Using DNA Computing to Solve the Scheduling Problem

Maryam S. Nuser *

Received on Oct. 12, 2009 Accepted for publication on Dec. 31, 2009

Abstract

This research suggests a method for solving the job scheduling problem, which is an NP-hard problem, based on DNA computing (DNAC). The method uses variable length representation of jobs. Each job is represented with a DNA sequence that is proportional to its duration time. The resulting DNA sequences are allowed to ligate and then the sequence with length that represents the minimum completion time is extracted. This proves the ability of DNA to solve computational problems and so it can be considered as a starting point to solve the general scheduling problem with dependencies.

Keywords: DNA Computing; DNAC; Scheduling Problem; Job Shop Problem; Sequence.

Introduction

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.

The idea behind using DNAC is its advantages over silicon based computers. These advantages include its massive parallelism, huge capacity, and energy efficiency [1]. DNA computing solves hard problems by emulating computer based algorithms taking advantage of the idea of parallel processing. It can be seen as more than $10^{18}$ processor working together on solving a problem. This is one of the powerful sides of DNAC.

In order to solve a problem using DNA you have first to represent the problem instances with DNA sequences. Each DNA sequence may be viewed as a chain of nucleotides, or bases. A DNA sequence of length $n$ consists of $n$ consecutive letters from the set \{adenine (A), guanine (G), cytosine (C) and thymine (T)\}. After the encoding process, DNA operations are applied to multiple copies of these DNA sequences in order to reach the correct answer. These operations usually include ligation, PCR, agarose gel electrophoresis. DNA ligation is the process of joining together two DNA molecule ends which is, in computer science, similar to 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 is a method that is used to separate DNA or RNA molecules by size.

**Literature Review**

Adleman [1] solved an instance of the Hamiltonian Path Problem (HPP) using DNA. The HPP is the problem of finding a path from the start node to the end node going through all nodes in a graph only once. This problem is NP-complete and intractable for conventional computers to solve. Adleman represented each node in the graph with a unique DNA sequence of length 20 (20-mer). Each edge between two nodes, say v1 and v2, was represented by a DNA sequence that consisted of the last ten nucleotides of the sequence representing v1 plus the first ten nucleotides of v2.

Many copies of these edges (10^{18}) were allowed to hybridize, which is the binding of two DNA strands by weak forces to form double stranded DNA (dsDNA) sequences. These dsDNAs were ligated, which is the process of covalently bonding two ends of DNA together. Because multiple copies of each edge existed in the test tube, the hybridization and ligation processes generated a large number of random paths through the graph. Using biological operations, Adleman only kept the paths that form the correct solution which are those that start at the start node, end at the end node, contain all nodes of the graph only once, and are of a specific length. A detailed explanation of Adleman’s solution with an explanation of the biological operations used in the solution can be found in [1]. This solution required \( O(n) \) biological steps to solve a graph of size \( n \), but an exponential amount of DNA.

Following Adleman’s success, solutions for other NP-complete problems using the same methodology were published [2, 3].

Lipton proposed a solution to another NP-complete problem, an instance of the satisfiability (SAT) problem [2]. After the Adleman-Lipton approach, several algorithms explored the ability of DNAC to solve different NP-complete problems in a polynomial time in addition to solving larger instances of previously solved NP problems. Starting from a four-variable instance of the SAT problem [4] and going through six [5] followed by nine [6]. After that a 20-variable 3-SAT problem was solved by Adleman et. al. [3] based on the separation of the sticker model architecture [7].

Xiao et al. [8] explore the characteristics of DNA molecules and reactions in order to realize DNA cryptography and discover possible development directions in the field. Next, a novel public-key system using DNA has been developed [9]. The system uses PCR amplification, followed by sequencing to retrieve the hidden message. In [10] a new forecasting technique, named DNA forecasting, is developed. This may be of use to a nonlinear time series forecasting. Clustering is another area where DNAC was used [11]. A detailed algorithm was illustrated that aims to arriving at a collection of meaningful relationships in data and information granules.

Scheduling is one of the problems that is related to the engineering discipline. Heuristic algorithms were used to solve instances of the problem using *in silico* computers [12, 13]. In [12] the authors presented heuristics for minimizing total
earliness/tardiness on a group of identical parallel machines subject to sequence-dependent setups. The algorithm starts with an initial schedule and then employs a local search procedure to improve this initial schedule when possible.

Several forms of the problem were solved using DNAC. In [14] the job shop scheduling problem was solved. The algorithm uses DNA operations such as ligation, synthesis, and electrophoresis in order to find the optimal solution. In their approach they mimicked the HPP approach by encoding the problem instances as a graph and trying to find the shortest path. [15] presented a design and implementation method to solve an elevator scheduling problem by representing all possible travel path combinations based on certain initial conditions such as present and destination floors, and hall calls from a floor and then extracting the optimal path. Experimental result obtained verifies the successfulness of their approach in which they also used a graph representation. In [16], a simple timetable problem, a special version of the optimization problems, was solved using an advanced biochip technique, laboratory-on-a-chip. Bakar et al. [17, 18] proposed another DNA computing model to solve re-arrangement of flexible manufacturing systems (FMS) in production line. In [19] DNA based algorithms for solving some single machine with limited availability scheduling problems were presented.

The Job Scheduling Problem.

The job scheduling problem is one of the most important industrial engineering problems. It is an NP-hard problem that requires an exponential time to be solved. In its simple form, it can be defined as follows:

Given a set of Jobs \( J = \{J_1, J_2, \ldots, J_n\} \), with duration times \( T_j \) for all \( j = 1, \ldots, n \) such that \( T_j \) is the duration time of job \( J_j \), and a set of equal machines \( \{M_1, M_2, \ldots, M_m\} \), find the best jobs allocation that minimizes the total completion time for the set \( J \). \( c_j \) is the completion time of job \( j \) and can be calculated depending on the start time of job \( j \) \( (s_j) \).

\[
c_j = s_j + T_j
\]

The total completion time \( C \) for the set \( J \) is the longest time taken to execute the jobs assigned to one of the machines since they all start at the same time.

The ideal solution for distributing the set \( J \) on the machines is to divide the jobs equally (according to their durations) on the machines and therefore, the ideal completion time \( t \) will be

\[
t = \frac{\sum_{j=1}^{n} T_j}{m}
\]

Clearly, most of the times it is impossible to distribute the jobs equally on the machines, therefore the optimal completion time \( C \) is: \( C \geq t \)
Methodology

Job Scheduling in Silico

One way to find the optimal solution for distributing $n$ jobs on $m$ machines is to find all possible combinations of assigning jobs to each machine (this includes assigning either one or more jobs), and then choose the shortest time of these assignments. The following algorithm illustrates a technique that can find the shortest completion time of a set of $n$ jobs on $m$ machines.

1. Find all possible combinations of assigning jobs to each machine separately.
2. Combine the results of step 1, which will produce all possible combinations of assigning jobs to all machines.
3. Using step 2 result, extract all job combinations with length equals to the sum of the execution time of all the $n$ jobs. This step eliminates some of the combinations that are not feasible solutions, such as assigning all jobs to all machines.
4. Extract all job combinations that have $j_1, j_2, \ldots, j_n$. This step eliminates combinations that have repetitions, such as assigning job $J_1$ to 2 machines, and ensures that every job is assigned to a machine.
5. Separate the combinations of jobs assigned to machine $M_1$ from the combinations of jobs assigned to machine $M_2$ from the combinations of jobs assigned to machine $M_m$.
6. Finding $C$
   a. Arrange the job combinations $(k)$ on each machine based on their length (completion time). This will find $C_{ik} i = 1, \ldots, m$ and $k = 1, \ldots$ which is the completion time for combination $k$ on machine $M_i$.
   b. Find, for each machine, the minimum completion time that is greater than or equal to $t$.
   c. The optimal completion time for the set of all jobs will be equal to the maximum of the minimum completion time for each machine.

Example

The following example illustrates the above algorithm.

Consider a problem of 4 jobs ($J_1, J_2, J_3, J_4$) and 2 machines ($m_1, m_2$) where the execution time of each job is as illustrated in table (1).
Using DNA Computing to Solve the Scheduling Problem

Table (1): The duration time for the jobs

<table>
<thead>
<tr>
<th>Job</th>
<th>Execution time in hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>J₁</td>
<td>5</td>
</tr>
<tr>
<td>J₂</td>
<td>10</td>
</tr>
<tr>
<td>J₃</td>
<td>3</td>
</tr>
<tr>
<td>J₄</td>
<td>7</td>
</tr>
</tbody>
</table>

Figure (1) represents the main steps of solving the job scheduling problem in silico (using silicon computers) and illustrates applying the previous procedure. Notice that in the first step all possible combinations of jobs including repetitions were constructed. In the second step where the combinations of jobs on the two machines were concatenated together, a symbol was added (M₁, M₂) to distinguish jobs combinations in each machine.

Both steps 3 and 4 ensure that all jobs are taken into consideration with no repetition whether on the same machine or different machines. Step 3 will eliminate combinations like M₁J₁J₂J₃J₄M₂J₅J₆J₁. Step 4 will eliminate combinations that are of length 25 but don’t form a correct solution such as M₁J₁J₃J₄M₂J₅J₁ and M₁J₁J₁J₁M₂J₁J₁.

Now based on the symbol added before, each machine’s jobs can be separated. This is done as a first step in finding the minimum completion time, which depends on the completion time on each machine. After the separation, jobs combinations should be ordered based on their length which represents the completion time for each combination. As discussed earlier, the optimal completion time C should be greater than or equal to the ideal completion time t. Therefore, for each machine the jobs combination with the closest distance from t is chosen.

Based on the previous definition of the total completion time C which is the longest time taken to execute the jobs assigned to one of the machines, the maximum completion time of all the machines is the optimal completion time. Notice that there is only one optimal completion time, but as a jobs allocation on machines there might be more than one combination such as M₁J₁J₄M₂J₂J₃ and M₁J₃J₂M₅J₄J₄ which both give a completion time of 13.
Figure (1): Solving the Job Scheduling *in Silico*
Using DNA Computing to Solve the Scheduling Problem

**Job Scheduling in Vitro**

When using DNAC to solve the job scheduling problem, we will follow the same algorithm above while encoding the problem instances with DNA sequences. The following steps present the DNAC technique that was used to solve the job scheduling problem:

1. **Initialization**
   a. Represent each job with a DNA sequence of a specific length that is proportional to its duration time. (i.e., \( J_1 \), on the previous example, will be represented with a sequence of length 5 since its duration time is 5 as for example GCAGT, while \( J_2 \) will be represented by a sequence of length 10 as for example CCGATAGCAC).
   b. Prepare \( m \) test tubes, where each test tube contains sufficient number of copies of the DNA sequences that represent the \( n \) jobs. The use of \( m \) test tubes is because there are \( m \) machines. Each test tube will contain almost all possible combinations of assigning jobs to a specific machine (as will be performed by step 2).

2. Allow the DNA sequences in each test tube to ligate. With the existence of multiple copies of each DNA sequence, it is likely that almost all possible combinations of these ligations are performed. Therefore, multiple copies of the optimal assignment of jobs to each machine were formed. In order to distinguish jobs in each test tube, which is needed in the coming steps, add a primer to all DNA sequences in the first test tube that is different from the primer added to the sequences in the second test tube … and so on. Note that the primer should be one of the cutting enzymes and it should be added after the ligation.

3. Combine the \( m \) test tubes together (in one test tube) and allow ligation. Since multiple copies exist in the test tube, it is likely that multiple copies of the best distribution of jobs on machines will be formed.

4. Extract DNA sequences that are of length equals to the maximum total completion time + the length of the primers. This can be implemented by agarose gel. The maximum total completion time can be calculated as the sum of all the duration times of all the jobs.

5. Extract the sequences that contain \( J_1, J_2, \ldots, J_n \). This can be done by incubating the single stranded DNA (ssDNA) with \( J_1 \) complement conjugated to magnetic beads. Only those ssDNA sequences which contained the sequence \( J_1 \) at least once will anneal to the bound \( J_1 \) and will be retained. This process should be repeated successively with \( J_2, \ldots, J_n \).

6. Repeat the following steps for \( m-1 \) times (\( i = 2, \ldots m \))
   a. Cut the resulting sequences using the primer \( P_i \) for machine \( M_i \) as the cutting enzyme. This will cut the sequences into 2 parts from the point of \( P_i \), and which will result in the sequences connected to primer \( P_{i-1} \), and the rest of the sequence.
b. Extract the sequences that start with primer $P_{i,1}$ in a separate test tube.

7. Run the contents of each test tube separately on agarose gel electrophoresis and extract the sequences with the length $\geq t$. This will produce $m$ lengths.

8. The minimum completion time for all the jobs will be the maximum length of the $m$ lengths produced in the previous step.

Applying the previous DNAC algorithm to the previous example is shown in Fig(2).
Using DNA Computing to Solve the Scheduling Problem

**Step 1**

**Test Tube 1**

**Step 2**

**Test Tube 1**
Figure (2): Solving the Job Scheduling *in Vitro*
Discussion of the Results

Note that the suggested approach is different from other approaches in that it uses a different length to represent each job. Consequently, the result (which is the optimal solution i.e., the optimal completion time) can be easily concluded from the length of the resulting sequences. If the job’s distribution over machines is what the researcher aims to, then it can be further investigated by sequencing the DNA sequences of the optimal length.

The limitations in our approach is the same as the limitations of DNAC when using the variable length method which is the limited number of lengths or job instances that can be represented with this method. Solving the problem with long duration times will result in long DNA sequences, and this will increase the possibility of errors when using some DNA operations [20].

Conclusion

A new algorithm to solve the job scheduling problem using DNA computing approach has been presented and discussed. Two methodologies were presented to solve the scheduling problem: “job scheduling in silico” which shows the steps of solving the problem using silicon based computers, and "job scheduling in vitro" which illustrates the steps of solving the problem using DNA computers. Both algorithms find the optimal solution or the optimal completion time for a set of jobs. The in silico algorithm finds the solution in an exponential time, while the in vitro algorithm finds the solution in a polynomial number of steps taking advantage of the parallelism ability of DNA. The steps and procedure for each algorithm were illustrated and discussed. The algorithm encodes the problem instances using variable length DNA sequences and then using ligation, PCR, and agarose gel electrophoresis extracts the optimal completion time of the jobs.

This solution further proved the ability of DNAC to solve computational problems. In addition, it proves its ability to solve engineering related problems. Future research may investigate solving the scheduling problem with dependencies.
استخدام الحوسبة بالحمض النووي لمعالجة مشكلة الجدول الزمني

مريم ساري نصير

ملخص

هذا البحث يقترح طريقة جديدة لمعالجة مشكلة الجدول الزمني، والتي تعتبر من المشاكل الصعبة. باستخدام الحوسبة بالحمض النووي، هذه الطرق تعتبر على أسلوب تمثيل الوظائف بأطوال متغيرة. تم تمثيل كل وظيفة بسلسلة حمض نووي، يستمد على مدى الزمنية. تم السماح لسلسلة الأحماض النووية بالاتصال، ثم استخلاص المتسلسلة التي تمثل أقل وقت إنجاز الوظائف. هذا يكتب قدرة الحمض النووي على حل المشكلات الحسابية لذا يمكن اعتباره نقطة انطلاق لحل مشكلة الجدول الزمني. ووجود اعتماد بين الوظائف.

References


Using DNA Computing to Solve the Scheduling Problem


