A Genetic Algorithm to Solve the Subset Sum Problem based on Parallel Computing

Lei Li¹, Kai Zhao²,∗ and Zuwen Ji³

1 Department of civil engineering, Xi’an University of Architecture & Technology, Xi’an 710055, PR China
2 Academic Affair Office, Pingdingshan University, Pingdingshan 467000, PR China
3 State Key Laboratory of Simulation and Regulation of River Basin Water Cycle, China Institute of Water Resources and Hydropower Research, Beijing 100048, P. R. China

Received: 12 Jun. 2014, Revised: 10 Sep. 2014, Accepted: 12 Sep. 2014
Published online: 1 Mar. 2015

Abstract: The subset sum problem is to find subsets in a given number set, meanwhile number sum of the subset is equal to appointed value. It is a classical NP-complete problem in graph theory. It can be solved by the electronic computer in exponential time. In this paper, we consider a DNA procedure for solving the subset sum problem in the Adleman-Lipton model. The procedure works in \( O(n) \) steps for the subset sum problem of an undirected graph with \( n \) vertices. The innovation of the procedure is the ingenious choice of the vertices strands’ length, which can get the solution of the problem in proper length range and simultaneity simplify the complexity of the computation.

Keywords: genetic algorithm; The subset sum problem; parallel computing

1 Introduction

DNA computing is calculated by molecular biology method to solve complex mathematical problems, which is a combined product of biology, mathematics, computer and information science. According to certain rules in the original problem of data information onto the DNA molecular chain, DNA computing uses the double helix structure and DNA molecule complementary principle of information coding, get the solution space data pool of the problem. After a series of biochemical reaction controlled parallel manipulation of DNA molecules, it excludes non feasible solution chain. Finally, it get the ultimate DNA chain using molecular biological extraction technique, which is the solution of the problem. As the pathbreaking work of DNA computation, Leonard Adleman [1] succeeded in solving an instance of the Directed Hamiltonian Path Problem solely by manipulating DNA strings, and also illustrated the potential parallel ability of DNA computing in 1994. Lipton [2] testified that the same method could be used to solve another NP-complete problem (satisfiability problem) in 1995. Since then, DNA computing, as an interdisciplinary science using DNA molecular biotechnologies to solve conundrum problems of computer science and computational mathematics, has a wide application prospect in solving difficult problems. Huge storage capacity, massive parallelism and low energy consumption are primary advantages of DNA computing. The advantages imply that we can utilize DNA molecule to solve harder, larger problems such as NP-complete problems in linearly increasing time complexity, in contrast to the exponentially increasing time complexity required by electronic computers. Recently, DNA computing has attracted more and more attention and interest from research scholars. In recent years, lots of papers have occurred for designing DNA procedures and algorithms to solve various NP-complete problems [3,4,5,6,7,8,9,10]. However, most of the previous works in DNA computing do not require the consideration of the representation of numerical data in DNA strands. In fact, many practical applications in the real world involve edge-weighted or vertice-weighted graph problems such as shortest path problem, subset sum problem, etc. Therefore, representation of numerical data in DNA strands is an important issue toward expanding the capability of DNA computing to solve numerical optimization problems. There have been some previous works to represent the numerical data with DNA.

∗ Corresponding author e-mail: jerrys76@163.com
Narayanan et al. [11] presented a conceptual encoding method that represents costs with the lengths of DNA strands. Shin et al. [12] proposed a method for representing the real numbers in fixed-length DNA strands by varying the number of hydrogen bonds. Yamamura et al. [13] proposed a concentration control method which encoded the numerical data by means of the concentrations of DNA strands.

In this paper, a DNA procedure is introduced for figuring out solutions of the subset sum problem: for an vertex-weighted graph \( G = (V, E) \) find vertex subsets. The subsets contain some vertices of \( V \), which sum of vertices weight is equal to specified \( L \). For instance, the vertex-weighted graph \( G \) in Fig. 1 defines such a problem. We assume that the specified \( L \) is 23. It is not difficult to find that the vertex subset \( \{v_1, v_2, v_5, v_6, v_7\} \) with corresponding weight \( \{5, 2, 7, 3, 6\} \) is one solution to the subset sum problem for graph \( G \) in Fig. 1. We encode the numerical data by means of the lengths of DNA strands, the same way as that in [11]. A DNA procedure is formally presented by means of the DNA operations proposed by Adleman [1] and Lipton [2].

The rest of this paper is organized as follows. In Section 2, the Adleman-Lipton model is introduced in detail. Section 3 introduces a DNA algorithm for solving the subset sum problem and the complexity of the proposed algorithm is described. We give conclusions in Section 4.

2 The genetic algorithm model

DNA computing is calculated by molecular biology method to solve complex mathematical problems, which is a combined product of biology, mathematics, computer and information science. According to certain rules in the original problem of data information onto the DNA molecular chain, DNA computing uses the double helix structure and DNA molecule complementary principle of information coding, get the solution space data pool of the problem. After a series of biochemical reaction controlled parallel manipulation of DNA molecules, it excludes non feasible solution chain. Finally, it get the ultimate DNA chain using molecular biological extraction technique, which is the solution of the problem. As the pathbreaking work of DNA computation, Leonard Adleman [1] succeeded in solving an instance of the Directed Hamiltonian Path Problem solely by manipulating DNA strands, and also illustrated the potential parallel ability of DNA computing in 1994. Lipton [2] testified that the same method could be used to solve another NP-complete problem (satisfiability problem) in 1995. Since then, DNA computing, as an interdisciplinary science using DNA molecular biotechnologies to solve conundrum problems of computer science and computational mathematics, has a wide application prospect in solving difficult problems. Huge storage capacity, massive parallelism and low energy consumption are primary advantages of DNA computing. The advantages imply that we can utilize DNA molecule to solve harder, larger problems such as NP-complete problems in linearly increasing time complexity, in contrast to the exponentially increasing time complexity required by electronic computers. Recently, DNA computing has attracted more and more attention and interest from research scholars.

A DNA(deoxyribonucleic acid) is a polymer, which is stringed together from monomers called deoxyribonucleotides [15]. Distinct nucleotides are detected only with their bases. Those bases are, respectively, abbreviated as adenine (A), guanine (G), cytosine (C), and thymine (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, e.g., the single strands 5’CTGCAGTACACC3’ and 3’GACGTCATGTTGG5’ can form a double strand. We also call the strand 3’GACGTCATGTTGG5’ as the complementary strand of 5’CTGCAGTACACC3’ and simply denote 3’GACGTCATGTTGG5’ by CTGCAGTACACC. 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, it is called a 20mer. 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 as 20 bp.

The DNA operations proposed by Aldeman [1] and Lipton [2] are described below. These operations will be used for figuring out solutions of the traveling salesman problem in this paper. The genetic algorithm model: A (test) tube is a set of molecules of DNA (i.e., a multi-set of finite strings over the alphabet \( \{A,C,G,T\} \)). Given a tube, one can perform the following operations:

(1) Merge \((T_1,T_2)\): for two given test tubes \(T_1,T_2\), it stores the union \(T_1 \cup T_2\) in \(T_1\) and leaves \(T_2\) empty;
(2) Copy \((T_1, T_2)\): for a given test tube \(T_1\), it produces a test tube \(T_2\) with the same contents as \(T_1\).

(3) Detect \((T)\): given a test tube \(T\), it outputs "yes" if \(T\) contains at least one strand, otherwise, outputs "no".

(4) Separation \((T_1, X, T_2)\): for a given test tube \(T_1\) and a given set of strings \(X\), it removes all single strands containing a string in \(X\) from \(T_1\), and produces a test tube \(T_2\) with the removed strands;

(5) Selection \((T_1, L, T_2)\): for a given test tube \(T_1\) and a given integer \(L\), it removes all strands with length \(L\) from \(T_1\), and produces a test tube \(T_2\) with the removed strands;

(6) Cleavage \((T, \gamma_0, \gamma_1)\): for a given test tube \(T\) and a string of two (specified) symbols \(\gamma_0, \gamma_1\), it cuts each double strand containing \(\begin{vmatrix} \gamma_0 & \gamma_1 \\ \beta_0 & \beta_1 \end{vmatrix}\) in \(T\) into two double strands as follows:

\[
\begin{bmatrix}
\alpha_0 \gamma_0 \gamma_1 \\
\alpha_1 \gamma_0 \gamma_1
\end{bmatrix} \rightarrow 
\begin{bmatrix}
\alpha_0 \gamma_0 \\
\alpha_1 \gamma_0
\end{bmatrix} \cdot 
\begin{bmatrix}
\gamma_1 \beta_0 \\
\gamma_1 \beta_1
\end{bmatrix}.
\]

(7) Annealing \((T)\): for a given test tube \(T\), it produces all feasible double strands in \(T\). The produced double strands are still stored in \(T\) after annealing;

(8) Denaturation \((T)\): for a given test tube \(T\), it dissociates each double strand in \(T\) into two single strands;

(9) Discard \((T)\): for a given test tube \(T\), it discards the tube \(T\);

(10) Read \((T)\): for a given test tube \(T\), the operation is used to describe a single molecule, which is contained in the tube \(T\). Even if \(T\) contains many different molecules each encoding a different set of bases, the operation can give an explicit description of exactly one of them.

(11) Append \((T, Z)\): for a given test tube \(T\) and a given short DNA singled strand \(Z\) it appends \(Z\) onto the end of every strand in the tube \(T\).

Since these ten manipulations are implemented with a constant number of biological steps for DNA strands [15], we assume that the complexity of each manipulation is \(O(1)\) steps.

3 DNA algorithm for the Subset Sum Problem

Let \(G = (V, E)\) be a vertex-weighted graph with the set of vertices \(V = \{v_i \mid k = 1, 2, \ldots, n\}\) and the set of edges \(E = \{e_{i,j} \mid 1 \leq i, j \leq n, i \neq j\}\). Note that the weight of vertex \(v_k\) is \(k_i\). The vertices \(v_i\) and \(v_j\) can be divided up by the edges \(e_{i,j}\) or \(e_{j,i}\) in \(E\). Without loss of generality, we assume that specified length is \(L\).

In the following, we use the distinct DNA singled strands symbols \(\#A_1\#, \#A_2\#, \ldots, \#A_n\#\) to denote the vertex \(v_k\) for which \(|\#| = |\#|\), where \(|\#|\) denotes the length of the DNA singled strand. Simultaneously the symbol \(\#\) is the signal of division between different vertices. Suppose that all weights of vertices in the given graph are commensurable. The DNA singled strands \(A_i\) and \(\overline{A_i}\) are both used to denote the weights \(k_i\) on the vertices \(v_i \in V\) with \(|A_i| = |\overline{A_i}| = k_i\), e.g., take \(|A_1| = |\overline{A_1}| = 5\)mer in Fig. 1, Then let \(m = \max_{v \in V} k_i\) and \(|\#| = |\#| = n \times m = t\). For example, for the graph in Fig. 1, we can let \(m = \max\{5, 2, 9, 10, 7, 3, 6\} = 10\)mer, Then \(|\#| = |\#| = n \times m = t = 7 \times 10 = 70\)mer. Let

\[
P = \{\#A_1\#, \#A_2\#, \#A_3\#, \ldots, \#A_n\# \mid v_i \in V, i = 1, 2, \ldots, n\}
\]

\[
Q = \{x\}
\]

We design the following algorithm to solve the subset sum problem and give the corresponding DNA operations as follows:

(1) We choose all possible subsets of vertices.

\((1-1)\) Annealing \((P)\); \((1-2)\) Denaturation \((P)\); \((1-3)\) Cleavage \((P)\);

After the above three steps of manipulations, the singled strands in tube \(P\) will encode all sets of vertices. For example, for the graph in Fig. 1, we have singled strands: \(\#A_1\#A_2\#A_3\#A_4\#A_5\#A_6\#\) which correspond to set of vertices \(\{v_1, v_2, v_3, v_4, v_5, v_6\}\) with the weight sum 32 respectively. This operation can be finished in \(O(1)\) steps since each manipulation above works in \(O(1)\) steps.

(2) Each single strand in tube \(P\) denotes one possible vertex set. Even because the effort of biotechnologies, we meantime get some strands which denote beyond subsets of vertices. For example, for the graph in Fig. 1, we have single stranded:

\(\#A_1\#A_2\#A_3\#A_4\#A_5\#A_6\#\) which correspond to set of vertices \(\{v_1, v_2, v_3, v_4, v_5, v_6\}\). Of course, it can not be the optimum solution because the set contains the vertex \(v_2\) twice. To find proper length subset sum, we append the vertex length information on each strand. So we can get all possible strands which contain whole vertices length information at least once.

For \(k = 1 \rightarrow k = n\)

\((2-1)\) Separation \((P, \{A_1\}, T_1)\); \((2-2)\) Separation \((P, \{A_2\}, T_2)\); \((2-3)\) Append \((P, x)\); \((2-4)\) Merge \((P, T_1)\); \((2-5)\) Merge \((P, T_2)\); \((2-6)\) Discard \((T_1)\); \((2-7)\) Discard \((T_2)\); End for

In the above operations, we get the strands that contain all vertices length information at least once.

For example, for the graph in Fig. 1, we have single stranded:

\(\#A_1\#A_2\#A_3\#A_4\#A_5\#A_6\#\) which denote the set \(\{v_1, v_2, v_3, v_4, v_5\}\). We append \(x\) twice on the strands for that original strands omit vertices \(v_6, v_7\) length information. In the above operation we use a "For" clause. Thus this operation can be finished in \(O(n)\)
steps since each single manipulation above works in $O(1)$ steps.

(3) First of all, We set the length of DNA strands as following:

\[ m = \max_{v_i \in V} k_i \] and \[ |\#| = |\overrightarrow{\#}| = |x| = n \cdot m = t. \]

If the vertex $v_j \in E_1$, we append $x$ with length $t$ on the correspond strands. Consequently the length of DNA strands which denote containing every vertex length information only once in tube $P$ must be between $(n + 1)t$ and $(n + 2)t$. We can get the strands denoting the optimum solution in this length range. This is done by the following manipulations.

(3-1) Selection $\{P, (n + 1)t + L, T\}$;
(3-2) Detect $(T)$

If Detect $(T)$ is ”yes”, then the solution of subset sum problem is obtained. This operation can be finished in $O(1)$ steps since each manipulation above works in $O(1)$ steps.

(4) Finally the Read operation is applied to giving the exact vertices subset in the subset sum problem. For example, for the graph in Fig. 1, the vertices subsets are \( \{v_1, v_2, v_3, v_6, v_7\}, \{v_1, v_2, v_3, v_4, v_7\}, \{v_1, v_2, v_3, v_5\} \) and \( \{v_4, v_5, v_7\} \) with total weights sum 23.

(4-1) Read $(T)$;

The following theorem tells that the algorithm proposed above really can get solutions of the subset sum problem in $O(n)$ steps using DNA molecules.

**Theorem 1.** The solutions of subset sum problems for a graph with $n$ vertices can be figured out in $O(n)$ steps using DNA molecules.

**Proof.** After the operations of first step, all the singled strands in tube $P$ denote all sets of vertices. Then strands can be described:

\[ \#A_{i_1} \overrightarrow{#A_{i_2}} \cdots \#A_{i_j} \overrightarrow{#A_{i_j}} \# \]

To eliminate the variance of vertices number in sets, We append $x$ with length $t$ one time on the strands if we detect the strands missing the vertex $v_j$ length information once. So After the operations of second step, all the strands in $P$ contain all the vertices length information at least once. we reasonably design the length of $\#A, A_i, \overrightarrow{A_i}$ and $x$. For $|A_i| = |\overrightarrow{A_i}| = k_i$ (where $k_i$ is the weights of vertex $v_i \in E$)

\[ m = \max \{k_i\} = \max \{|A_i|, |\overrightarrow{A_i}|\} \]

\[ |\#| = |\overrightarrow{\#}| = |x| = n \cdot m = t \]

So we define $S$ as the strands after the second step. Then $S$ can be described:

\[ \#A_{i_1} \overrightarrow{#A_{i_2}} \cdots \#A_{i_{|S|}} \overrightarrow{#A_{i_{|S|}}} #x \cdots x \]

The number of appending $x$ can be decided by the missing vertices information on the strands. Due to the possible of containing vertex $v_k$ information more than once, then $i_j + \lambda \geq n$. So

\[ |S| = |#| + |A_{i_1}| + |\overrightarrow{A_{i_2}}| + \cdots + |\overrightarrow{A_{i_{|S|}}}| + |x| + \cdots + |x| \]

\[ = (i_j + 1)|#| + \sum_{v_i \in V} |A_{i_1}| + \lambda |x| \]

\[ = (i_j + \lambda + 1)t + \sum_{v_i \in V} |A_{i_1}| \]

\[ \geq (n + 1)t + \sum_{v_i \in V} |A_{i_1}| \]

\[ : 0 \leq \sum_{v_i \in V} |A_{i_1}| \leq t \]

So the length of strands which denote containing all the vertices information only once must be between $(n + 1)t$ and $(n + 2)t$. The length of strands which denote containing a same vertex information more than once must be longer than $(n + 2)t$. So we can get the solution in step (3) in appropriate length range.

Besides, the manipulates of algorithm can be entirely finished in finite operations. Such as step (1), (3), (4) in $O(1)$, Simultaneity step (2) in $O(n)$. In conclusion, We can get the solution of subset sum problem with $n$ vertices in $O(n)$.

**4 Conclusion**

In this paper, we propose a procedure for the subset sum NP-complete problem in the genetic algorithm model. The procedure works in $O(n)$ steps for the subset sum problem of an vertex-weighted graph with $n$ vertices. All our results in this paper are based on a theoretical model. However, to solve some complex problems using the new method can help us understand more about the characteristic of problem and promote the development of DNA computing research, for that DNA-based computers may be a good choice for performing massively parallel computations. Up to now, there are still many unsolved mathematical NP-complete problems because they are difficult to support basic biological experiment operations. We hope that our results can have certain help and effect to the DNA-based computing development.

**Acknowledgement**

The project was supported by CNSF (Grant No. 51108376) and SSF (Grant No. 2014JQ7231). The corresponding author thanks the fund support by the Science and Technology Project of Henan Province (Grant No. 142102210225). This research was also supported by the Open Research Fund of State Key Laboratory of Simulation and Regulation of Water Cycle in River BasinChina Institute of Water Resources and Hydropower Research (Grant No. 2014ZY05) and 12th
Five-Year Plan to Support Science and Technology Project (Grant No. 2012BAB04B02) respectively. The authors are grateful to the anonymous referee for a careful checking of the details and for helpful comments that improved this paper.

References


Lei Li received the PhD degree in structural engineering from the Xi’an University of architecture and technology China in 2010. Currently he is a lecturer in civil engineering department of Xi’an University of architecture and technology, China. His research interests include computational mathematics and SVM.

Kai Zhao received a master’s degree in Computer Engineering from Xidian University China in 2010. Currently he is a lecturer in Pingdingshan University of computer science and technology, China. His research interests include computational mathematics and SVM.

Zuwen Ji was born in Jiangsu, China in 1967. He received the BS degree from Wuhan University of Hydraulic and Electrical Engineerin. His research interests include parallel computing and large scale data processing.