Applying DNAC in Solving the Subset Sum Problem

Maryam S. Nuser *

Received on Jan. 13, 2010                        Accepted for publication on Nov. 4, 2010

Abstract

DNA computing (DNAC), from computer science point of view is considered as a wide biological and computational concept that focuses on the idea of using DNA for non-biological applications. Several applications especially NP-complete problems were solved using DNAC. One of these hard problems, that is related to the mathematics discipline, is the subset sum problem. This paper proposes a method for solving the subset sum problem based on DNAC. The method uses variable length representation of elements in the set based on their values, and extracts the sequence with length that represents the required sum. This research provides more evidence on the ability of DNAC to solve computational problems.

Keywords: DNA Computing, DNAC, Subset Sum Problem, NP Complete.

Introduction

DNA stands for deoxyribonucleic acid; it is made up of four bases known as Adenine (A), Guanine (G), Cytosine (C), and Thymine (T). There are approximately three billion bases found in human DNA, the order of these four bases, which distinguishes one DNA sequence from another, is responsible for building and supporting an individual [1].

DNA exists, in living organisms, as a pair of molecules rather than a single molecule. These strands are twisted in the shape of a double helix. The helix is kept stable by hydrogen bonds, which can be found between the bases attached to the two strands [1].

DNA bases are specific in that an Adenine base only pairs with a Thymine base, and a Cytosine base bonds to a Guanine base. This base pairing is also known as complementary base pairing. The concept is quite simple but it is significant for DNA [1].

Using DNAC, DNA sequences are used to represent problem instances, and then several DNA operations are applied to the DNA sequences in a specific order in order to find a solution. The operations used usually include ligation, Polymerase Chain Reaction (PCR), agarose gel electrophoresis, and fluorescence labeling.

DNA ligation, which is illustrated in figure 1, is the process of joining two DNA molecule ends which is, in computer science, similar to the concatenation of two strings to form a single string. PCR, or polymerase chain reaction, is a method by which DNA sequences can be duplicated into millions in a few hours. It is usually used to amplify the
number of specific sequences in the test tube. Agarose gel electrophoresis, as shown in figure 2, is a method that is used to separate DNA or RNA molecules by size. Fluorescent labeling is the process of covalently attaching a fluorophore to another molecule. Fluorescent labels are generally used for detection of a labeled molecule via some fluorescence-reading instrument. This can be useful in localization of a target within a cell.

Figure (1): DNA Ligation

The paper is organized as follows. Section 2 introduces a literature review about solving problems using DNA and especially the subset sum problem (SSP). Section 3 shows an algorithm to solve the SSP using computers and then using DNA. Analysis of the algorithm with implementation steps are discussed in section 4. Results and discussion are illustrated in section 5 followed by a conclusion in section 6.
Applying DNAC in Solving the Subset Sum Problem

![Figure (2): Agarose Gel Electrophoresis](image)

Literature Review

Adleman used DNA to solve an instance of a well known graph problem, the Hamiltonian Path Problem (HPP) [2]. Following Adleman’s success, solutions for other NP-complete problems in different fields using DNA were published [3-8]. The use of DNA to solve computational problems is mainly because of its massive parallelism. Several fields including computer science, engineering, and mathematics benefited from DNAC in solving a field related problem, especially hard problems, or building specific applications.

In spite of the ability of DNA to solve different kinds of problems, it showed its suitability to solve NP complete problems more than conventional problems. One of these NP complete problems is the Subset Sum Problem (SSP). The SSP can be defined as [9]:

Given a set of integers $S$ and a target integer $k$, we ask whether there is a subset of $S$ whose elements sum to $k$?
For example, given the set \( S = \{10, 3, 2, 1, 8\} \). The question: is there any subset of \( S \) that sum to 5? The answer is yes because the subset \( \{3, 2\} \) sums to 5.

Several algorithms in the literature were suggested to solve the SSP. In [10] five techniques were proposed for solving the problem. One technique that is based on enumeration methods aims at solving small problems. Two techniques are based on an efficient number partitioning algorithm. These techniques can solve small problems very efficiently when the solution uses approximately half the available elements. The other two techniques use a direct approach of improving a solution, and they were shown to perform very well on large problems.

A technique that is based on the idea of divide and conquer and which uses recursion is proposed in [11]. The algorithm divides the SSP instance into subinstances where each subinstance is divided into smaller subinstances and so on until the new subinstances are efficiently solvable. The solution of the original instance is formed by collecting the solutions of the smaller instances.

Another technique that was implemented to halt in a polynomial time is proposed in [12]. The running time was against finding the solution; this is because the algorithm stops sometimes before finding the solution in order to obtain the polynomial time. The algorithm converts the problem to one of finding a particular short vector in a lattice, and then uses a lattice basis reduction algorithm to attempt to find the vector.

Since there is no deterministic polynomial time algorithm to solve the subset sum problem [9], an alternative is to use DNAC to do so. Several researches took this step and suggested different algorithms to solve the SSP.

Chang et al. proposed an algorithm for solving the SSP based on the Adleman–Lipton model and the solution space of stickers in the sticker-based model. They build, using DNA, an \( n \)-bit parallel adder and an \( n \)-bit parallel comparator to find the solution with a lower rate of hybridization errors than the Adleman’s approach. The adder was designed to find the sum of a set of numbers. They apply the adder on all possible subsets of the original set. After that, the comparator was used to compare the results obtained from the adder with the required sum to determine if there is a solution to the specified SSP instance [13].

Mihai and Oana solved the SSP in a time that is proportional with the target sum. They used an optical computational device which uses light rays to solve the problem. The device has a graph-like representation and the light rays traverse it by following the routes given by the connections between nodes [14].

The nodes are connected with arcs in a way that allows the generation of all possible subsets of the given set. Each arc is assigned either a number from the given set or a predefined constant. When the light is passing through an arc, it is delayed by the amount of time indicated by the number placed in that arc. If there is a ray at the destination node whose total delay is equal to the target value of the SSP (plus some constants), then that indicates the existence of a solution to the SSP. Their solution requires an exponential amount of energy [14].
Another algorithm incorporates the method of fluorescence labeling and the technique of gel electrophoresis to the Adleman-Lipton model and the solution space of stickers in the sticker-based model to solve the SSP. Their technique for solving the SSP used the strategy of divide and conquer with a new designed algorithm of Parallel Searcher. The proposed algorithm can solve the $n$-dimension subset sum instances by using $O(1.414^n)$ shorter DNA strands than the best molecular algorithm which uses $O(2^n)$ DNA strands. The algorithm is more scalable than other previous algorithms [15].

Another solution for two related and NP complete problems, namely subset-sum and knapsack, is suggested in [16]. The solution is based on the sticker based model. The algorithm used finite set sorting subroutine together with the descriptor procedure to formally verify the designed programs through the lab using inductive techniques.

In [17], a different technique that is based on quantum physics is proposed. Quantum theorems are implemented in an algorithm that solves an instance of the SSP. A Nuclear magnetic resonance (NMR) experiment for the simplest SSP to test the proposed theory is also performed and showed the ability of the algorithm to find the correct solution.

A linear solution of the SSP by using membrane creation is proposed in [18]. The researchers based their solution in creating all possible solutions which will produce an exponential amount of solutions. After that, all candidates are checked simultaneously to find the solution, if exists.

Methodology

The next section shows the methodology used in solving the SSP using silicon based computers followed by a section that shows the implementation of the same technique using DNA based computers.

Subset Sum Problem in silico

To solve the SSP using silicon based computers, one needs to find all possible subsets of the specified set and computes the sum of each subset to see if at least one of these subsets has a sum that equals the target value. The following algorithm illustrates a technique that can find if there exists a subset of the set $S$ with a sum that is equal to $k$.

1. Find all possible subsets of the set $S$.

Assume $S = \{10, 3, 2, 1, 8\}$. If the number of elements of the set $S$ is $n$, then the number of all possible subsets of $S$ is $2^n$. Therefore, the set of all possible subsets of $S$ is

$$\{\{\}, \{10\}, \{3\}, \{2\}, \{1\}, \{8\}, \{10, 3\}, \{10, 2\}, \{10, 1\}, \{10, 8\}, \{3, 2\}, \{3, 1\}, \{3, 8\}, \{2, 1\}, \{2, 8\}, \{1, 8\}, \{10, 3, 2\}, \{10, 3, 1\}, \{10, 3, 8\}, \{10, 2, 1\}, \{10, 2, 8\}, \{10, 1, 8\}, \{3, 2, 1\}, \{3, 2, 8\}, \{3, 1, 8\}, \{2, 1, 8\}, \{10, 3, 2, 1\}, \{10, 3, 2, 8\}, \{10, 3, 1, 8\}, \{10, 2, 1, 8\}, \{3, 2, 1, 8\}, \{10, 3, 2, 1, 8\}\}$$

2. Compute the sum of the elements of each subset. For the previous example, this will produce respectively the values $\{0, 10, 3, 2, 1, 8, 13, 12, 11, 18, 5, 4, 11, 3, 10, 9, 15, 14, 21, 13, 20, 19, 6, 13, 12, 11, 16, 23, 22, 21, 14, 24\}$.
3. If one of the values resulting from step 2 is equal to $k$, then the corresponding subset is the solution; otherwise there is no subset with the sum $k$. Assume $k = 5$ then the answer is yes and the subset is $\{3, 2\}$. On the other hand, assume $k = 7$, then the answer is no.

The previous algorithm has an exponential time complexity and therefore when the size of the set (i.e., the number of its element) increases, the time needed to solve the problem which is mainly consumed by step 1 that takes $O(2^n)$ (where $n$ is the number of elements in the set) also increases exponentially.

**Subset Sum Problem in Vitro**

When using DNAC to solve the SSP, we will build our solution based on the *in silico* algorithm illustrated above while encoding the problem instances with DNA sequences. The following steps present the DNAC technique that can be used to find if there exists a subset of sum $k$ from the set $S$:

1. Represent each element of $S$ with a DNA sequence of a specific length that is proportional to its value (as for example GCAGTAGGTC represents the number 10, CGC represents the number 3, GG represents 2, A represents 1, and AAAGGTGT represents 8). For simplicity, we will encode each element with a sequence of length that is exactly equal to its value, while in reality sequences of very short lengths such as 1 are not suitable. Therefore the encoded sequences should be proportional to the elements values. Note that the selection of the DNA sequences is not random, it should be such that the sequences form good encodings of the problem instances. This means the selected sequences should have the right percentage of each base as suggested by several researchers. In addition, the sequences should not cross hybridize. Several programs and suggestions are written by researchers to generate these sequences [20].

2. Add a sufficient number of copies of the DNA sequences that represent all the elements of the set $S$ to a test tube. Add a ligase and allow the DNA sequences to ligate. With the existence of multiple copies of each DNA sequence, it is likely that almost all possible subsets will be formed; if not then the solution may not be formed. Therefore, a sufficient number of copies of each used DNA sequence should be added. So, for the previous example some of the formed sequences will be:
   
   GCAGTAGGTC represents $\{10\}$, CGCGG represents $\{3, 2\}$, GCAGTAGGTCCGC represents $\{10, 3\}$ and CGCCGCAGTAGGTC represents $\{3, 3, 3, 10\}$ which should not be part of the solution as we will see in the next steps.

3. Extract DNA sequences that are of length equals $k$. This can be implemented by agarose gel. The resulting subsets are of sum $k$ and therefore they are strong candidates to be the solution except if they have redundant elements. As a result, redundancy should be checked and one way is to use fluorescence labeling. For the previous example, if $k = 5$, then the agarose gel operation will extract sequences that looks like:
Applying DNAC in Solving the Subset Sum Problem

CGCGG represents \{3,2\}, GGCGC represents \{2,3\}, GGGGA represents \{2,2,1\}, GGAGG represents \{2,1,2\}, and AAAAA represents \{1,1,1,1,1\}.

4. Label the complement of the DNA sequences that represent the elements of the set \(S\) with a specific fluorescent dye. The aim of this labeling is to be able to detect DNA sequences by color, and therefore it will be easier to detect if a DNA sequence contains a redundant subsequence. Add the labeled complements to the test tube and allow hybridization so that the single strands will bond with their complements, if exist, and form double stranded sequences.

5. Use the appropriate fluorescence reading instrument to detect if there is any double stranded DNA (dsDNA) sequence with no more than one label of each element. As for example, the dsDNA sequence CGCGG/GCGCC will have one label of each type and therefore forms a solution, while AAAAA/TTTTT will have 5 positions of the same label and therefore doesn’t form a solution.

6. If a sequence is found then a solution to the SSP exists and that sequence is the corresponding subset. Otherwise, no solution exists.

Note: remember to use PCR to amplify the resulting sequences after each step. This is necessary to increase the concentration of sequences and reduce errors. The \textit{in vitro} algorithm main steps are shown in Figure 3.
Analysis and Implementation

It is clear that the in silico algorithm has a time complexity of $O(2^n)$ where $n$ is the number of elements in the set $S$ whereas, the in vitro nondeterministic algorithm has a polynomial time. Adleman’s solution to the Hamiltonian path took 7 days, and less than that is expected here because the time consuming step that Adleman used is not used here.

According to the space complexity, the in silico algorithm needs a space of $O(2^n)$, where $n$ is the number of elements in the set $S$, to store all possible combinations of subsets of $S$. The in vitro algorithm needs also a space to store multiple copies of all subsets, but storing information in molecules of DNA allows for an information density
Applying DNAC in Solving the Subset Sum Problem

of approximately 1 bit per cubic nm, a dramatic improvement over existing storage media such as video tape which store information at a density of approximately 1 bit per $10^{12}$ cubic nanometers [2].

The implementation of such an algorithm can be done in the lab following the same approach of Adleman’s experiment where 50 pmol of each DNA sequence (i.e., more than $10^{13}$ copy) can be mixed together at a suitable temperature to allow ligation. After that, the product is PCR amplified and run on a gel. Using $n$ labels, the Watson-crick complement of each DNA sequence can be labeled with a different color. The labeled sequences can be added to the tube that contains the single stranded DNA sequences. Having the appropriate temperature, double strands can be formed, and finally checked with the appropriate fluorescence reading instrument.

Results and Discussion

The algorithm suggested here uses a variable length representation of problem instances. It uses ligation, gel electrophoresis, and fluorescence labeling to check the existence of a solution to the SSP. The algorithm has the ability to solve the SSP by checking the length of DNA strands rather than performing summation. The solution for the problem, which is a yes/no answer, can be found without the need for sequencing. In addition, the algorithm uses the massively parallel processing capabilities of DNA computers to speed up the time required for solving the problem.

Despite the previous advantages of the algorithm, it has a number of limitations. Solving large instances of the problem requires large amount of memory. In addition, sets with large elements require large DNA sequences. This limits the ability of the algorithm to solve instances of the problem with large elements in the set. This might be solved by using small DNA sequences to represent large numbers provided that the length of the small DNA sequence used is not one of the members of the set $S$.

Conclusion

DNAC was used to solve one of the famous NP complete problems which is the SSP. The algorithm is based on representing the elements of the set with variable length DNA sequences such that no two elements have the same length. Therefore, instead of using an adder to sum the elements, the ligation of sequences and then using gel electrophoresis indicates the length of sequences which represents the sum of the associated elements.

The new algorithm finds if there is a solution for the SSP with a polynomial number of steps taking advantage of the parallelism ability of DNA. The technique used can overcome the limitation of using very large DNA sequences. This can be done by representing large numbers with small DNA sequences provided that no two different sequences have the same length, and therefore can be considered as more scalable than several previous algorithms. As a future work, solutions for other NP complete problems using DNA might be presented.
استخدام الحوسبة بالحمض النووي لحل مسألة مجموع عنصر المجموعة الجزئية

مريم ساري نصير

ملخص

تعتبر الحوسبة البيولوجية، من وجهة نظر علوم الحاسب الآلي، فهوم مستقبلي. واسع، يركز على فكرة استخدام الحمض النووي للتطبيقات غير البيولوجية. هناك تطبيقات عدة ولا سيما المشاكل الم تعددة التي تتطلب وقتًا طويلاً لحلها حيث يمكن حلها باستخدام الحوسبة البيولوجية. واحدة من هذه المشاكل هي مسألة مجموع عنصر المجموعة الجزئية. تُقترح هذه الأسلوب.words ًا من أجل حل هذه المشكلة باستخدام الحوسبة البيولوجية. ويستخدم هذا الأسلوب مسلسلات حيوية بأطوال متغيرة لتمثيل الأرقام في المجموعة لاستخلاص المسألة بالطول المطلوب. هذه الطريقة تقدم مزيدًا من الابتكارات على قدرة الحوسبة البيولوجية في حل المشاكل الحسابية الصعبة.

References

Applying DNAC in Solving the Subset Sum Problem


