DNA BASED SUPERCOMPUTERS !

Faisal A. Shaikh *(S.Y.Chem.Engg.)*

For forty years , man has turned to inorganic materials like silicon in order to build computational machines which are the basis for most of the computers in use today. However, this silicon based technology will be inadequate to cope with the ever increasing performance requirements of supercomputers.Faced with this 'silicon wall', scientists are now experimenting with a variety of different materials instead of silicon. Out of these the strangest approach ,and also the most promising ,is a technique that uses something as old as mankind itself...the human DNA !

A conventional computer represents information on silicon chips as a series of electrical impulses –zeroes and ones -and manipulates the information by performing mathematical computations with those zeroes and ones. By contrast, a DNA computer represents information as a pattern of molecules in a strand of synthetic DNA. This information is manipulated by subjecting it to precisely designed chemical reactions that may mark the strand , lengthen it, or even destroy it.

Why DNA?

A paper in Science (by Adleman) recently described the "Molecular Computation of Solutions of Combinatorial Problems". This was the first ever implementation of a DNA - based computer. Since then, many advancements have been proposed to refine the protocol for programming a DNA computer to reduce the complexity of operations and eliminate errors.

Despite their respective complexities, biological and mathematical operations have some similarities:

a) The very complex structure of a living being is the result of applying simple operations to initial information encoded in a DNA sequence.

b The result *f(w)* of applying a computable function to an argument *w* can be obtained by applying a combination of basic simple functions to *w*.

For the same reasons that DNA was presumably selected for living organisms as a genetic material and its stability and predictability in reactions, DNA strings can also be used to encode information for mathematical systems.

Subsets of difficulty :

Fig.1 shows how the complexity of computational problems is classified. Adleman used DNA to solve the equivalence class of computational problems called 'NP-complete'

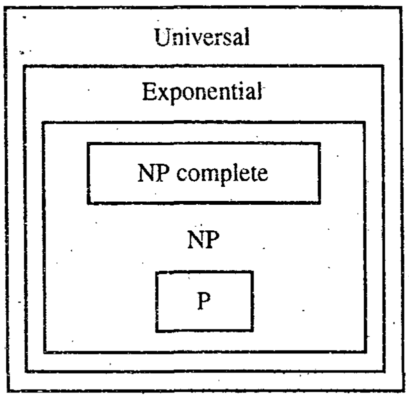

Fig.1: Classification of complexity.
P= Polynomial time problems.
NP= Non deterministic polynomial time problems.

The simplicity of Adleman's method is surprising given that it solves a hard computational problem. The figure shows how computer science problems are ranked in difficulty, starting with easy polynomial time problems on the inside, ranging up to difficult universal machines on the outside. The ranking is based on how long the best algorithm to solve a problem will take to execute on a single computer, considering the size of the problem. Algorithms whose running time is bounded by a polynomial function are in the complexity class "polynomial time" (P for short), and algorithms whose running time is bounded by an exponential function are in the complexity class "exponential time".

The Hamiltonian path problem :

Adleman has approached the Hamiltonian path problem (the objective is to find a path from start to end going through all the given sets of points only once) in a biological context where each vertex and edge of the graph can be represented by a short oligonucleotide. An oligonucleotide is a synthetic single stranded DNA molecule that is made with a chosen sequence. Adleman shows that the binding together of chosen oligonucleotides representing vertices results in DNA molecules that encode the solution to a Hamiltonian path problem. A complete solution is encoded in a single DNA molecule. Adleman further describes how to choose the right oligonucleotides to "input" a problem into his method, and how to isolate and interpret the DNA molecules that represent solutions which are the "output" of his method. In essence, Adleman has used the enormous parallelism of solution-phase chemistry to solve a complex computational problem.

Figure (2) shows the Hamiltonian path problem. The objective is to find a path from start to end going through all the points only once. This problem is difficult for conventional computers to solve because it is a "non-deterministic polynomial time problem " (NP). NP problems are intractable with deterministic (conventional/serial) computers, but can be solved using non-deterministic (massively parallel) computers. A DNA computer is a type of non-deterministic computer. The Hamiltonian path problem was chosen because it is known as "NP-complete" and every NP problem can be reduced to a Hamiltonian path problem.

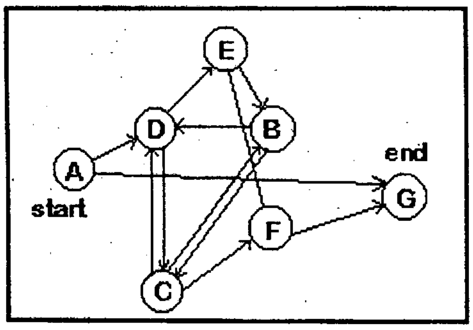

Figure 2: The Hamiltonian path problem. The goal is to find a path from the start city to the end city going through every city only once.

Solving the Problem

The following algorithm solves the Hamiltonian Path problem:

1. Generate random paths through the graph.
2. Keep only those paths that begin with the start city (A) and conclude with the end city (G).
3. If the graph has n cities, keep only those paths with n cities (n=7).
4. Keep only those paths that enter all cities at least once.
5. Any remaining paths are solutions.

   The key to solving the problem was using DNA to perform the five steps in the above algorithm.

Unrestricted model of DNA computing

The following operations can be performed with DNA:

1. *Synthesis* of a desired strand.
2. *Separation* of strands by length.
3. *Merging*: pour two test tubes into one to perform union.
4. *Extraction*: extract those strands containing a given pattern.
5. *Melting/Annealing*: break/bond two single stranded DNA molecules with complementary sequences.
6. *Amplification*: use PCR to make copies of DNA strands (PCR = Polymerase Chain Reaction).
7. *Cutting*: cut DNA with restriction enzymes
8. *Ligation*: Ligate DNA strands with complementary sticky ends using ligase.
9. *Detection:* Confirm presence/absence of DNA in a given test tube

   The above operations can be used to "program" a DNA computer.

Programming with DNA

To implement step (1) of the algorithm, Adleman created a 20-mer sequence of DNA for each city A through G. For each path i→j, an oligonucleotide was created that was the 3' 10-mer of i and 5' 10-mer of j (see Fig. 3). A→B contained all of A and the 5' 10-mer of B, and E→G contained the 3' 10-mer of E and all of G to eliminate any possible sticky ends. For each city i, a Watson-Crick complementary oligonucleotide was synthesized and designated ~i.

For each city except A and G, and for each path, 50 pmol of ~i and 50 pmol of i→j, respectively, were mixed together in a single ligation reaction. The oligonucleotides served as splints to bring paths between common cities to associate for ligation. The ligation reaction therefore resulted in the formation of DNA molecules encoding random paths through the graph.

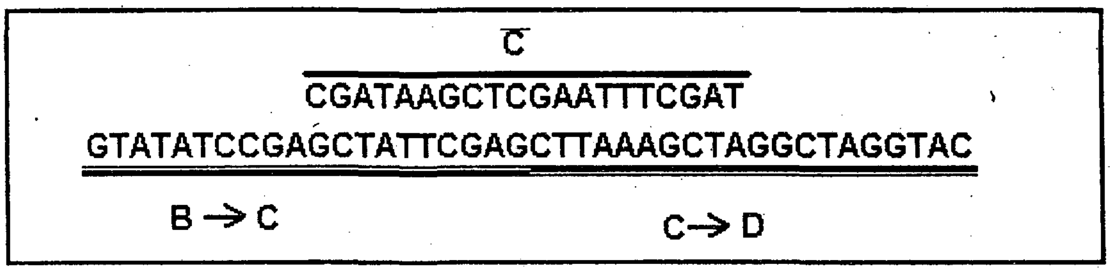

Figure 3: Encoding a path in DNA . For each city A through G, a random 20-mer oligonucleotide was generated. A complementary sequence was generated for each city: ~C is the oligonucleotide complementary to the oligonucleotide C, which represents city C. For each path, a 20-mer oligonucleotide was created which was the 3' 10-mer of the originating city, *and the 5' 10-mer of the target city, except for paths from A or to G. Shown above is DNA encoding the path from B→C→D.*

The scale of this ligation reaction far exceeded what was necessary for the graph under consideration. For each edge in the graph, approximately 3 x10^13 copies of the associated oligonucleotide were added to the ligation reaction. Hence it is likely that many DNA molecules encoding the Hamiltonian path were created. In theory, the creation of a single such molecule would be sufficient. As a result, for this graph quantities of oligonucleotides less than an attomole would probably have been sufficient. Alternatively, a much larger graph could have been processed with the picomole quantities used here. Roughly, the quantity used should be just sufficient to ensure that during the ligation step (step 1) a molecule encoding a Hamiltonian path will be formed with high probability if such a path exists in the graph. This quantity should grow exponentially with the number of vertices in the graph.

Step 2 was implemented by using A and ~G as primers for PCR to amplify all "paths" starting with A and ending with G.

For step 3, the product of step 2 was run on an agarose gel, and the 140 bp band (corresponding to a double stranded DNA path entering exactly seven cities) was excised and soaked in double-distilled water to extract the DNA. This product was then PCR amplified and gel-purified several times to enhance its purity.

In order to complete step 4, the product of step 3 was affinity-purified with biotin-avidin magnetic beads system. The dsDNA (double stranded) from step 3 was melted and the ssDNA (single stranded) product was incubated with ~A conjugated to magnetic beads. Only those ssDNA molecules that entered city A at least once were annealed to the bound ~B and retained. The process was successively repeated with ~C, ~D, ~E, and ~F.

Step 5 was performed by amplifying the product of step 4 with PCR and running it on a gel. Figure 4 shows the results of these procedures. In figure 4A, lane 1 is the result of the ligation reaction in step1. The smear is consistent with the construction of molecules encoding random paths through the graph. Lanes 2-5 show the results of the PCR reaction in step 2. The densest bands correspond to the amplification of molecules encoding paths that began at A and ended at F.

Figure 4B shows graduated PCR of the final product of the computer. Graduated PCR allows one to "print" the results of a computation, using A as the right primer and ~B through ~G as the left primer lanes 1-7, respectively. The graduation shows the progression from A→B→C→D→E→F→G.

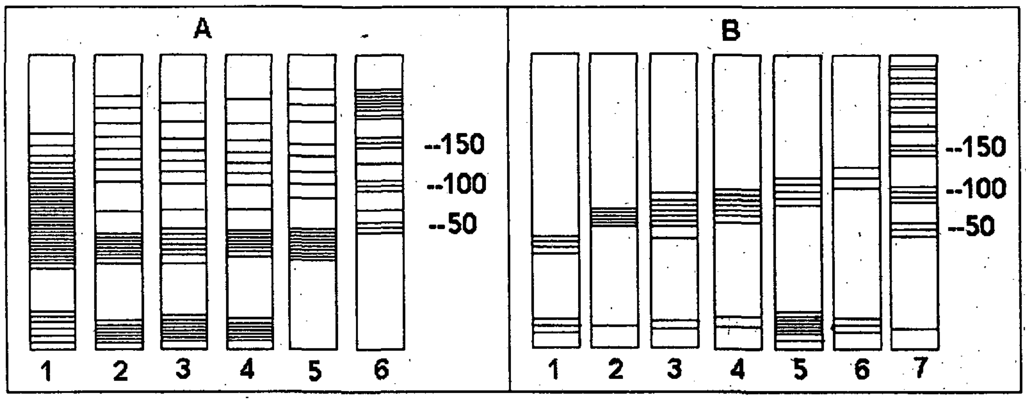

Figure 4: Agarose gel electrophoresis of various products of Adleman's DNA computer. (A) Raw product of the ligation reaction in lane 1; lanes 2-5 contain the PCR amplification of the product of the ligation reaction; lane 6 contains the molecular weight marker in base pairs. (B) Graduated PCR of the final product of the experiment revealing the Hamiltonian path in lanes 1-6 ; molecular weight marker in lane 7.

Power of DNA computation

What is the power of this method of computation? A typical desktop computer can execute approximately 10^ 6 operations per second. The fastest supercomputers currently available can execute approximately 10^ 12 operations per second. If the ligation (concatenation) of two DNA molecules is considered as a single operation and, if it is assumed that about half of the approximately 4 x 10^ 14 edge oligonucleotides in step 1 were ligated, then during step 1 approximately 10^ 14 operations were executed. Clearly, this step could be scaled up considerably, and 10^ 20 or more operations seems entirely plausible (for example, by using micromole rather than picomole quantities). At this scale, the number of operations per second during the ligation step would exceed that of current supercomputers by more than a thousandfold.

Furthermore, hydrolysis of a single molecule of adenosine triphosphate plus pyrophosphate provides the Gibbs free energy ($\Delta G = -8$ kcal/mol) for one ligation operation ; hence in principle 1J is sufficient for approximately 2 x 10^ 19 such operations. This is remarkable energy efficiency, considering that the second law of thermodynamics dictates a theoretical maximum of 34 x 10^ 19 (irreversible) operations per joule (at 300 K). Existing supercomputers are far less energy-efficient, and execute at the most 10^ 9 operations per joule. The energy consumed during other parts of the molecular computation, such as oligonucleotide synthesis and PCR, should also be small in comparison to that consumed by current supercomputers. Finally, storing information in molecules of DNA allows for an information density of approximately 1 bit per cubic nanometer, a dramatic improvement over existing storage media such as video tapes, which store information at a density of approximately 1 bit per 10^ 12 cubic nanometer .

Thus, the potential of molecular computation with respect to speed, energy consumption and data storage is impressive. What is not clear is whether such massive number of inexpensive operations can be productively used to solve real computational problems. One major advantage of electronic computers is the variety of operations they provide and the flexibility with which those operations can be applied. Whereas two 100-digits integers can be multiplied quite efficiently on an electronic computer, it would be a daunting task to do such a calculation on a molecular computer using currently available protocols and enzymes. Nonetheless, for certain intrinsically complex problems, such as the directed Hamiltonian path problem where existing electronic computers are very inefficient and where massively parallel searches can be organized to take advantage of the operations that molecular biology currently provides, it is conceivable that molecular computation might compete with electronic computation in the near term. It is a research problem of considerable interest to elucidate the kinds of algorithms that are possible with the use of molecular methods and the kinds of problems that these algorithms can efficiently solve.

The above procedure took approximately one week to perform. Although this particular problem could be solved on piece of paper in under an hour, when the number of cities is increased to 70, the problem becomes too complex for even a supercomputer .The fastest supercomputers can currently perform 1000 million instructions per second (MIPS) ; a single DNA molecule requires approximately 1000 seconds to perform an instruction (.001 MIPS). Obviously if you want to perform one calculation at a time (serial logic), DNA computers are not a viable option. However, If one wanted to perform many calculations simultaneously (parallel logic), a computer such as the one described above can easily perform 10^14 MIPS. DNA computers also require less energy and space.

The Restricted Model

Since Adleman's original experiment, several methods to reduce error and improve efficiency have been developed. The problems with implementing a DNA computer can be separated into two types:

1) Physical obstructions: difficulties with large scale systems and coping with errors.

A group of researchers stated that a Hamiltonian path problem with 23 nodes would require kilogram quantities of DNA. Another group estimated that if the number of nodes was increased from 7 to 70 then the amount of DNA required to solve such a problem would be 1025kg of nucleic acids. Also the practical difficulties of encoding and manipulating large assemblies of oligonucleotides should be taken into account .

2) Logical obstructions: Concerning the versatility of molecular computers and their capacity to efficiently accommodate a wide variety of computational problems.

The Restricted model of DNA computing solves several physical problems with the Unrestricted model. The Restricted model simplifies the physical obstructions in exchange for some additional logical considerations . The purpose of this restructuring is to simplify biochemical operations and reduce the errors due to physical obstructions.

The Restricted Model of DNA computing

1) *Separation* : isolate a subset of DNA from a sample.
2) *Merging* : pour two test tubes into one to perform union.
3) *Detection* : confirm presence/absence of DNA in a given test tube.

Despite these restrictions, this model can still solve NP-complete problems such as the 3-colourability problem, which decides if a map can be coloured with three colours in such a way that no two adjacent territories have the same colour.

Certain assumptions must be made about the oligonucleotides used in the manipulations:

1) Under easily achievable conditions (temperature, pH, etc.) each oligonucleotide reliably forms stable hybrids with its Watson-Crick complement.
2) Under easily achievable conditions, each oligonucleotide reliably dissociates from its Watson-Crick complement.
3) Under neither of the conditions above does any oligonucleotide form hybrids with itself or another oligonucleotide (except it's complement),nor another oligonucleotide's Watson-Crick complement.

Error control is achieved mainly through logical operations, such as running all DNA samples showing positive results a second time to reduce false positives. Some molecular proposals, such as using DNA with a peptide backbone for stability, have also been recommended.

Applications to Biology, Chemistry & Medicine

Recently, Bartel and Szostak used the methods of combinatorial chemistry to make a pseudo-enzyme. The goal of their experiment was to find a molecule of RNA which would ligate two substrate molecules of RNA . They used a pool of approximately 4^ 25 random sequences of RNA to ligate the two substrate molecules, and after isolating the product of reaction, they were able to sequence the pseudo-enzyme.

The applications of combinatorial chemistry go far beyond this one example. In the above example, the pseudo-enzyme remained bound to the product, making it easy to isolate, but combinatorial chemistry can also be used when anchoring is not physically possible. Protocols have been proposed to find an RNA molecule that truly catalyzes (as an enzyme) the ligation of two DNA molecules. This sort of detection can also be used to isolate an endonuclease, or to find a drug which crosses a cell membrane and then binds a particular host membrane, and therefore cannot be anchored.

The Future

DNA computing is a recent advancement in computing, and for this reason, it is too early for either great optimism or great pessimism. As nucleic acid hybridization is akin to massively parallel computation, DNA computers may have an inherent, although currently unrealised, superiority over silicon-based computers. Thus computationally hard problems like NP-complete, may be soluble with linear increase in time for additional variables in an equation or nodes in a graph, rather than the polynomial or exponential increase in time that constrains silicon-based computers.

Early computers such as ENIAC filled entire rooms, and had to be programmed by punch cards. Since that time, computers have become much smaller and easier to use. It is possible that DNA computers will become more common for solving very complex problems, and just as PCR and DNA sequencing were once manual tasks, DNA computers may also become automated . In addition to the direct benefits of using DNA computers for performing complex computations, some of the operations of DNA computers already have, and perceivably more will be used in molecular and biochemical research.

In the future, research in molecular biology may provide improved techniques for manipulating macromolecules. Research in chemistry may allow for the development of synthetic designer enzymes . One can imagine the eventual emergence of a general purpose computer consisting of nothing more than a single macromolecule conjugated to a ribosome like collection of enzymes that act on it !

References

1. J. Colin Cox , Cohen D.S. ,and Ellington A.D.(1999) *Trends in Biotech* 17, 151-154
2. Adleman L.M. (1994), *Science* 266, 1021
3. Gifford D.K. (1994), *Science* 266, 993
4. http://crypto.stanford.edu/~dabo/biocomp.html
5. http://www.wi.LeidenUniv.nl/~pier/dna.html
6. http://wwwLmmb.ncifcrf.gov/~toms/molecularcomputation.html
7. http://www.cis.udel.edu/Research/DNAcomp/dnacomp.html