Applied Mathematics and Computation 215 (2009) 2055–2062

Contents lists available at ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

A polynomial-time DNA computing solution for the Bin-Packing Problem

Carlos Alberto Alonso Sanches *, Nei Yoshihiro Soma

ITA, Computer Science Division, Brazil

ARTICLE INFO

Keywords:
DNA computing
Bin-Packing Problem
Parallel algorithm

ABSTRACT

It is suggested here an algorithm based on *stickers* for the DNA Computing model [S. Roweis, E. Winfree, R. Burgoyne, N. Chelyapov, M. Goodman, P. Rothemund, L. Adleman, A sticker-based model for DNA Computation, Journal of Computational Biology 5 (1998) 615–629] that solves the well known *Bin-Packing Problem* (BPP), that belongs to the class $\mathscr{NP}$-Hard in the strong sense, in a time bounded by $\mathscr{O}(n^2q)$, where n is the quantity of items and q the space requirements expressed in bits.

To the best of the authors' knowledge, this is the first polynomial time algorithmic solution for the BPP in such a model.

1. Introduction

The *Bin-Packing Problem* (BPP) is a classical discrete mathematics problem and imposes stern difficulties for solving it exactly in a *von Neumann* architecture based machine [14], since it also belongs to the class $\mathscr{NP}$-Hard *in the strong sense* [12]. However, if a different computational model is considered, it is possible to solve BPP in a time bounded by a polynomial in the size of the input. A seminal approach based on bio-inspired machine model came with Adleman [1], for a Hamiltonian Circuit Problem *instance*. In the sequence, Lipton [17] solved the SAT Problem, that is the very first problem proved to be *complete* in $\mathscr{NP}$. Those two works constitute a biological framework that is known as the *Adleman–Lipton* model. It is suggested here a polynomial time algorithm for solving the BPP that, to the best knowledge of the authors, is the first bio-inspired solution for that problem. The paper is organized as follows: Section 2 gives an overview of the major results in DNA models of computation; the BPP and the algorithm for solving it with its convergence are presented at Section 3. In Section 4 it is given the conclusions of the ideas here introduced.

2. DNA computing models

The Watson and Crick seminal result [21] on the information concerning the codification of any type of life is represented in the DNA (deoxyribonucleic acid). The DNA is a long polymer formed by units called nucleotides that connect among themselves by four different types of molecules called bases: adenine (A), cytosine (C), guanine (G) and thymine (T). To the context of this work, a nucleotide and its corresponding base are considered as the same element.

To form the DNA sequences, the nucleotides are joined among them by phosphate groups – bonds – that are asymmetric with respect of the geometry of each other, and they are referred as the 5′ (five prime) and 3′ (three prime) ends. The DNA double helix structure comes as a result of the annealing of complementary bases (A with T and C with G). The reverse process – melting – separate the double helix into two bases sequences. Moreover, a sequence of pieces of DNA is composed by genes.

* Corresponding author.
E-mail addresses: alonso@ita.br (C.A. Alonso Sanches), soma@ita.br (N.Y. Soma).

0096-3003/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.amc.2009.07.051

The human genome has near 3 billions base pairs of DNA. Each gene is composed from few thousands up to many millions of bases and it is located in a concrete position of a given chromosome. The bases sequences contains all the necessary information for the protein biosynthesis, that has an important role in the cell division.

Under the computational point of view, a DNA sequence can be considered as a string formed by the combination of four symbols: *A*, *G*, *C* and *T*. This implies that one can use four symbols to code information, being this quantity more than sufficient, since in a usual computer just two digits are involved.

The following basic operations can be carried out on DNA sequences by commercial available enzymes:

- *Cutting*. An enzyme *restriction endonuclease* permits to recognize a small portion of the DNA. Any double helix that contains it can be cut in that exact place.
- *Linking*. An enzyme *DNA ligase* allows to join the end of a DNA sequence with the beginning of another one.
- *Replication*. An enzyme *DNA polymerase* makes possible the DNA replication.
- *Destruction*. With the enzyme *exonuclease*, it is possible to eliminate certain DNA subsequences.

The term *DNA Computing* appeared in 1994 with Adleman [1]. A *DNA-computer* can be seen as a parallel SIMD (*Single Instruction stream, Multiple Data stream*) such that their "*biological processors*" work synchronously, but with no communication occurring between any two of them. An usual test tube has approximately 10^{18} DNA molecules and, in a DNA Kilogram, it can be stored a quantity of information equivalent to 10^{23} *bytes*. Therefore, even being near 10^9 times slower than their electronic computers counterparts, the sheer amount of processing cells, allied with the series of operations possible to be carried out over the molecules, can configure in a near future a feasible alternative to efficiently solve many computational problems, even the intractable ones.

In the last few years, a series of articles on both the codification and solution of difficult problems via *DNA Computing* appeared in the literature: *Maximal Clique* [19]; *3-SAT* [17,7]; *Subgraph Isomorphism* [15,16], *3-Coloring*, *Maximal Clique* and *Independent-Set* [2]; *Dominating-Set* [13]; *3-Dimensional Matching* and *Set-Packing* [3]; *Set-Covering* [4]; *Set-Partition* [6]; *Subset-Sum* [8]; *Knapsack* [9,10,18]; *Factoring Integers* [5]; *3-SAT*, *3-Coloring* and *Independent-Set* [11].

In spite of the fact that all of these results are theoretical ones, without the laboratory practical tests, it is possible to prove that the total number of biological operations necessary to solve those problems is bounded by a polynomial in terms of processing time. This indicates that this computational model maybe seen as an efficient alternative to solve problems belonging to the $\mathscr{NP}$ class.

2.1. The Adleman–Lipton model

The results in Adleman [1] and Lipton [17], as mentioned before, compose a system of biological operations that are known as the *Adleman–Lipton* model.

In this model, there exists a set of test tubes with DNA's sequences, i.e., a multi-set of finite *strings* over the alphabet $\{A, C, G, T\}$. The biological operations within this model, that form the basic steps of a *DNA-computer*, are defined by the following:

- *Extract*. Given a test tube T and a sequence S, this operation generates two tubes: $+(T,S)$ with all the sequences in T that had S as a subsequence, and $-(T,S)$ with the remaining sequences of T.
- *Merge*. Given two test tubes T_1 and T_2, this operation generates a new tube with the content of both, that is, it is equivalent to decant the contents of the T_1 and T_2 into a third one without modifying no molecule.
- *Detect*. Given a tube T, this operation returns the logic value `yes` if there is at least a DNA molecule in it, and `no` otherwise.
- *Discard*. Given a test tube T, this operation simply discards it.
- *Amplify*. Given a test tube T, *amplify*(T, T_1, T_2) produces two new test tubes T_1 and T_2 that are identical copies of T, and this latter one becomes empty.
- *Append*. Given a test tube T and a sequence S, *append*(T,S) affixes S at the end of each sequence in T.

One of the most important problems with these biological operations is the high error rate when of their executions. A large variety of suggestions appeared to overcome such type of error and the most relevant one comes with the use of *stickers* [20], summarized next.

2.2. The sticker-based model

The *sticker-based* model [20] simulates a computer with a binary memory in the following way:

- There is a new way to present the information: each *bit* corresponds to a subsequence of k nucleotides, called a *sticker*.
- The computer memory is formed by a sequence of *stickers*: those that do have a match can be associated to a *bit* with the value 1. If such a matching does not occur, then the corresponding *bit* receives the value 0.
- Each *sticker* differs significantly from the others, and it exactly matches with just another one.