6]. However, when DNA is used to compute *in vitro*, the conditions are hardly as good as those *in vivo*, and although the error rate may seem acceptable for other experiments, computational problems generally require higher standards. In order to rival conventional computers, all of the procedures described here would need to lower their error rates to possibly unattainable levels.

Finally, the most apparent drawback is the time required for each computation. Whereas a simple desktop computer can solve the seven-city instance of the Travelling Salesman Problem in less than a second, Adleman took seven days[1]. The use of DNA chips[2] or other approaches may eventually lead to automation, which would save considerable amounts of time, but fundamental DNA computing technology needs to advance far beyond its current bounds before it can be made practical.

DNA computing has its advantages, though. One is its massive parallelism — that is, brute-force algorithms can search through quadrillions of molecules at the same time and find a correct solution, akin to *in vitro* selection[3]. Another is miniaturization. And once the procedures are under control, the raw materials cost less too. "Here's nature's toolbox," commented Adleman[7], "a bunch of little tools that are dirt cheap; you can buy a DNA strand for 100 femtocents."

The near future

Now is an exciting time in the field of DNA computing, as there is so much that has not been tried. In June, over 120 molecular biologists, computer scientists, mathematicians and chemists from around the world gathered in Leiden[8] to discuss the latest in DNA computing technology.

Clearly a next step is automation. McCaskill and colleagues in Germany have constructed a 'microflow reactor' on which they propose to solve a 20-bit satisfiability problem in an hour and a half[8]. One could also construct a microfluidic device consisting of gated channels so small that only one molecule can pass through at a time[9], vastly improving readout[8]. And a team led by Adleman recently solved a 6-variable, 11-clause satisfiability problem using a 'dry' computer consisting of thin, gel-filled glass tubes[8].

As for DNA chips, their future in DNA computing looks bright as well, because 'universal' DNA chips could contain every possible DNA sequence of a given length (probably about 8–12 base pairs). Hagiya and colleagues in Tokyo are finding creative uses for single-stranded DNA molecules that fold into intrastrand 'hairpins'[8,10]. Winfree, Seeman and colleagues — responsible for construction of beautiful assemblies with DNA, such as a DNA nano-cube[11] — have proposed the assembly of even more ordered structures that show patterned algorithmic supramolecular self-assembly[8,11–13]. Even a handful of mathematicians have lent a hand, proposing faster and more efficient algorithms tailored to the needs of DNA computing[8].

Whether or not nucleic acid computers ultimately prove feasible, they have already contributed to multi-disciplinary science by causing us to question the nature of computing and to forge new links between the biological and computational sciences. For example, it has led us to focus on the nature of biological DNA computations, such as the assembly of modern genes from encrypted building-blocks in the genomes of some single-celled ciliates (FIG. 5)[14]. After all, our bodies already contain millions of complicated, efficient, evolved molecular computers called cells.

Adam J. Ruben and Laura F. Landweber are in the Department of Ecology and Evolutionary Biology, Princeton University, Princeton, New Jersey 08544, USA.
e-mail: lfl@princeton.edu

Links

FURTHER INFORMATION DNA computing: a primer | Laura Landweber's homepage

1. Adleman, L. Molecular computation of solutions to combinatorial problems. *Science* **266**, 1021–1023 (1994).
2. Liu, Q. *et al.* DNA computing on surfaces. *Nature* **403**, 175–179 (2000).
3. Faulhammer, D., Cukras, A., Lipton, R. J. & Landweber, L. F. Molecular computation: RNA solutions to chess problems. *Proc. Natl Acad. Sci. USA* **97**, 1385–1389 (2000).
4. Ouyang, Q., Kaplan, P. D., Liu, S. & Libchaber, A. DNA solution of the maximal clique problem. *Science* **278**, 446–449 (1997).
5. Henegariu, O., Heerema, N. A., Dlouhy, S. R., Vance, G. H. & Vogt, P. H. Multiplex PCR: Critical parameters and step-by-step protocol. *Biotechniques* **23**, 504–511 (1997).
6. Karp, G. *Cell and Molecular Biology: Concepts and Experiments* 2nd edn (John Wiley & Sons, New York, 1999).
7. Seife, C. RNA works out knight moves. *Science* **287**, 1182–1183 (2000).
8. Condon, A. & Rozenberg, G. (eds) *Prelim. Proc. 6th Int. Meet. DNA Based Computers* (Leiden Univ., The Netherlands, 2000).
9. Meller, A. *et al*. Rapid nanopore discrimination between single polynucleotide molecules. *Proc. Natl Acad. Sci. USA* **97**, 1079–1084 (2000).
10. Sakamoto, K. *et al*. Molecular computation by DNA hairpin formation. *Science* **288**, 1223–1226 (2000).
11. Seeman, N. C. DNA engineering and its application to biotechnology. *Trends Biotechnol*. **17**, 437–443 (2000).
12. Winfree, E. *et al.* in *DNA Based Computers II. DIMACS Series in Discrete Mathematics and Theoretical Computer Science* Vol. 44 (eds Landweber, L. F. & Baum, E. B.) (American Mathematical Soc., Providence, Rhode Island, 1999).
13. Winfree, E., Liu, F., Wenzler, L. A. & Seeman, N. C. Design and self-assembly of two-dimensional DNA crystals. *Nature* **394**, 539–544 (1998).
14. Landweber, L. F., Kuo, T.-C. & Curtis, E. A. Evolution and assembly of an extremely scrambled gene. *Proc. Natl Acad. Sci. USA* **97**, 3298–3303 (2000).

TIMELINE

Hayflick, his limit, and cellular ageing

Jerry W. Shay and Woodring E. Wright

Almost 40 years ago, Leonard Hayflick discovered that cultured normal human cells have limited capacity to divide, after which they become senescent — a phenomenon now known as the 'Hayflick limit'. Hayflick's findings were strongly challenged at the time, and continue to be questioned in a few circles, but his achievements have enabled others to make considerable progress towards understanding and manipulating the molecular mechanisms of ageing.

To set Hayflick's discoveries in context, we need to go back to 1881 (TIMELINE, overleaf), when the German biologist August Weismann[1] speculated that "death takes place because a worn-out tissue cannot forever renew itself, and because a capacity for increase by means of cell division is not everlasting but finite". This concept, which was almost entirely forgotten by the time Hayflick began his work, was later challenged by the French Nobel-prize-winning surgeon Alexis Carrel, who suggested that all cells explanted in culture are immortal, and that the lack of continuous cell replication was due to ignorance on how best to cultivate the cells. Carrel's view was based on his and Albert Ebeling's work, done at the Rockefeller Institute in New York City, in which they claimed that chick heart fibroblasts grew con-

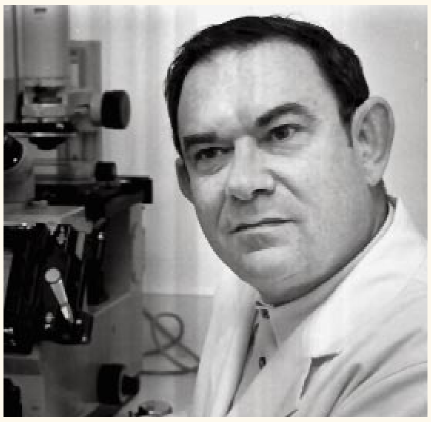
Figure 1 | **Leonard Hayflick in 1988.** (Photograph: Peter Argentine.)