A Surface-Based DNA Algorithm for the Expansion of Symbolic Determinants

Z. FRANK QIU and MI LU

Department of Electrical Engineering
Texas A&M University
College Station, Texas 77843-3128, U.S.A.
{zhiquan, mlu}@ee.tamu.edu

Abstract. In the past few years since Adleman's pioneering work on solving the HPP(Hamiltonian Path Problem) with a DNA-based computer [1], many algorithms have been designed on solving NP problems. Most of them are in the solution bases and need some error correction or tolerance technique in order to get good and correct results [3] [7] [9] [11] [21] [22]. The advantage of surface-based DNA computing technique, with very low error rate, has been shown many times [12] [18] [17] [20] over the solution based DNA computing, but this technique has not been widely used in the DNA computer algorithms design. This is mainly due to the restriction of the surface-based technique comparing with those methods using the DNA strands in solutions. In this paper, we introduce a surface-based DNA computing algorithm for solving a hard computation problem: expansion of symbolic determinants given their patterns of zero entries. This problem is well-known for its exponential difficulty. It is even more difficult than evaluating determinants whose entries are merely numerical [15]. We will show how this problem can be solved with the low error rate surface-based DNA computer using our naive algorithm.

1 Introduction

Although there are a flood of ideas about using DNA computers to solve difficult computing problems [2] [16] [19] [15] since Adleman [1] and Lipton [16] presented their ideas, most of them are using DNA strands in solution. They all take advantage of the massive parallelism available in DNA computers as one liter of water can hold 10^{22} bases of DNA strands. Because they all let DNA strands float in solution, it is difficult to handle samples and strands may get lost during some bio-operations.

A well developed method, in which the DNA strands are immobilized on a surface before any other operations, is introduced to DNA computing area by Liu [18]. This method, which is called surface-based DNA computing, first attaches a set of oligos to a surface (glass, silicon, gold, etc). They are then subjected to operations such as hybridization from solution or exonuclease degradation, in order to extract the desired solution. This method greatly reduces losses

of DNA molecules during purification steps [18]. The surface-based chemistries have become the standard for complex chemical syntheses and many other chemistries.

Although the surface-based DNA computer has been demonstrated as more reliable with low error rate and easier to handle [8] [12] [18] [20], only a little research work about utilizing these properties of this kind of computer has been presented [12]. This happens mainly because when the oligos are attached to a surface, we lose flexibility due to the restriction that the oligos can not grow in the direction of the attachment on the surface. In order to take advantage of the new mature method, algorithms of surface-based computing need to be developed.

In this paper, we present a new algorithm to be implemented on a surface-based DNA computer that will take fully advantage of these special properties of low error rate. We will use the expanding symbolic determinants problem as an example to show the advantage of our algorithm comparing with an existing algorithm based on general DNA computer in solution. Both algorithms will be able to solve some intractable problems that are unrealistic to be solved by current conventional electronic computers because of the intense computing power requirement. These problems are harder to solve than the problem in NP-Complete. Our algorithm has all the advantages of surface-based computers over an existing algorithm introduced in [15].

The rest of the paper are organized as follows: the next section will explain the methodology, including the logical and biological operations of surface-based DNA computers. The problem of expansion of symbolic determinants and our algorithm to solve it will be presented in section 3. Section 4 will analyze our new surface-based algorithm and the last section will conclude this paper.

2 Surface-Based Operations

In this section, we show the logical operations of DNA computers and then explain how these operations can be implemented on surface-based DNA computers. All these operations are necessary for solving the computational hard problem given in the next section.

A simple version of surface-based DNA computer uses three basic operations, mark, unmark, and destroy [17] plus the initialization and append operations introduced in [8]. The explanation of these operations are clearly shown as follows.

2.1 Abstract Model

1. **reset(S)**: It can also be called initialization. This step will generate all the strands for the following operations. These strands in set **S** can be generated to represent either the same value or different values according to the requirement.
2. **mark(C, S)**: All strands in set **S** satisfying the constraint **C** are identified as marked. A strand satisfies this constraint if and only if there is a number

represented by a strand with bit i agrees with the bit value specified in the constraint. If no constraint is given, all strands are marked [8].

3. **unmark()**: Unmark all the marked strands.
4. **delete(C)**: All strands satisfying condition **C** are removed from set **S** where $C \in \{marked, unmarked\}$.
5. **append(C, X)**: A word **X** represented by a strand segment is appended to all strands satisfying constraint **C**. **C** can be defined as marked or unmarked. If the constraint is marked strands, a word **X** is appended to all marked strands. Otherwise, a word **X** will be appended to all unmarked strands.
6. **readout(C, S)**: This operation will select an element in **S** following criteria **C**. If no C is given, then an element is selected randomly. We will use this step to obtain the expected answer.

2.2 Biological Implementation

In this section, we include the fundamental biological operations for our surface-based DNA computation model.

1. **reset(S)**: The initialization operation used here is different from those widely used biological DNA operations described in [1] [2] [4] [10] [19]. All the strands generated are attached to a surface instead of floating in the solution. In order to prepare all these necessary strands on the surface, both the surface and one end of the oligonucleotides are specially prepared to enable this attachment. A good attachment chemistry is necessary to ensure that the properly prepared oligonucleotides can be immobilized to the surface at a high density and unwanted binding will not happen on the surface [8] [18] [17].
2. **mark(C, S)**: Strands are marked simply by making them double-strands at the free end as all the strands on the surface are single strands at the beginning. These single strands being added in to the container will anneal with the strand segments that need to be marked. Partial double strands will be formed according to the Watson-Crick(WC) complement rule [1] [16] [6].
3. **unmark()**: This biological operation can be implemented using the method introduced in [8]. Simply washing the surface in distilled water and raising the temperature if necessary will obtain the resultant container with only single strands attaching to the surface. Because with the absence of salt which stabilizes the double strand bond, the complementary strands will denature from the oligonucleotides on the surface and will be washed away.
4. **delete(C)**: This operation can be achieved using some enzymes known as exonucleases which chew up DNA molecules from the end. Detail of this operation is introduced in [8]. Exonucleases exist with specificity for either the single or double stranded form. By picking different enzymes, marked (double strands) or unmarked (single strands) can be destroyed selectively.
5. **append(C, X)**: Different operations are used depending on whether marked or unmarked strands are going to be appended. If X is going to be appended

to all marked strands, the following bio-operations will be used for appending. Since marked strands are double stranded at the free terminus, the append operation can be implemented using the ligation at the free terminus. The method introduced in [8] can be used here. More details may be found in [8]. To append to unmarked strands, simple hybridization of a splint oligonucleotide followed by ligation as explained in [1] [16] may be used.

6. **readout(C, S)**: This procedure will actually extract out the strand we are looking for. There are many existing methods developed for solution based DNA computing readout [1] [6] [20]. In order to use these methods, we have to detach the strands from the surface first. Some enzymes can recognize short sequences of bases called restriction sites and cut the strand at that site when the sequence is double-stranded [8]. When the segment which is attaching to the surface contains this particular sequence, they can all be detached from the surface when the enzyme is added in.

3 Hard Computation Problem Solving

3.1 Expansion of Symbolic Determinants Problem

We will use the expansion of symbolic determinants problem as an example to show how our surface-based DNA computer can be used to solve hard problems that are unsolvable by currently electronic computers.

Problem: Assuming the matrix is n×n:

$$\begin{matrix} a_{11} & a_{12} & a_{13} & . & . & . & a_{1n} \\ a_{21} & a_{22} & & & & & \\ a_{31} & & . & & & & . \\ . & & & & & & \\ . & & & . & & & . \\ . & & & & & & \\ . & & & & . & & . \\ a_{n1} & . & & . & . & . & a_{nn} \end{matrix}$$

Generally, the determinant of a matrix is:

$$det(A) = \sum_{\sigma \in S_n} (-1)^{\sigma} A_{i_{\sigma_1} 1} \cdots a_{i_{\sigma_n} n} \tag{1}$$

where $S_n = (\sigma_1, \ldots, \sigma_n)$ is a permutation space [13] [5] [14]. A complete matrix expansion has n! items. When there are many zero entries inside, the expansion will be greatly simplified. We are going to solve this kind of problem–to obtain the expansion of matrices with many zero entries in them.

3.2 Surface-Based Algorithm

In order to make the process easy, we encode each item in the matrix a_{ij} by two parts: $(a_{ij})\_L$ and $(a_{ij})\_R$ while all the $(a_{kj})\_L's$ are with the same k but

different j and all the $(a_{ik})\_R's$ are with the same k but different i. Using this coding method, all items from the same row will have the same left half code, and all the items from the same column will have the same right code. It seems like that we construct a_{ij} by combining a_i and a_j. So, for example, a_{13} and a_{19} will be represented by the same left half segment but different right halves because they are in the same row but different columns. For another example, a_{14} and a_{84} will have the same right half but different left halves because they are in the same column but different rows. The following is an algorithm using the methodology of the previous section. It can be accomplished as follows:

a-1 **reset(S)**: A large amount of strands will be generated on the surface. All the strands are empty initially, they only have the basic header to be annealed to the surface.

a-2 **append(X, S)**: This will make the strands on the surface grow with X. The X here is $a_{ij} \neq 0$ while i is initially set as one and $j \in (1 : n)$. All the strands will grow by one unit and each will contain one item in the first row. After the append operation finishes, wash the surface to get rid of all unnecessary strand segment remained on the surface.

a-3 Repeat the above steps a-2 with i incremented by one until i reaches n. Now we have each strand should represent n units while each unit is an item from one row. So, each strand should have n items from n different rows.

a-4 **mark(X, S)**: We mark all strands containing X and X is initially set as a_i, the code for left half of each item representing the row number, with $i = 0$.

a-5 **delete(UM)**: Destroy all strands that are unmarked. This will eliminate those strands containing less than n rows because no matter what i is, it represents a row and every strand should contain it.

a-6 Repeat the above steps a-4 and a-5 n times with different i's while $i \in (1 : n)$. This will guarantee that one item from each row is contained in each strand.

a-7 Repeat the above steps a-4 and a-5 and a-6 with different a_j's, the codes for the right half of each item representing the column number, while $j \in (1 : n)$. This is used to keep only those strands that have items from each column and eliminate those that do not satisfy.

a-8 **readout(S)**: Readout all the remaining strands on the surface and they will be the answer for the expansion of our symbolic determinant. Each strand will contain one item from each row and one item from each column.

4 Analysis of the Algorithm

The complexity of this new algorithm is O(n) where n is the size of the matrix. In order to show the advantage of our surface-based DNA computer, we need to analysis the traditional method for expanding the symbolic determinants. The computing complexity of the traditional method is O(n!). Compare with the traditional method, we have solved a problem harder than NP within linear steps. The advantage of using DNA computer to solve the expansion of symbolic determinants problem is huge. Because the surface-based DNA technology is used, the DNA computer will be more reliable with low error-rate.

5 Conclusion

In this paper, we have proposed an algorithm to solve the expansion of symbolic determinants using surface-based model of DNA computer. Compare with other given applications of DNA computers, our problem is a more computation intensive one and our surface-based DNA computer will also reduce the possible errors due to the loss of DNA strands.

Further research includes expanding the application of surface-based DNA computing in order to make DNA computers more robust. With the goal of even lower error rate, we may combine the existing error-resistant methods [3] [7] [9] [11] [21] [22] and the surface-based technology to achieve better results.

References

1. Len Adleman. Molecular computation of solutions to combinatorial problems. *Science*, November 1994.
2. Martyn Amos. *DNA Computation.* PhD thesis, University of Warwick, UK, September 1997.
3. Martyn Amos, Alan Gibbons, and David Hodgson. Error-resistant implementation of DNA computations. In *Second Annual Meeting on DNA Based Computers*, pages 87–101, June 1996.
4. Eric B. Baum. DNA sequences useful for computation. In *Second Annual Meeting on DNA Based Computers*, pages 122–127, June 1996.
5. Fraleigh Beauregard. *Linear Algebra 3rd Edition.* Addison-Wesley Publishing Company, 1995.
6. D. Beaver. Molecular computing. Technical report, Penn State University Technical Report CSE-95-001, 1995.
7. Dan Boneh, Christopher Dunworth, Jeri Sgall, and Richard J. Lipton. Making DNA computers error resistant. In *Second Annual Meeting on DNA Based Computers*, pages 102–110, June 1996.
8. Weiping Cai, Anne E. Condon, Robert M. Corn, Elton Glaser, Tony Frutos Zhengdong Fei, Zhen Guo, Max G. Lagally, Qinghua Liu, Lloyd M. Smith, and Andrew Thiel. The power of surface-based DNA computation. In *RECOMB'97. Proceedings of the first annual international conference on Computational modecular biology*, pages 67–74, 1997.
9. Junghuei Chen and David Wood. A new DNA separation technique with low error rate. In *3rd DIMACS Workshop on DNA Based Computers*, pages 43–58, June 1997.
10. R. Deaton, R. C. Murphy, M. Garzon, D. R. Franceschetti, and Jr. S. E. Stevens. Good encodings for DNA-based solutions to combinatorial problems. In *Second Annual Meeting on DNA Based Computers*, pages 131–140, June 1996.
11. Myron Deputat, George Hajduczok, and Erich Schmitt. On error-correcting structures derived from DNA. In *3rd DIMACS Workshop on DNA Based Computers*, pages 223–229, June 1997.
12. Tony L. Eng and Benjamin M. Serridge. A surface-based DNA algorithm for minimal set cover. In *3rd DIMACS Workshop on DNA Based Computers*, pages 74–82, June 1997.

13. Paul A. Fuhrmann. *A Polynomial Approach To Linear Algebra.* Springer, 1996.
14. Klaus Jänich. *Linear Algebra.* Springer-Verlag, 1994.
15. Thomas H. Leete, Matthew D. Schwartz, Robert M. Williams, David H. Wood, Jerome S. Salem, and Harvey Rubin. Massively parallel dna computation: Expansion of symbolic determinants. In *Second Annual Meeting on DNA Based Computers*, pages 49–66, June 1996.
16. Richard Lipton. Using DNA to solve SAT. Unpulished Draft, 1995.
17. Qinghua Liu, Anthony Frutos, Liman Wang, Andrew Thiel, Susan Gillmor, Todd Strother, Anne Condon, Robert Corn, Max Lagally, and Lloyd Smith. Progress towards demonstration of a surface based DNA computation: A one word approach to solve a model satisfiability problem. In *Fourth Internation Meeting on DNA Based Computers*, pages 15–26, June 1998.
18. Qinghua Liu, Zhen Guo, Anne E. Condon, Robert M. Corn, Max G. Lagally, and Lloyd M. Smith. A surface-based approach to DNA computation. In *Second Annual Meeting on DNA Based Computers*, pages 206–216, June 1996.
19. Z. Frank Qiu and Mi Lu. Arithmetic and logic operations for DNA computer. In *Parallel and Distributed Computing and Networks (PDCN'98)*, pages 481–486. IASTED, December 1998.
20. Liman Wang, Qinghua Liu, Anthony Frutos, Susan Gillmor, Andrew Thiel, Todd Strother, Anne, Condon, Robert Corn, Max Lagally, and Lloyd Smith. Surface-based DNA computing operations: Destroy and readout. In *Fourth Internation Meeting on DNA Based Computers*, pages 247–248, June 1998.
21. David Harlan Wood. applying error correcting codes to DNA computing. In *Fourth Internation Meeting on DNA Based Computers*, pages 109–110, June 1998.
22. Tatsuo Yoshinobu, Yohei Aoi, Katsuyuki Tanizawa, and Hiroshi Iwasaki. Ligation errors in DNA computing. In *Fourth Internation Meeting on DNA Based Computers*, pages 245–246, June 1998.