Available online at www.sciencedirect.com
SCIENCE DIRECT®
BioSystems 80 (2005) 233–250
BioSystems
www.elsevier.com/locate/biosystems

Fast parallel molecular solutions for DNA-based supercomputing: the subset-product problem

Michael (Shan-Hui) Ho*

Department of Information Management, Southern Taiwan University of Technology, Tainan County, Taiwan 710, ROC

Received 5 May 2004; received in revised form 10 November 2004; accepted 29 November 2004

Abstract

In this paper our main purpose is to give molecular solutions for the subset-product problem. In order to achieve this, we propose three DNA-based algorithms – parallel adder, parallel multiplier and parallel comparator – that formally verify our designed molecular solutions for the subset-product problem. We also show that Boolean circuits are not needed to perform mathematical operations on a molecular computer. Furthermore, this work indicates that the subset-product problem is solved and also presents clear evidence of the ability of molecular computing to perform complicated mathematical operations.

Keywords: Biological parallel computing; Molecular-based supercomputing; DNA-based supercomputing; NP-complete problem

1. Introduction

Feynman (1961) first proposed molecular computation in 1961, but his idea was not implemented by experiment for a few decades. In 1994 Adleman (Adleman, 1994) succeeded in solving an instance of the Hamiltonian path problem in a test tube, just by handling DNA strands. Lipton (1995) demonstrated that the Adleman techniques could be used to solve the satisfiability problem (the first NP-complete problem). Adleman and his co-workers (Roweis et al., 1999) proposed *sticker* for enhancing the Adleman–Lipton model.

Through advances in molecular biology (Sinden, 1994), it is possible to produce roughly 10^{18} DNA strands that fit in a test tube. Those 10^{18} DNA strands can also be applied for representing 10^{18} bits of information. Basic biological operations can be used to simultaneously operate 10^{18} bits of information. Or we can say that 10^{18} data processors can be executed in parallel. Hence, it becomes obvious that biological computing can provide a huge parallelism for dealing with problems in the real world.

In this paper, first we use *sticker* to construct solution spaces of DNA strands for the *subset-product*

* Tel.: +886 6 2533131x4300; fax: +886 6 2541621.
E-mail address: MHoInCerritos@yahoo.com.

0303-2647/$ – see front matter © 2004 Elsevier Ireland Ltd. All rights reserved.
doi:10.1016/j.biosystems.2004.11.009

problem. Then by using biological operations in the Adleman–Lipton model, we develop three DNA-based algorithms—parallel adder, parallel multiplier and parallel comparator for performing the functions of addition, multiplication and comparative instructions. We also show that using the biological operations in the Adleman–Lipton model for the sticker solution space solves the subset-product problem. Furthermore, this work presents clear evidence of the ability of molecular computing to solve the NP-complete problem with mathematical operations.

The paper is organized as follows: Section 2 introduces the Adleman–Lipton model in detail then this model is compared with other models. Section 3 introduces the DNA program to solve the subset-product problem for the sticker solution space. In Section 4, experimental results by simulated DNA computing are given. Conclusions are drawn in Section 5.

2. DNA model of computation

2.1. The Adleman–Lipton model

A deoxyribonucleic acid (DNA) is a polymer, which is strung together from monomers called deoxyribonucleotides (Sinden, 1994; Paun et al., 1998). Distinct nucleotides are detected only by their bases. These bases are abbreviated as *A*, *G*, *C* and *T*. Two strands of DNA can form (under appropriate conditions) a double strand, if the respective bases are the Watson–Crick complements of each other – *A* matches *T* and *C* matches *G*; also 3′ end matches 5′ end. The length of a single stranded DNA is the number of nucleotides comprising the single strand. Thus, if a single stranded DNA includes 20 nucleotides, we can say that it is a 20 mer (it is a polymer containing 20 monomers). The length of a double stranded DNA (where each nucleotide is base paired) is counted in the number of base pairs. Thus if we make a double stranded DNA from a single stranded 20 mer, then the length of the double stranded DNA is 20 base pairs, also written 20 bp. (For more discussions of the relevant biological background refer to Sinden, 1994; Boneh et al., 1996; Paun et al., 1998).

In the Adleman–Lipton model (Adleman, 1994; Lipton, 1995), *splints* were used to construct the corresponding edges of a particular graph of paths, which represented all possible binary numbers. As it stands, their construction indiscriminately builds all splints that lead to a complete graph. This is to say that hybridization has a higher probability of errors. Hence, Adleman and his co-workers (Roweis et al., 1999) proposed the sticker-based model, which was an abstract of molecular computing based on DNA with a random access memory as well as a new form of encoding the information.

The DNA operations in the Adleman–Lipton model (Adleman, 1994, 1996; Lipton, 1995; Boneh et al., 1996) are described below. These operations will be used for finding solutions of the subset-product problem.

The Adleman–Lipton model:

A (test) tube is a set of molecules of DNA (a multi-set of finite strings over the alphabet $\{A, C, G, T\}$). Given a tube, one can perform the following operations:

1. *Extract*: Given a tube P and a short single strand of DNA, S, the operation produces two tubes $+(P, S)$ and $-(P, S)$, where $+(P, S)$ is all of the molecules of DNA in P which contain S as a sub-strand and $-(P, S)$ is all of the molecules of DNA in P which do not contain S.
2. *Merge*: Given tubes P_1 and P_2, yield $\cup(P_1, P_2)$, where $\cup(P_1, P_2) = P_1 \cup P_2$. This operation is to pour two tubes into one, without any change in the individual strands.
3. *Detect*: Given a tube P, if P includes at least one DNA molecule we have 'yes', and if P contains no DNA molecule we have 'no'.
4. *Discard*: Given a tube P, the operation will discard P.
5. *Amplify*: Given a tube P, the operation, *Amplify*(P, P_1, P_2), will produce two new tubes P_1 and P_2 so that P_1 and P_2 are totally a copy of P (P_1 and P_2 are identical) and P becomes an empty tube.
6. *Append*: Given a tube P containing a short strand of DNA, Z, the operation will append Z onto the end of every strand in P.
7. *Append-head*: Given a tube P containing a short strand of DNA, Z, the operation will append Z onto the head of every strand in P.
8. *Read*: Given a tube P, the operation is used to describe a single molecule, which is contained in tube