CHAPTER 6
RESULTS AND FINDINGS

6.1 INTRODUCTION

During the last two and a half years many different DNA models have been developed and analysed under several points of view. It turned out to be quite fruitful that many researchers with different interests have picked up that theme. Thereby complexity, stability and realizability played an important part. The results up to now show, that DNA-computers will not turn our insights of efficient computation from upside down. Nevertheless they are a powerful instrument for the implementation of parallel processes. The future of molecular machines could be in a combination with classical computers. Such systems should profit by the specific suitability of the components: well parallelizable tasks should be computed with DNA, whereas inherently sequential jobs should be done by silicon-based chips. It is still too early to expect high efficient realizations.

The possibility for practical DNA computing has a certain aesthetic appeal. It is based on nature’s way of generating complexity. However, there are several reasons, besides this intellectual fascination, why a DNA computer would be useful. Foremost among these is that DNA dented scale. Even in Adleman’s initial experiment, the number of strands was $10^{14}$ by a conservative estimate. Many researchers believe that operating on $10^{15}$ to $10^{20}$ strands in parallel is attainable with current technology. The individual operations themselves are very slow compared to electronic implementations. In the order of minutes to hours, Adleman’s experiment the biggest problem with DNA computing might well be that, in general, our expectations are too high. DNA computing is quite a novel idea. It has a long road ahead of it before attaining practical applicability, let alone becoming a mature technology. Along this road, a number of parameters have to be scaled up such as the size of the experiments, the speed of individual operations, the stability of the information carrier, the number of consecutive operations in an experiment, and the reliability of individual operations.
and of sequences of operations. It is impossible to tell where the road will lead without travelling it. However, even if DNA computing should turn out to go nowhere, the trip itself may be worth it due to the potential spin-offs.

DNA (Deribose Nucleic Acid) computing, also known as molecular computing is a new approach to massively parallel computation based on groundbreaking work by Adleman. DNA computing was proposed as a means of solving a class of intractable computational problems in which the computing time can grow exponentially with problem size (the 'NP-complete' or non-deterministic polynomial time complete problems). A DNA computer is basically a collection of specially selected DNA strands whose combinations will result in the solution to some problem, depending on the problem at hand. Technology is currently available both to select the initial strands and to filter the final solution. DNA computing is a new computational paradigm that employs (bio)molecular manipulation to solve computational problems, at the same time exploring natural processes as computational models. In 1994, Leonard Adleman at the Laboratory of Molecular Science, Department of Computer Science, University of Southern California surprised the scientific community by using the tools of molecular biology to solve a different computational problem. The main idea was the encoding of data in DNA strands and the use of tools from molecular biology to execute computational operations. Besides the novelty of this approach, molecular computing has the potential to outperform electronic computers. For example, DNA computations may use a billion times less energy than an electronic computer while storing data in a trillion times less space. Moreover, computing with DNA is highly parallel: In principle there could be billions upon trillions of DNA molecules undergoing chemical reactions, that is, performing computations, simultaneously.

L. M. Adleman launched the field of DNA computing with a demonstration in 1994 that strands of DNA could be used to solve the Hamiltonian path problem for a simple graph. He also identified three broad categories of open questions for the field. First, is DNA capable of universal computation? Second, what kinds of algorithms can DNA implement? Third, can the error rates in the manipulations of the DNA be controlled enough to allow for useful computation? In the two years that have followed, theoretical work has shown that DNA is in fact capable of universal computation. Furthermore, algorithms for solving interesting questions, like breaking
the Data Encryption Standard, have been described using currently available technology and methods. Finally, a few algorithms have been proposed to handle some of the apparently crippling error rates in a few of the common processes used to manipulate DNA. It is thus unlikely that DNA computation is doomed to be only a passing curiosity. However, much work remains to be done on the containment and correction of errors. It is far from clear if the problems in the error rates can be solved sufficiently to ever allow for general-purpose computation that will challenge the more popular substrates for computation. Unfortunately, biological demonstrations of the theoretical results have been sadly lacking. To date, only the simplest of computations have been carried out in DNA. To make significant progress, the field will require both the assessment of the practicality of the different manipulations of DNA and the implementation of algorithms for realistic problems. Theoreticians, in collaboration with experimentalists, can contribute to this research program by settling on a small set of practical and efficient models for DNA computation.

It may be noticed that for an undirected graph $G=(V,E)$ and a positive integer $k$, $k \leq |V|$, the vertex cover problem is to find a vertex subset $V'$ of size $k$ that satisfies: Each edge in $E$ is incident to at least one vertex in $V'$. We give a DNA solution to the vertex cover problem through designing an improved polynomial transformation from the vertex cover problem to the Hamiltonian circle problem. It first constructs the improved cover subgraph of each edge in $G$, which has 4 vertices and 4 edges instead of 12 vertices and 14 edges in the previous method. For each vertex, the improved cover subgraphs of the edges incident to it are linked together to form one sub path; for each selection vertex, the start and end points of each sub path are linked to it. The obtained graph $G'$ has a Hamiltonian circle if and only if $G$ has a vertex cover of size $k$. Thus, based on Adleman’s experiment of solving the Hamiltonian path problem, we give a DNA solution to the vertex cover problem. In 1994, Adleman[1] solved a 7-vertex instance of the Hamiltonian path problem by means of the molecular biology techniques. The pioneering work and some other researches provide a promising computation paradigm that uses DNA molecules to solve computational problem, especially NP-complete problem. Lipton proposed a DNA computing method for the SAT problem; In 1997, Ouyang et al[3] designed a DNA computing method to solve the maximal clique problem, etc. We also gave a DNA computing method for the
Chinese postman problem by means of general edge graph[4] and a DNA computing method for the travelling salesman problem by means of relative length graph. For an undirected graph \( G = (V, E) \) and a positive integer \( k \), the vertex cover problem is to find a vertex subset \( V' \) of size \( k \) that satisfies: \( V' \) subset of \( V \) and each edge in \( E \) is incident to at least one vertex in \( V' \). The vertex cover problem has been proved to be \( NP \)-complete, and there are deterministic algorithms and DNA algorithm of solving it. We have studied that scientist have present a DNA solution based on hybridization to the vertex cover problem by means of an improved polynomial transformation. Just as other arithmetical operations in silicon-based computer are implemented through converting them into the addition operation, other operations in DNA-based computer should be implemented through converting them into several basic DNA operations. The proposed DNA solution to the vertex cover problem provides a foundation in this aspect of DNA computing.

Researchers have convert the vertex cover problem to the Hamiltonian circle problem in a polynomial time. Thus, based on the basic operations in Adleman’s experiments[1] for the Hamiltonian path problem, we give an improved DNA solution to the vertex cover problem. Just as other arithmetical operations in silicon-based computer are implemented through converting them into the addition operation, other operations in DNA-based computer should be implemented through converting them into several basic DNA operations.

6.2 ANALYSIS OF DNA COMPUTING MODELS

Objective of our research is to study and analyze various specific model of DNA computing to obtain a Generalized model. Various DNA computing model have been developed. Some of there problem specific, such as [Aldeman,1994]. Compared with electronic computers, these model show potential advantage in solving the hard problems. due to the highly parallel characteristics of DNA operation, the corresponding DNA algorithm scale well in the size of the problem.
### Results and Findings

<table>
<thead>
<tr>
<th>Sno</th>
<th>Model</th>
<th>Objective</th>
<th>Algorithm</th>
<th>Limitation</th>
<th>Scientist</th>
<th>Findings</th>
</tr>
</thead>
</table>
| 1   | Filtering Model | A computation consists of a sequence of operations on finite multi-sets of strings. It is normally the case that a computation begins and terminates with a single multi-set. Within the computation, by applying legal operations of a model | 1. HPP  
2. Vertex Colouring  
3. SubGraph Isomorphism  
4. Maximum Clique Problem  
5. Maximum Independent Set | It is not believed they provide the full algorithmic computational power of a Turing Machine. Without the availability of string editing operations, it is difficult to see how the transition from one state of the Turing Machine to another may be achieved using DNA | Adleman  
Lipton  
Amos/Gibbons/Hodges  
Karp/Kenyon/Waarts  
Liu/Guo  
Roweis/Winfree | 1. HPP = O(n)  
2. Vertex Colouring = O(n)  
3. SubGraph Isomorphism = O(|V|)  
4. Maximim Clique Problem = O(|V|)  
5. Maximum Independent Set = O(|V|) |
| 2   | Splicing Model | Concern the simulation of nondeterministic Turing Machines and the simulation of Parallel Random Access Machines (specially CREW PRAMs). | 1. Reif’s PAM model | The quest for algorithms which proceed through polynomialized sets of strings (i.e., volumes of DNA). It is clear that the naive, filtering approach will not sustain such algorithms, since they rely upon the existence of an exponential-sized initial set of strings | Reif  
Freund/Kari/Paun | Reif’s PAM model = O(logT) |
| 3   | Constructive Model | Propose as a computational tool the tendency of DNA structures to self-assemble. They show that the self-assembly of DNA molecules into two dimensional sheets or three dimensional solids is a powerful model, capable of universal computation by virtue of the fact that it simulates a one-dimensional cellular automaton | 1. Boolean Circuit Model | experimental investigations of the power of self-assembly are still at a very preliminary stage. Winfree et al. show that, in principle, the two dimensional self-assembly model is experimentally implementable | Ogihara/Ray  
Baum/Boneh  
Guarnieri/Fliss/ Bancroft  
Winfree/Yang/Seeman | Self Assemble  
and  
Boolean Circuit |

**Table 6.1 Analysis of DNA computing Models**
There are several reasons to pursue the construction of molecular scale computers. One of the most obvious is just following the trend of miniaturization, advocated already by Feynman which has been present in microelectronics over the last four decades. This tendency was first recognized by Moore and is now known as Moore’s law. An economic principle rather than a law of nature, it states that transistor sizes will continue to shrink so the space they occupy halves roughly every two to three years. This leads to the possibility of increasingly complex logic chips, higher capacity memory chips and lower switching times. Current lithographic technology produces microchips with defining details of only 90 nanometres (meaning that some parts are of even smaller dimensions). If Moore’s law is made to hold much longer, transistor sizes will eventually reach the scale of individual molecules and atoms. It is far from certain that it will be possible to construct integrated circuits of silicon-based solid state transistors using familiar ‘top-down’ technology (using light-directed lithography), and if so, whether they will be functional. Both quantum phenomena and increasing heat generation appear prohibitive for the persistence of the trend. A recent technology which hopes to deal with these problems is molecular electronics, which tries to replace conventional electronic elements (such as semiconductor transistors and wires) with molecules (Tour, 2000). Most of the components considered are organic molecules or carbon nanotubes and even biological macromolecules are promising, as in the proposed light-addressable rhodopsin memory. Manufacturing techniques for molecular electronics and nanotechnology are generally ‘bottom-up’, in which individual components arrange themselves through local interactions.
Results and Findings

Figure 6.1. DNA Vs Silicon.

Defining feature sizes produced in mass production silicon lithography. Circles indicate production processes employed for microprocessor production (Intel, 2004), and squares represent technology projections up to 2016 by the International Technology Roadmap for Semiconductors (2003). The line indicates the Moore’s law trend of miniaturization. Extrapolation predicts molecular scale transistors by the 2030s, illustrated here with the 2 nm helix dimensions of DNA.

(self-assembly). However, the general functionality that is aimed for is still very similar to solid state electronics: elements should act as switches, pass electrons, and have permanent and definable contacts with other components. Another reason to pursue the construction of molecular scale computing devices is their scale. Some applications may simply call for very tiny, but not necessarily powerful computers. Finally, molecules may provide ways to implement completely different computing architectures. All current computers are still largely based on variants of the traditional von Neumann architecture—a single logic processing unit, a single sequentially addressed memory, a control unit and a user interface, and consequences such as the distinction between hardware and software. While this design has proved hugely successful, it is not necessarily synonymous with a computer, and other designs may cover computing needs that are hard to achieve using conventional means. This notion can be illustrated with a trade-off: ‘A system cannot at the same time be effectively programmable, amenable to evolution by variation and selection,
and computationally efficient’. This certainly seems plausible when one compares von Neumann computers to biological systems. The former is multi-purpose, and very programmable. However, its use of space, time and energy resources is quite inefficient. Biological systems are lacking in programmability and general control, but through superior adaptability are able to efficiently solve complex problems. Both systems are extremes in this trade-off, and if it holds, it is conceivable that some middle ground exists for powerful and practical molecular computers.

The work described was presented at the International Conference on Bio-Computing and Emergent Computation [6]. Here we examine complexity issues in DNA computation. We believe that these issues are paramount in the search for so-called “killer applications”, that is, applications of DNA computation that would establish the superiority of this paradigm over others in particular domains. An assured future for DNA computation can only be established through the discovery of such applications. We demonstrate that current measures of complexity fall short of reality. Consequently, we define a more realistic model, a so-called Tough model of computation which provides better estimates of the resources required by DNA algorithms. We also compare the complexities of published algorithms within this new model, extant model which is commonly (often implicitly) assumed. Following the initial promise and enthusiastic response to Adleman’s seminal work [2] in DNA computation, progress towards the realisation of worthwhile computations in the laboratory has become stalled. One reason for this is that the computational paradigm employed by Adleman, and generalised by the theoretical work of others [52, 66], relies upon filtering techniques to isolate solutions to a problem from an exponentially sized initial solution of DNA. This volume arises because all possible candidate solutions have to be encoded in the initial solution. As Hartmanis points out in [42], the consequence is that, although laboratory computations should work for the smallest problem sizes, the experiments do not realistically scale because vast amounts of DNA are required to initiate computations with even modest problem size. For example, Hartmanis shows that a mass of DNA greater than that of the earth would be required to solve a 200-city instance of the Hamiltonian Path Problem. If practitioners of DNA computation insist on this mode of computation, there can be no hope of discovering so-called killer applications, that is, applications of DNA
computation that would establish the superiority of this paradigm over others in particular domains. An assured future for DNA computation can only be established through the discovery of such applications. It is not inherently the case that exponentially sized volumes of DNA need be used in DNA computation. Indeed, polynomially sized computations have been (at least in theory) described (e.g., in [61]). Clearly, if exponentially sized volumes are to be avoided, then an alternative algorithmic paradigm to that employed by Adleman in [2] is required. Such a successful paradigm is always likely to emulate traditional computations which construct individual solutions rather than sift them out of a vast reservoir of candidates. It might still be argued that the “exponential curse” could not, even then, be avoided for the so-called NP-complete problems [31]. If an exact solution is required for any of these, then (employing any extant algorithm) exponential sequential running time is required. A DNA computation, in seeking to reduce this to sub-exponential parallel running time, will certainly require an exponential volume of DNA. However, in general, no-one sensibly seeks exact solutions to the NP-complete problems. In traditional computation, we either employ heuristics to obtain approximate answers or use randomised methods to obtain exact solutions with high probability. These revised algorithmic lead to solutions within polynomial sequential time. Such a view should also be taken for these problems within DNA computation, that is, we should use algorithms which do not inherently require exponential resources. It is unlikely to be enough, in the quest for killer applications, to simply have polynomial-volumed computations. We ought, at the same time, to ensure that the vast potential for parallelism is employed to obtain rapid computations. The view taken by the silicon-based parallel computing community [32] is that efficient parallel algorithms, within the so-called Parallel Random Access Machine (P-RAM) model of computation, should have polylogarithmic running time (and use a polynomial number of processors). Problems for which such solutions exist define the complexity class NC. If DNA computation is to compete within this domain, then we should clearly also look for polylogarithmic running times within polynomially-volumed computations. At the present time, no-one has described (even theoretically) DNA computations which run in polylogarithmic time using a polynomial volume of DNA. The discovery of such solutions might well provide candidates for “killer applications”. Regardless of the problem considered, it is unlikely to provide a “killer
applications” unless the computational resources required for a DNA computation (the product of the running time and volume of DNA required) match those needed for a conventional computation (the product of the running time and the number of processors used). For such a combination of resources, the DNA computation might well provide feasible solutions for problem sizes far greater than can be achieved by conventional computation. It is clearly crucial, especially when judging the candidacy of a proposed DNA computation for the role of “killer applications” to have a grasp of the computational resources that it requires. It also the case that there is not an agreed model of computation in the literature within which we may agree what the required resources are for any particular computation. Here we attempt to address these issues in a realistic way. Traditional computational complexity theory [4, 31] is concerned with quantifying the resources (generally time and space) needed to solve computational problems. Meaningful analysis of the complexity of algorithms may only take place in the context of an agreed model of computation, or machine model. Many different machine models have been proposed in the past, including the Deterministic Turing Machine, Boolean circuit [25, 41] and P-RAM [28, 32]. The nascent field of DNA computing also suffers from the problem of proliferation of machine models. Several models have been proposed, within which we may construct algorithms for the solution of computational problems. However, complexity analyses of algorithms within different models of DNA computation are meaningless, since there are no uniform definitions of the concepts of time and space. Furthermore, if we are to compare a DNA-based model with a more traditional machine model, we require a way of demonstrating equivalence between the two.

6.3 ANALYSIS OF GENERALIZED MODEL

The Generalized model has been developed to compute the NP complete problems viz. Hamiltonian Path problem, Maximum clique problem, Sub Graph Isomorphism, Maximum Independent Set and 3-Vertex colouring problem. In this model the each of the above mentioned algorithm has the result of permutation as the input it. The permuted out put is then processed according to the algorithms as can be seen below.
Attempts have been made to characterise DNA computations using traditional measures of complexity, such as time and space. Such attempts, however, are misleading due to the nature of the laboratory implementation of the computation. We first examine these algorithms from a time complexity standpoint. Most extant models quantify the time complexity of DNA-based algorithms by counting the number of “biological steps” required to solve the given problem. Such steps include the creation of an initial library of strands, separation of subsets of strands, sorting strands on length, chopping and ligating strands. Within these models, operations such as those described above are considered to be atomic actions performed in constant time.

6.3.1 The fragile model and Tough Model

Here we recall [7] the basic legal operations on sets within what we now refer to as the weak model. The operation set described here is constrained by biological feasibility, but all operations are currently realisable with current technology.
1. Remove(U, {Si}). This operation removes from the tube U, in parallel, any string which contains at least one occurrence of any of the substrings Si.

2. Union(∪i, U). This operation, in parallel, creates the tube U which is the set union of the tubes Ui.

3. Copy(U, Ui). In parallel, this operation produces a number of copies, Ui, of the tube U.

4. Select(U). This operation selects an element of U uniformly at random, if U is the empty set then empty is returned.

From the point of view of establishing the parallel time complexities of algorithms within the model, these basic operations are assumed to take constant time. This assumption has been commonly made by many authors in the literature [7, 52, 61]. However, these operations are frequently implemented in such a way that it is difficult to sustain this claim. For example, the union operation consists of pouring a number of tubes into a single tube, and this number is usually, in some way, problem size dependent. Assuming that in general we have a single laboratory assistant, this implies that such operations run in time proportional to the problem size. Obviously, in the general case, a single laboratory assistant may not pour n tubes into one tube in parallel, nor may s/he split the contents of one tube into n tubes in parallel. This observation, if we are to be realistic in measuring the complexity of DNA computations, requires us to introduce the following constant time atomic operation:

5. Pour(U, U'). This operation creates a new tube, U, which is the set union of the tubes U and U'.

As we have observed, the pour operation is a fundamental component of all compound operations. It therefore follows that more realistic analyses of the time complexities of algorithms may be obtained by taking this operation into consideration. The Tough model

In what follows we refine the fragile model just described. We assume that the initial tube (which takes at most linear time to construct) is already set up. The pour operation is fundamental to all compound operations within our fragile model. We must therefore reassess the time complexity of these operations. The remove operation requires the addition to U of
1. i tubes containing primers, and
2. A tube containing restriction enzymes

This operation is inherently sequential, since there must be a pause between steps 1 and 2 in order to allow the primers to anneal correctly. Therefore, the remove operation takes $O(i)$ time. Creating the union of $i$ tubes is an inherently sequential operation, since the technician must first pour $U_1$ into $U$, then $U_2$, and so on, up to $U_i$. Rather than taking constant time, the union operation actually takes $O(i)$ time. It is clear that the copy operation may be thought of as a reverse-union operation, since the contents of a single tube $U$ are split into many tubes, $\{U_i\}$. Therefore, copy takes $O(i)$ time.

Complexity comparisons in the fragile and Tough models

In this section we compare time complexities for algorithms previously described [7] within both the fragile and Tough models.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Fragile</th>
<th>Tough</th>
</tr>
</thead>
<tbody>
<tr>
<td>Three Coloring</td>
<td>$O(n)$</td>
<td>$O(n)^2$</td>
</tr>
<tr>
<td>Hamiltonian Path</td>
<td>$O(1)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>Sub graph isomorphism</td>
<td>$O(n)$</td>
<td>$O(n)^2$</td>
</tr>
<tr>
<td>Maximum Clique</td>
<td>$O(n)$</td>
<td>$O(n)^2$</td>
</tr>
<tr>
<td>Maximum Independent set</td>
<td>$O(n)$</td>
<td>$O(n)^2$</td>
</tr>
<tr>
<td>Permutation</td>
<td>$O(n)$</td>
<td>$O(n)^2$</td>
</tr>
</tbody>
</table>

Table 6.2 Complexity Analysis

In February 1995, one of the founding fathers of the theory of computational complexity (q.v.) published a short paper on DNA computing. In his article, titled “On the Weight of Computations”, Juris Hartmanis sounded a cautionary note amidst the growing interest surrounding DNA computing. By calculating the smallest amount of DNA required to encode all possible paths through a graph, he calculated that Adleman’s experiment, if scaled up from 7 cities to 200 cities, would require an initial set of DNA strands that would weigh more than the Earth. Hartmanis was quick to point out the value of the initial work – “Adleman’s molecular solution of the
Hamiltonian path problem is indeed a magnificent achievement and may initiate a more intensive exploration of molecular computing and computing in biological systems." However, he also emphasised the long-term futility of hoping that DNA-based computers could ever beat silicon machines in this particular domain. Soon after Adleman’s experiment, hopes were expressed that a lot of the promise of molecular computers could be derived from their massive inherent parallelism – each operation is performed at the same time on trillions of DNA strands. However, as Turing showed, computers are not limited to any particular physical construction, and a DNA computer will suffer just as much as its silicon counterpart from the “exponential curse”. If we require a molecular algorithm for a difficult problem to run in reasonable time then there is a fundamental requirement (unless P is shown to be equivalent to NP) for an exponential amount of space (in this case, the amount of DNA required). As Hartmanis observed, “...the exponential function grows too fast and the atoms are a bit too heavy to hope that the molecular computer can break the exponential barrier, this time the weight barrier.” This view would now seem to largely reflect the community consensus. As Ogihara and Ray argued in 2000, “It is foolish to attempt to predict the future of technology, but it may be that the ideal application for DNA computation does not like in computing large NP problems” [30]. In an interview in 2002, nanotechnologist Ned Seeman stated that “I am not a computer scientist, but I suspect it [molecular computing] is not well suited to traditional problems. I certainly feel that it is better suited to algorithmic self-assembly and biologically-oriented applications” [37]. That is not to say that molecular-based computations do not have a future, and in the final Section we briefly discuss some possible future directions that molecular computing may take (and offer pointers to other articles in the current volume).