AN IMPROVED PARALLEL DNA ALGORITHM OF 3-SAT

Wei Liu[1*], Shou Xia Sun[2], and Ying Guo[3]

*[*1] College of Mathematics and Information, Lu Dong University, Yantai 264025*
Email: liuyiwei1030@yahoo.com.cn
[2] Beijing University of Technology, The College of Applied Science, Beijing100022
Email: sundhouxia0@emails.bjut.edu.cn
[3] Department of Communication Engineering, Central South University, Changsha 410083
Email: yingguo1001@sjtu.edu.cn

ABSTRACT

There are many large-size and difficult computational problems in mathematics and computer science. For many of these problems, traditional computers cannot handle the mass of data in acceptable timeframes, which we call an NP problem. DNA computing is a means of solving a class of intractable computational problems in which the computing time grows exponentially with problem size. This paper proposes a parallel algorithm model for the universal 3-SAT problem based on the Adleman-Lipton model and applies biological operations to handling the mass of data in solution space. In this manner, we can control the run time of the algorithm to be finite and approximately constant.

Key words: Grouping strategy, 3-SAT, Parallel processing, NP problem, DNA computation

1 INTRODUCTION

Nowadays, we generate such large amounts of data that even modern computers cannot handle effectively. Researchers have realized that the defect of most computing lies in its serial operation, and some began to search for parallel algorithms. This work, however, did not solve the problem until 1994 when Adleman (1994) proposed a new computational approach called DNA computing. DNA computing solves computational problems by employing molecular biology-based techniques to manipulate DNA strings, so that the strings code information of the related algorithm. Further, the strings carry answer statements that can be recognized. Adleman first postulated that DNA strands could be applied for solving the HPP problem (Adleman, 1994). Lipton (1995) demonstrated that Adleman's experiment could be used to determine the satisfiability (SAT) problem. Since then, many groups have worked on NP-complete problems with the Adleman-Lipton model computing (Ouyang et al., 1997; Sakamoto et al., 2000; Faullhammer et al., 2000; Adleman et al., 2002; Liu et al., 2003; Dafa Li et al., 2003; Minyi Guob et al., 2003, 2004).

For a universal CNF Boolean formula F with n-variables, m-clauses instance of the 3-SAT problem, we apply biological operations to dispose 2^n DNA strands, which contain the data of solution space. In this paper, a DNA-based improved parallel algorithm model is proposed to reach the exact solution. The advantage of this approach is that we can control the algorithm to execute in a reasonable run time. In the process of handling the mass of data within DNA strands, we take a grouping strategy to reduce the time-complexity of the algorithm to constant.

The paper is organized as follows. In Section 2, we briefly introduce background about the SAT satisfiability problem and the structure of the probe module. Section 3 describes the improved DNA parallel grouping algorithm. Conclusions are drawn in Section 4.

2 BACKGROUND

In Section 2.1, we introduce background of the famous SAT problem, which is one of the most important optimization combinatorial problems. Section 2.2 introduces the structure of the probe modules of the Adleman-Lipton algorithm model.

2.1 The SAT problem

The SAT problem has both a theoretical and practical importance. On the one hand it constitutes the core of the computational complexity analysis, and on the other, it permits the indirect solution of many interesting problems. SAT stands for satisfiability, and it refers to the propositional logic problem. The SAT problem is the first one ever shown to be NP-complete. The 3-SAT problem is known as the hardest of all NP-complete problems, for which the fastest known sequential algorithms require exponential time. The problem has become the benchmark for testing the performance of DNA computation.

The SAT problem is to find assignments of a set of Boolean variables that produce correct output of a Boolean formula. It is a simple search problem, which is known to be one of the hardest NP problems. Every NP problem can be seen as the search for a solution that simultaneously satisfies a number of logical clauses, each composed of several variables (which can be true or false) connected by 'OR' statements. Consider a Boolean formula

$$F = (x \vee \neg y) \wedge (\neg x \vee z) \wedge (\neg y \vee \neg z)$$

Where '∧' , '∨'and '¬' are the logic 'AND,' 'OR,' and 'NOT' operations, respectively. Boolean variables *x*, *y*, and *z* are allowed to range over "true" and "false" values. A literal is a variable x_i or its negation $\neg x_i$. The 3-SAT problem is to decide whether a given 3-CNF formula can be found.

2.2 Probe modules

Probe modules are prepared as follows: a 5´-end Acrydite-modified oligonucleotides probe module for each clause C_t（$t = 1,2,...,m$) is prepared using the design protocol given below.

For each clause, a probe module is created by adding the corresponding Acrydite-modified probe to a polyacrylamide gel. If the t^{th} clause C_t contains the literal x_j then the corresponding Acrydite-modified probe $\overline{x}_j^T$, where $\overline{x}_j^T$ denotes the Watson-Crick complement of x_j . If the t^{th} clause C_t contains literal $\neg x_j$ then the corresponding Acrydite-modified probe $\overline{x}_j^F$ is added,where $\overline{x}_j^F$ denotes the Watson-Crick complement of $\neg x_j$.

3 DNA PARALLEL ALGORITHM

3.1 The coding of the database

Because of the mass of data and number of feasible solutions, we must use some strategy to reduce the complexity of the algorithm. This is done by introducing a grouping strategy (Liu et al., 2006) by which to design the coding of DNA sequences used in DNA computing. This coding strategy can thus deal with the database of library strands effectively. In order to achieve this, only four sub-libraries $S^{(i)}(i = 1,2,3,4)$ must be designed. All the library strands, which denote information about feasible solutions, must be synthesized before the algorithm starts. Every library sequence L_j has one of four unique device forms by which to implement the parallel algorithm model shown as follows:

$$S^{(1)}: \ 5'-(L_j,h)-3'$$

$$S^{(2)}: \ 3'-(\overline{h},L_j,h)-5'$$

$$S^{(3)}: \ 5'-(\overline{h},L_j,h)-3'$$

$$S^{(4)}: \ 3'-(\overline{h},L_j)-5'$$

Where: (1): L_j encodes all possible truth assignments of each sub-set $S^{(i)}(i = 1,2,3,4)$;

(2): $5'-(h)-3'$ denotes 5′-3′ sticker end of every library sequence L_j and $3'-(\overline{h})-5'$ denotes its Watson-Crick complement. $3'-(h)-5'$ denotes the 3′-5′ sticker end of every library sequence L_j and $5'-(\overline{h})-3'$ denotes its Watson-Crick complement.

(3): $S = \{S^{(1)}, S^{(2)}, S^{(3)}, S^{(4)}\}$.

3.2 Processing data by biochemical operation

For all feasible solutions of formula F, we can handle the great amount of data by the following biochemical operations:

Step 1: a) Distribute m tubes into $\lceil m/4 \rceil$ or $\lceil m/4 \rceil + 1$ groups. That is quaternion. b) For each group, put sub-set $S^{(1)}, S^{(2)}, S^{(3)}, S^{(4)}$ of S_1 into four different tubes, respectively. c) Put m different probes into the m different tubes $T_1, T_2, ..., T_{m_1}$ and refrigerate to $4°C$ simultaneously. Then the strands in all of test tubes perform hybridization reactions. d) The strands of S encoding truth assignments satisfying the corresponding clause are captured in the corresponding capture layer. e) Those strands encoding non-satisfying assignments can then be removed out of the test tubes with the buffer.

Step 2: Wash the tubes and then add a new buffer, and raise the temperature to $85°C$: The probe modules in the tubes melt the captured strands into the buffer. Then remove the probe modules.

Step 3: Integrate the contents of the group tubes into one of the four tubes, and then add T_4 DNA ligase. The strands in them undergo a link reaction under the function of T_4 DNA ligase and they will become duplex.

Step 4: Keep those DNA strands whose length is $4l+3k$ (l is the length of L_j, while k is length of sticker end) by polyacrylamide gel electrophoresis. That is, to separate out the longest length strands:

$$5' - \left(L_{j1}, D(h), L_{j1}, D(h), L_{j1}, D(h), L_{j1} \right) - 3'$$

where $D(h)$ denotes the duplex of the h sticker end.

Step 5: Raise the temperature to $95°C$ for each group tube so that the duplex strands in those tubes are denatured. As a result, these strands turn into single-strand form. Then regroup the remaining $[m/4]$ or $[m/4]+1$ test tubes after the first recycle into $[m/16]$ or $[m/16]+1$ groups.

Step 6: Put four probes into corresponding tubes of each group respectively. The four different tubes contain the corresponding four different probes as follows:

$$probe1 : 3' - \left(\overline{h} \right) - 5'$$
$$probe2 : 5' - \left(h \right) - 3'$$
$$probe3 : 3' - \left(h \right) - 5'$$
$$probe4 : 5' - \left(\overline{h} \right) - 3'$$

The four probes for each group will capture the corresponding four different forms of S respectively, while those strands not captured will be removed out of each tube with the buffer.

Step 7: Repeat step 2 to step 6 until there is only one test tube left. If there are only two or three tubes in one group in the course of one cycle, then randomly take any two or three probes in step 6. If there is just one tube in one group in the course of the cycle, then do not discard it, but put it directly into the next cycle.

Step 8: Identify the longest DNA strands by polyacrylamide gel electrophoresis from the last test tube. These are the answer strands for formula F.

Step 9: Raise the temperature of the last tube, and the strands captured by the probe modules will be denatured to form single strands. Then remove all the probe modules.

Step 10: Check whether any DNA molecular strands are left in the last tube. If there are some strands remaining, then extract them, as they are the answer strands. Next, use a PCR-amplification operation to amplify the answer strands. Finally, apply the “read” technique to verify the content of these answer strands.

4 CONCLUSION

The work in this paper proposes an improved universal DNA parallel algorithm of the 3-SAT problem. This algorithm model designs four forms for each data’s DNA strand so that we can implement the parallel self-assembly algorithm of DNA computing to dispose of the mass data strands. In this way, many intractable NP problems can be solved effectively because it uses a DNA parallel operation to dispose of the data instead of a serial computer operation. The advantages of this algorithm are that it can get constant time-complexity, the

tube number, which we use in the operation process, will reduce at a speed of $\log_3 m$, and the sum of the tubes which we use is m+3. However we must say that there is still a long way to go in the search for a universal DNA computer.

5 REFERENCES

Adleman, L. (1994) Molecular Computation of Solutions to Combinatarial Problems. *Science*, Vol. 266, 1021-1024.

Adleman, L. (2002) Solution of a 20-Varuable 3-SAT Problem on a DNA Computer. *Science 1026*.

Chang, W., et al. (2004) Towards solution of the set-splitting problem on gel-based DNA computing. *Future Generation Computer Systems 20*, 875-885.

Chang, W. & Guo, M. (2003) Solving the set cover problem and the problem of exact cover by 3-sets in the Adleman–Lipton model. *BioSystems 72*, 263–275.

Faullhammer, Lipton, Landweber, et al. (2000) Molecular computation: RNA solutions to chess problems. *Procedures of the National Academy of Sciences*. U.S.A.

Li, D, et al. (2003) Hairpin formation in DNA computation presents limits for large NP-complete problems. *BioSystems 72*, 203–207.

Lipton, R. (1995) DNA Solution of Hard Computational Problems. *Science 268*, 49-66.

Liu, Q., Wang, L., et al. (2003) DNA Computing on Surfaces. *Nature 403*, 175-179.

Liu, W., et al. (2006) Grouping parallel algorithm model of 3-SAT in DNA computing. *Impulsive Dynamic Systems and Applications 4*, 1533-1535.

Ouyang, Q., Kaplan, et al. (1997) DNA solution of the maximal clique problem. *Science 278*, 446–449.

Sakamoto, Hagiya, et al. (2000) Molecular Computation by DNA Hairpin Formation. *Science 288*, 1223-1226.