

Dr. Leonard Adleman published the first paper on the use of DNA to solve computationally difficult problems in 1994. He set up a problem and showed how DNA could be used to solve it. The problem that he set up was a Hamiltonian path problem. The well known traveling salesman problem is a related problem where a salesman is given a list of cities that he must visit and the connections between cities, and the goal is to find the shortest route to visit every city exactly once. With 100 cities, for example, there exists an algorithm to solve the problem in  $10^{147}$  operations which would take much longer than the age of the universe to solve by all the computers in the world. Adleman demonstrated, using only 50 picomoles of DNA, how DNA could be used to solve a Hamiltonian path problem with 7 cities and 14 connections. With DNA, the initial state is created by synthesizing DNA molecules with a certain sequence and after some reactions, a new molecule is produced with the answer. It took one second for the DNA to come up with answers, but it took him a week to dig out the answer from the DNA soup.

The Hamiltonian path problem is one of a class of problems called NP-complete. NP problems are problems that can be solved by a non-deterministic Turing machine in polynomial time, which means that a conventional computer must guess a solution (non-deterministic) and then check the solution in polynomial time. The process of “guessing” though makes the entire process unachievable in polynomial time which makes the problem difficult in the sense that it can take an extremely long time to get a solution as the problem gets bigger. A NP-complete problem is essentially the hardest of all NP problems, since all NP problems can be reduced to it. Therefore, theoretically, any NP problem can be solved using Adleman’s method. Other NP-complete search problems have been successfully solved using DNA and recombinant DNA technology. For example, another problem used restriction enzymes, which cut at specific sites of DNA, and looking for the answer in the shortest pieces of uncut DNA. The answer was retrieved from the DNA by inserting the answer DNA into M13 bacteriophage which was then introduced

into *E. coli*. After allowing it to grow, the DNA was then extracted and sequenced to find the answer. Other applications include attacking the governmental encryption scheme DES using molecular computers, solving the satisfiability problem, and using DNA to factor integers.

Richard Lipton in December 1994 published an important paper allowing the development of molecular computers to move forward. He described how DNA base pairs can be translated into 0s and 1s and how boolean algebra can be done with molecules. For example, “and” is done by separating DNA strands by their sequence while “or” is performed by mixing two DNA solutions together. Theoretically, that is all that is necessary for DNA to simulate any normal desktop computer. The unrestricted DNA model described by Lipton include the following operations that can be done with DNA. Synthesis of a desired strand, separation of DNA strands, merging of DNA strands, extracting a strand with a given pattern, annealing single stranded complementary DNA into double stranded DNA, amplification with PCR, cutting of DNA with restriction enzymes, ligation of DNA strands, and finally detecting the presence of certain DNA. With these primitive operations, it is possible to write a molecular program that does something useful. In fact, a lot of these programs look a lot like programs in common languages. The DNA operations can be viewed as subroutine calls that are made when needed.

It is interesting to compare three types of computers: the personal computer, our brain, and molecular computers. Traditional personal computers are very good at doing one thing after another, with each thing being very easy to do. Molecular computers are good at doing a large number of things at once, with very few sequential steps. Supercomputers can perform 1000 million instructions per second (MIPS). In contrast, a single DNA molecule does something at the speed of .001 MIPS. However, molecular computers the size of a tiny test tube could have  $10^{20}$  DNA molecules all doing something simultaneously allowing massively parallel computations with a speed faster than all the traditional computers in the world combined. Our

brain is another type of computer. It has far fewer processors available in a DNA computer but far more processors than a personal computer. Yet, our brain is far slower than a personal computer.

While traditional computers represent information as a sequence of 0s and 1s, DNA computers encode the information directly in their chemical sequence. The density of information that can be stored on DNA is 1 bit per cubic nanometer which is a trillionth of the space required of conventional computers with a density of 1 bit per  $10^{12}$  cubic nanometers. Also, DNA computers are efficient according to Adleman. 1 joule is sufficient for  $2 * 10^{19}$  operations where an operation is defined to be the ligation of two DNA molecules. Computers have an efficiency of about  $10^9$  operations per joule. However, as others have pointed out, Adleman did not include the cost of extracting the information, the PCR that was needed to amplify the initial DNA, and so forth.

One way that DNA may be used is to store information. Searching for information would only require the synthesis of a small probe complementary to the region of interest, and then to locate where the probe bound to the DNA. DNA in a test tube could hold a million times more information than is thought to be stored in the human brain. Another consideration is of error control as mistakes can be made by a molecular computer. Molecules like DNA are naturally degraded, and one proposal has been to use DNA with a peptide backbone to increase stability. Other errors can come from errors in the steps such as separation required in the molecular computer. I believe these errors will eventually be negligible as biochemistry moves forward with better methods. Another possibility is to use non-biological molecules for computation. Non-biological molecules have the potential to be much smaller than DNA. Another related area that is developing is combinatorial chemistry in which pseudo-enzymes can be developed using brute force methods.

One last application of DNA computers is to use it for DNA sequencing. The traditional method, the double digest method, involves breaking up the DNA into overlapping small parts, sequencing the short strands, and then combining the results. The last step is a NP-complete problem. Boneh and Lipton present a new approach which is similar to the quicksort algorithm. To sequence a piece of DNA, a random part of it is chosen, called the pivot. Then the DNA is broken up around the pivot, and like quicksort, the two pieces are recursively sequenced. While this approach has not been tested and may not be quicker than traditional methods, it is still interesting the varied applications that have been devised for molecular computers.

However, there are also some practical issues that present problems with respect to molecular computing. The same problems that occur for traditional computers crop up for molecular computers. As problems get a little bit harder, for example, adding one more point in a Hamiltonian problem, the amount of work increases exponentially. For traditional computers, the amount of time required increases exponentially for these difficult problems. For molecular computers, the amount of material needed increases exponentially. Mac Dónaill shows that for NP problems (by assuming a computational complexity of  $O(2^n)$ ), increasing the scale of the molecular system by  $10^k$ , will increase the size of the solvable problem by  $k / \log_{10} 2$ . Assuming that there are  $10^{80}$  particles in the entire universe, the potential increase in the scale of problems that can be solved is  $80 / \log_{10} 2$  which is on the order of  $10^2$ . And that's only if every particle in the universe was used to make a computer.

NP problems are considered hard because it takes in the worst case exponential time to solve the problem. However, that is the worst case scenario. In practice, with traditional computer systems, efficient algorithms have been devised that can solve NP problems which do not have to consider every possible alternative. This is a definite advantage over molecular computers, since as the technology stands now, every possible combination is tried until the

solution is found. This brute force approach may not be applicable to most problems. However, there are some problems in which brute force is the only alternative and in those cases, molecular computers may have the advantage over traditional computers. And as methods become better, it may be possible to also increase the efficiency and practicality of molecular computers.

#### References:

- Adleman, L.M. "Molecular Computation of Solutions to Combinatorial Problems." *Science*. 266: 1021-1024 (Nov. 11, 1994).
- Adleman L.M. "On Constructing a Molecular Computer."
- Boneh D, Lipton R. "A Divide and Conquer Approach to Sequencing." URL: <http://www.cs.princeton.edu/~dabo/papers/bioseq.ps.Z>. Accessed 11/30/1998.
- Boneh D, Lipton R. "Making DNA Computers Error Resistant." URL: <http://www.cs.princeton.edu/~dabo/papers/bioerror.ps.Z>. Accessed 11/30/1998.
- Bunow, B. "On the Potential of Molecular Computing." *Science*. 268: 481-482 (1995).
- Friedman, Y. "DNA Computers." URL: <http://www.clearlight.com/~morph/dna.htm>. Accessed: 11/24/1998.
- Lipton, R. J. "DNA solution of Hard Combinatorial Problems." *Science*. 268: 542-545 (1995).
- Lipton, R. J. "Speeding up Computation via molecular biology." URL: <ftp://ftp.cs.princeton.edu/ftp/pub/people/rjl/bio.ps>. Accessed: 11/22/98.
- Mac Dónail, D. A. "On the Scalability of Molecular Computational Solutions to NP Problems." *J. Universal Computer Science*. Vol. 2. No.2. 87-95.
- Ouyang Q., Kaplan P. D., Liu S., et. al. "DNA Solution of the Maximal Clique Problem." *Science*. 278: 446-449 (1997).