ELSEVIER

BioSystems 70 (2003) 73–78

Bio Systems

www.elsevier.com/locate/biosystems

The general form of 0–1 programming problem based on DNA computing[☆]

Yin ZhiXiang[a,b,*], Zhang Fengyue[b], Xu Jin[b]

[a] *Department of Mathematics and Physics, Anhui University of Science and Technology, Anhui 232001, China*

[b] *Department of Control Science and Engineering, Huazhong University of Science and Technology, Huibei Wuhan 430074, China*

Received 7 December 2001; received in revised form 2 January 2003; accepted 27 February 2003

Abstract

DNA computing is a novel method of solving a class of intractable computational problems, in which the computing speeds up exponentially with the problem size. Up to now, many accomplishments have been made to improve its performance and increase its reliability. In this paper, we solved the general form of 0–1 programming problem with fluorescence labeling techniques based on surface chemistry by attempting to apply DNA computing to a programming problem. Our method has some significant advantages such as simple encoding, low cost, and short operating time.

Keywords: DNA computing; 0–1 programming problem; Feasible solution; Fluorescence labeling technique

1. Introduction

Feynman (1961) gave a visionary talk describing the possibility of building computers that were sub-microscopic. Despite remarkable progress in computer miniaturization, this goal has yet to be achieved. Computer scientists rank computational problems in three classes: easy, hard, and incomputable (Ouyang et al., 1997). One of the major achievements of computer science in the last two decades is the understanding that many important computational search problems are NP-complete, and thus are unlikely to have efficient algorithms that solve the problem exactly. Adleman (1994) showed that DNA can be used to solve a computationally hard problem, the directed Hamiltonian path problem (DHPP), and demonstrated the potential power of parallel, high-density computation by molecules in solution. This parallelism allows DNA computers to solve larger, harder problems such as NP-complete problems in linearly increasing time, in contrast to the exponentially increasing time required by an electronical computer. After Adleman initiated the field of DNA computing in 1994, Lipton (1995) proposed DNA experiments to solve the SAT problem. Ouyang et al. (1997) presented a molecular biology-based experimental solution to the “maximal clique” problem. Liu et al. (2000) designed a DNA model system where multi-based encoding strategy is used in a one-word approach to surface-based DNA

[☆] The project supported CNSF (Grant number: 60274026; 60174047).

[*] Corresponding author. Tel.: +86-27-8754-3563.
E-mail address: zxyin@mail.hust.edu.cn (Y. ZhiXiang).

0303-2647/03/$ – see front matter © 2003 Elsevier Science Ireland Ltd. All rights reserved.
doi:10.1016/S0303-2647(03)00053-4

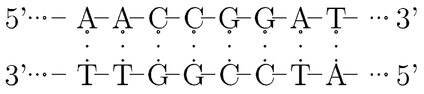
5'···– A–A–C–C–G–G–A–T– ··· 3'
3'···– T–T–G–G–C–C–T–A– ··· 5'

Fig. 1. Illustration of antiparallel configuration.

computation. Wu (2001) analyzed and improved their surface-based method. All of these efforts made use of molecular biology and demonstrated the feasibility of carrying out computation at the molecular level. One of the formal frameworks for molecular computations was provided by Head (1987) with a splicing system that provides a theoretical foundation for computing based on DNA recombination. In this paper, the general form of the 0–1 programming problem was solved by fluorescence labeling techniques based on surface chemistry. Despite significant progress, several problems remain and need to be resolved. The first, for a complex issue, there is a need for a great amount of DNA in decoding, which is difficult to achieve. Second, DNA computing is inaccurate, which can be caused by inaccurate hybridization, the effect of DNA secondary structure, and the inaccuracy of an experiment, and all of these can affect the result of DNA computing.

DNA is a high-molecular weight compound. Its basic composition includes one phosphate group, one deoxyribose sugar, and one nitrogenous base. There are four kinds of nitrogenous bases: adenine (A), guanine (G), cytosine (C), and thymine (T). DNA is a double-helix consisting of two single strand deoxynucleotide chains running in an antiparallel configuration (Fig. 1). After determining the precise structure of DNA, many experimental methods have been invented including annealing, amplifying, melting, separating, cutting, ligating, and so on, which can be used to help to discover the mechanisms of information storage and output.

For terminologies and notations not defined in this paper, the readers are referred to Bondy and Murty (1976).

2. The 0–1 programming problem

An integer programming problem in which all variables are required to be integer is called a *pure integer programming problem*. If some variables are restricted to be integers and some are not then the problem is a *mixed integer programming problem*. The case where the integer variables are restricted to be 0 or 1 comes up surprisingly often. Such problems are called *pure* (*mixed*) 0–1 *programming problems* or *pure* (*mixed*) *binary integer programming problems*.

The 0–1 programming problem is a special form of an integer programming problem, in which the value of variable x_i is only 0 or 1. In this condition, x_i can be referred to as either a “binary” or “0–1” variable. The general form of 0–1 programming problem is:

$$\max(\min) z = c_1x_1 + c_2x_2 + \cdots + c_nx_n$$

$$\times \begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \le (=, \ge) b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \le (=, \ge) b_2 \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \le (=, \ge) b_m \\ \qquad x_1, x_2, \dots, x_n = 0, 1 \end{cases}$$

0–1 programming problem is an important problem in operation research and has very widespread application. A wide variety of algorithms have been written to solve the 0–1 programming problem including methods of exhaust algorithm, invisible enumeration (Christos and Papadimitriou, 1982), and others (Gass, 1984). The computing time of these methods increases exponentially with problem size. Otherwise, 0–1 programming problem and the satisfiability problem are both closely related, and 0–1 programming problem is a generalization of the satisfiability problem. Up to now, there have been many results for solving the satisfiability problem (Yoshida, 1999; Skamoto, 2000; Cukras et al., 1999; Braich et al., 2002). However, the model of 0–1 programming problem based on DNA computing has never been studied. In the paper, we solve the general form of 0–1 programming problem with DNA computing. When a_{ij} is a real number, both sides of the inequality are number of times to make every coefficient of a variable become an integer. When a_{ij} is an integer, $x_j + x_j + \cdots + x_j$ (where the number of x_j is $|a_{ij}|$) it is represented as $a_{ij}x_j$. Thus, the general form of 0–1 programming problem can be transformed into the following:

$$\max(\min) z = c_1x_1 + c_2x_2 + \cdots + c_nx_n$$
$$\times \begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \leq (=, \geq) b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \leq (=, \geq) b_2 \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \leq (=, \geq) b_m \\ x_1x_2, \ldots, x_n = 0, 1; \;\; a_{ij} = -1, 0, 1; \\ b_j \text{ are non-negative integer} \\ (i = 1, 2, \ldots, m; \; j = 1, 2, \ldots, n) \end{cases} \tag{1}$$

We designed the following algorithm to solve the 0–1 programming problem corresponding to (1):

Step 1. Generate all possible combinations of variable 0 or 1 in the given problem.

Step 2. Reject infeasible solutions according to constraint inequalities (reserved feasible solution).

Step 3. Generate remaining solutions.

Step 4. Repeat Steps 2 and 3. We can remove all infeasible solutions and obtain feasible solutions of the problem; then proceed Step 5.

Step 5. By comparing to value of object function corresponding to every feasible solution, we can obtain an optimum solution.

We also designed the following biology operation of the above algorithm:

Step 1a. Combine a set of single-stranded DNA molecules that represent all variables in the computational problem at hand. Synthesize and place samples in an addressed fashion on a surface, and arrange these single-stranded DNA molecules according to the form of dot matrix. Certain DNA oligonucleotides are tagged with two different fluorescent colors as DNA probes.

Step 2a. For each inequality, by adding the corresponding complementary strand to the surface, any solution that satisfies this inequality will be hybridized by a complementary strand that is tagged with a fluorescent label, with a differential value (*D*-value) of two different colors that is at least (not exceeding) b_i. Further, we can determine the solution for satisfying (dissatisfying) constraint conditions by a method of fluorescence imaging.

Step 3a. The temperature is raised to separate all double-stranded DNA into single-strands by thermal denaturizing. The surface is returned to the initial state by washing in a buffer that is made of 10 μM Tris–HCl, 5 μM KCl, 5 μM $MgCl_2$, 10 μM SDS, and 50 μM H_2O (without regard for infeasible solution determined in Step 2a).

Step 4a. Repeating Steps 2a and 3a, we can reject all infeasible solutions and obtain a feasible solution of the problem: then proceed Step 5a.

Step 5a. By calculating and comparing the value of the object function corresponding to every feasible solution, an optimum solution can be obtained.

3. The model system

For a system of equations that contains *n* variables $x_1, x_2, \ldots, x_n$, and *m* equations, in order to implement Step 1a, two steps are needed. One of them is to synthesize 4*n* oligonucleotides divided into four groups, which include *n* oligonucleotides in each group. The oligonucleotides in the first group represent variables $x_1, x_2, \ldots, x_n$, respectively; the second represents variables $\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n$, respectively ($x_i = 1$ if and only if $\bar{x}_i = 0$); the third group represents complementary strands of the first group respectively, write individually as $x'_1, x'_2, \ldots, x'_n$; the fourth group represents complementary strands of the second group, respectively, write individually as $\bar{x}'_1, \bar{x}'_2, \ldots, \bar{x}'_n$. We elect oligonucleotides $x_1, x_2, \ldots, x_n$ and $\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n$ such that they are very different, to evade mispairing among them, at least four base pairs (bp) must differ in any two oligonucleotide sequences (note that oligonucleotide x_i represents variable $x_i = 1$ and oligonucleotide $\bar{x}_i$ represents variable $\bar{x}_i = 0$). The other is to structure a DNA probe with the former two-group 2*n* oligonucleotides.

The process of constructing DNA probes can be separated into five steps:

(1) Construct $2n$ probes corresponding to above single-stranded DNA molecules (oligonucleotide) and tag oligonucleotides $x'_1, x'_2, \ldots, x'_n$ and $\bar{x}'_1, \bar{x}'_2, \ldots, \bar{x}'_n$ with two different color fluorescents (fluorescent blue Cy3$^{\text{TM}}$, fluorescent green ABI).

(2) Fix untagged DNA strands to the surface by means of a connection of six to nine atoms where the DNA strands are arranged in 2^n rows representing all variables of the given computational problem. The surface bound oligonucleotide sequences utilized were of the form 5′-HS-C_6-$T_{15}, \ldots$-3′. The T_{15} sequence serves as a spacer group to separate the hybridizing sequence from the support (Guo and Guilfoyle, 1994).

(3) To implement Step 2, add the complementary strand corresponding to each variable of the constraint equation to the surface. Any solution that satisfies this inequality will be hybridized by a complementary strand tagged with a fluorescent label, with a D-value of two different colors that is at least (not exceeding) b_i. Further, we can determine the solution for satisfying (dissatisfying) the constraint equation by a method of fluorescence image, and observe their color and record.

(4) To implement Step 3, the temperature is raised to separate all double-stranded DNA into single-strands by thermal denaturation. After washing with a certain buffer (Tris–HCl, KCl, $MgCl_2$, H_2O), the surface is returned to the initial state (without regard for infeasible solution determined before).

(5) To implement Step 4, we can remove all infeasible solutions and obtain a feasible solution of the problem by repeating (2) and (3). To implement Step 5, comparing the value of object function to corresponding every feasible solution, we can obtain optimum solution. We discuss in detail the simple 0–1 programming problem as below:

$$\min(u) = 2x + 3y + 2z \qquad \begin{cases} x + y - z \geq 1 \\ x + z \leq 1 \\ y + z \leq 1 \\ x, y, z = 0, 1 \end{cases}$$

To discuss the 0–1 programming problem, the process was separated into six steps:

(1) We first synthesized 12 oligonucleotides divided into the same four groups. Three oligonucleotides of the first group represented variables x, y, z, respectively; the oligonucleotides in the second group represented variable $\bar{x}, \bar{y}, \bar{z}$, respectively ($x = 1$ if and only if $\bar{x} = 0$, such as y, z); the oligonucleotides in the third group represented the complementary strand of the first group respectively, write individually as x', y', z'; the oligonucleotides in the fourth group represented the complementary strand of the second oligonucleotides group respectively, write individually as $\bar{x}', \bar{y}', \bar{z}'$ (see Fig. 2). In order to evade mispairing among oligonucleotides, we choose oligonucleotides x, y, z and $\bar{x}, \bar{y}, \bar{z}$ such that they must be very different (in which at least four base pairs differ), oligonucleotide x represents variable $x = 1$ and oligonucleotide $\bar{x}$ represents variable $\bar{x} = 0$, such as y, z. Then, we structured DNA probes by respectively tagging the latter two-group six oligonucleotides with fluorescent blue and fluorescent green, fixation of untagged DNA strands to the surface by means of a connection of six to nine atoms where the DNA strands are arranged in three lines and eight rows representing all variables of the given computational problem (see Fig. 3).

(2) For the first constraint equation, we added the complementary strands x', y' tagged with fluorescent green and the z' tagged with fluorescent blue corresponding to variables x, y, z to the

x: 5’-AACCTGGT-3’	y: 5’-ACCATAGC-3’	z: 5’-AGAGTCTC-3’
$\bar{x}$: 5’-CCAAGTTG-3’	$\bar{y}$: 5’-GTTGGGTT-3’	$\bar{z}$: 5’-AGCTTGCA-3’
x': 3’-ACCAGGTT-5’	y':3’-GGTATGGT-5’	z':3’-GAGACTCT-5’
$\bar{x}'$: 3’-CAACTTGG-5’	$\bar{y}'$: 3’-AACCCAAC-5’	$\bar{z}'$: 3’-TGCAAGCT-5’

Fig. 2. Detailed encoding of all variables.

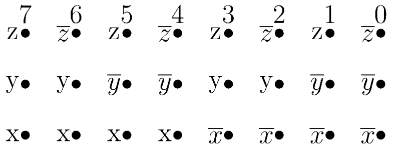

Fig. 3. Fixed untagged DNA strands on the surface.

surface. Any solution satisfying this inequality is hybridized with the complementary strand tagged with a fluorescent label (with a *D*-value of fluorescent green and fluorescent blue is at least one). Further, we can determine the solution of the satisfying constraint equation by a method of fluorescence imaging, and observe their color and record (the feasible solution of the problem is "2, 4, 6, 7," see Fig. 4).

(3) The temperature is raised to separate all double-stranded DNA into single-strands by thermal denaturation. After washing in a buffer under certain conditions (room temperature and 38 °C), the surface is returned to the initial state.

(4) For second constraint equation, proceed similar to Steps 2 and 3 above by adding the complementary strands *x*′, *z*′ tagged with fluorescent green corresponding to variables *x*, *z* to the surface, we can determine the solution of satisfying the constraint equation by a method of fluorescence image, and observe their color and record. Any solution satisfying this inequality is hybridized with at least one complementary strand tagged with a fluorescent green (the feasible solution of the problem is "0, 1, 2, 3, 4, 6," see Fig. 5), repeat Steps 2 and 3 above.

(5) For third constraint equation, proceed similar to Steps 2 and 3 above by adding the complementary strands *y*′, *z*′ with fluorescent green corresponding to variable *y*, *z* to the surface. Any solution that satisfies this inequality will be hybridized at least one complementary strand tagged with a fluorescent green (the feasible solution of the problem is "0, 1, 2, 4, 5, 6," see Fig. 6).

(6) By comparing to the value of the object function corresponding to every feasible solution, we can obtain optimum solution (because there is only a feasible solution "6" in the problem, it must be optimum solution). Variable value corresponding to "6" is (1, 1, 0), the minimum value of object function is 5.

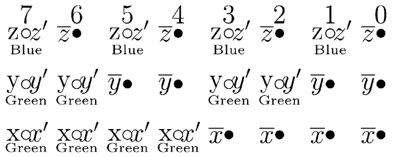
7 6 5 4 3 2 1 0
zoz′ z̄• zoz′ z̄• zoz′ z̄• zoz′ z̄•
Blue Blue Blue Blue
yoy′ yoy′ ȳ• ȳ• yoy′ yoy′ ȳ• ȳ•
Green Green Green Green
xox′ xox′ xox′ xox′ x̄• x̄• x̄• x̄•
Green Green Green Green

Fig. 4. Hybridize figure of the first constraint equation.

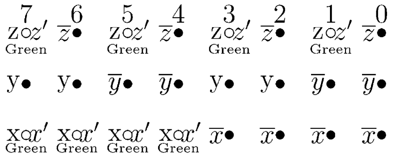

Fig. 5. Hybridize figure of the second constraint equation.

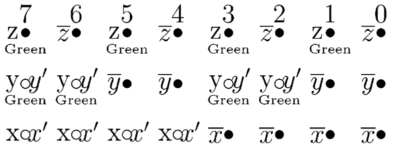
7 6 5 4 3 2 1 0
z• z̄• z• z̄• z• z̄• z• z̄•
Green Green Green Green
yoy′ yoy′ ȳ• ȳ• yoy′ yoy′ ȳ• ȳ•
Green Green Green Green
xox′ xox′ xox′ xox′ x̄• x̄• x̄• x̄•

Fig. 6. Hybridize figure of the third constraint equation.

The experiment is not complicated and we can accomplish a result that is similar to the experiment performed by Wu (2001).

4. Analysis and conclusions

Because computers have obvious limits in storage, speed, intelligence, and miniaturization, the methods of DNA computation have arisen, especially for their efficient parallelism. In order to solve a practical issue, there are still some problems that need further study in biologic technology. In this article, we highlight a DNA computing model to solve a problem of 0–1 programming problem. The model we proposed has a potential to solve the linear programming problem, which is an important issue in operations research. With the advances of the biological technology and the molecular biology, the general linear programming problem will be solved. In our method, we adopt fluorescence marking technique and laser focus technique, and determine the solution by analyzing fluorescence, the method of which has some significant advantages such as low cost, low error, short operating time, reusable surface and simple experimental steps.

Acknowledgements

The authors sincerely appreciate the encouraging comments of the Editor of the journal on this paper. They also wish to thank an anonymous referee of this paper who provided many useful and constructive suggestions for the improvement of this paper. Finally, thanks to all the authors that appear in the references.

References

Adleman, L.M., 1994. Molecular computation of solutions to combinatorial problems. Science 266, 1021–1023.

Bondy, J.A., Murty, U.S.R., 1976. Graph Theory with Applications. Macmillan, New York.

Braich, R.S., Chelyapov, N., Johnson, C., Rothemund, P.W.K., Adleman, L., 2002. Solution of a 20-variable 3-SAT problem on a DNA computer. Science 296, 499–502.

Christos, H., Papadimitriou, T., 1982. Combinatorial Optimization: Algorithms and Complexity. Prentice-Hall, Englewood Cliffs, NJ.

Cukras, A.R., Faulhammer, D., Lipton, R.J., Landwerber, L.F., 1999. Chess games: a model for RNA based computation. Biosystems 52, 35–45.

Feynman, R.P., 1961. In: Gilbart, D.H. (Ed.), Minaturization. Reinhold, New York, pp. 282–296.

Gass, S.L., 1984. Linear Programming Methods and Applications, 5th ed. McGraw-Hill, New York.

Guo, Z., Guilfoyle, R.A., 1994. Direct fluorescence analysis of genetic polymorphisms by hybridization with oligonecleotide arrays on glass supports. Nucl. Acid. Res. 22, 5456–5465.

Head, T., 1987. Formal language theory and DNA: an analysis of the generative capacity of specific recombinant behaviors. Bull. Math. Biol. 49, 737–759.

Lipton, R.J., 1995. DNA solution of hard computational problems. Science 268, 542–545.

Liu, Q.H., et al., 2000. DNA computing on surfaces. Nature 403, 175–179.

Ouyang, Q., Kaplan, P.D., Liu, S., 1997. DNA solution of the maximal clique problem. Science 278, 446–449.

Skamoto, K., 2000. Molecular computation by DNA hairpin formation. Science 288, 1223–1226.

Wu, H.Y., 2001. An improved surface-based method for DNA computation. Biosystems 59, 1–5.

Yoshida, H., 1999. DNA based computers 5. In: Winfree, E., Cifford, D.K. (Eds.), Vol. 54 of DIMACS Series in Discrete Mathematics and Theoretical Compute Science. American Mathematical Society, Providence, Rl, pp. 9–22.