Int. J. Mol. Sci. **2015**, *16*, 25338-25352; doi:10.3390/ijms161025338

OPEN ACCESS
International Journal of
Molecular Sciences
ISSN 1422-0067
www.mdpi.com/journal/ijms

Article

A Parallel Biological Optimization Algorithm to Solve the Unbalanced Assignment Problem Based on DNA Molecular Computing

Zhaocai Wang [1], Jun Pu [2], Liling Cao [3] and Jian Tan [4,*]

[1] College of Information, Shanghai Ocean University, Shanghai 201306, China; E-Mail: zcwang1028@163.com

[2] Center for Finance and Accounting Research of University of International Business and Economics, Beijing 100029, China; E-Mail: junp1980@163.com

[3] College of Engineering Science and Technology, Shanghai Ocean University, Shanghai 201306, China; E-Mail: appll188@163.com

[4] Key Laboratory of Digital Earth Science, Institute of Remote Sensing and Digital Earth, Chinese Academy of Sciences, Beijing 100094, China

* Author to whom correspondence should be addressed; E-Mail: jtan1980@163.com; Tel./Fax: +86-10-8217-8075.

Academic Editor: Mihai V. Putz

Received: 7 September 2015 / Accepted: 8 October 2015 / Published: 23 October 2015

Abstract: The unbalanced assignment problem (UAP) is to optimally resolve the problem of assigning n jobs to m individuals ($m < n$), such that minimum cost or maximum profit obtained. It is a vitally important Non-deterministic Polynomial (NP) complete problem in operation management and applied mathematics, having numerous real life applications. In this paper, we present a new parallel DNA algorithm for solving the unbalanced assignment problem using DNA molecular operations. We reasonably design flexible-length DNA strands representing different jobs and individuals, take appropriate steps, and get the solutions of the UAP in the proper length range and $O(mn)$ time. We extend the application of DNA molecular operations and simultaneity to simplify the complexity of the computation.

Keywords: DNA molecules computing; the unbalanced assignment problem; biological optimization algorithm; NP-complete problem

1. Introduction

In the pathbreaking work of DNA computation, Adleman [1] firstly described how to solve a seven-node instance of a well-known NP-hard problem utilizing biological operations, and also demonstrated the potential parallel power of DNA computation. In 1995, Lipton [2] proved that Adleman's experiment could be used to figure out the NP-complete satisfiability problem. DNA computation, 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 computation. The advantages imply that we can utilize DNA molecules to solve harder, larger problems such as NP-complete problems in linearly increasing time, in contrast to the exponentially increasing time required by an electronic computer. In recent years, DNA computation has received considerable interest from researchers. Some typical DNA computing models, such as the Adleman-Lipton model [1,2], the sticker model [3], the restriction enzyme model [4], the self-assembly model [5], the hairpin model [6], and the surface-based model [7], have already been established. Based on these models, lots of papers have been written for designing DNA procedures and algorithms to solve various NP-complete problems [8–21]. In order to fully understand the power of biological computation, it is worthwhile to try to solve more kinds of computationally-intractable problems with the aid of DNA biologic operations.

The assignment problem is a common topic in the fields related to operation management and network flow theory. This problem is known to be NP-hard and it is hard from a computational point of view as well. A standard assignment problem is to optimally resolve the problem of assigning *n* jobs to *m* individuals, such that minimum cost or maximum profit can be obtained. Various algorithms, including standard linear programming [6–9], Hungarian algorithm [10], neural network [11], and genetic algorithms [12], have been developed to find solutions. When we deal with real life situation, it becomes quite difficult to ensure that the number of jobs is exactly equal to individuals. Thus, the need arises to solve the unbalanced assignment problem in such a way that total assignment cost may be optimized along with the other constraints. In general, the unbalanced assignment problem can be considered as a particular case of the transportation problem, and can be formulated as a 0–1 integer linear programming [22,23]. The problem can be mathematically formulated as follows:

$$\min Z = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij}$$

$$s.t. \sum_{i=1}^{m} x_{ij} = 1 \; i = 1, 2, \cdots, m$$

$$1 \le \sum_{j=1}^{n} x_{ij} \le n - m + 1 \; j = 1, 2, \cdots, n$$

$$x_{ij} = 0 \text{, or } 1$$

where the decision variable $x_{ij} = 1$ means that the *j*-th job is assigned to the *i*-th individual; otherwise, $x_{ij} = 0$ is in reverse; c_{ij} is the associated cost incurred by the assignment (if c_{ij} is the correlative profit, the object function will be max *Z*). For instance, the Table 1 defines the cost matrix A = $[c_{ij}]_{m \times n}$ of the unbalanced assignment problem.

Table 1. Cost matrix A = $[c_{ij}]_{3\times 5}$.

Cost	j_1	j_2	j_3	j_4	j_5
i_1	5	9	1	2	7
i_2	9	8	6	4	4
i_3	4	7	8	5	2

In this paper, based on a combination of Adleman-Lipton model, A theoretically-efficient DNA algorithm is introduced for figuring out solutions of the unbalanced assignment problem, which is executed in $O(mn)$ operations, where n is number of jobs and m is number of individuals. With the progress of molecular biology techniques, the proposed algorithm might be of practical use in treating medium-sized instances of UAP.

The rest of this paper is organized as follows. In Section 2, the Adleman-Lipton model is introduced in detail. Then, we use a DNA molecular algorithm for solving the unbalanced assignment problem. And prove DNA algorithm complexity and feasibility. In Section 3, we use computer to simulate the DNA experiment and get correct solution of the Table 1. We get conclusions in Section 4.

2. Results and Discussion

2.1. The Adleman-Lipton Model

The DNA operations proposed by Adleman [1] and Lipton [2] are described below. These operations will be used for figuring out solutions of the unbalanced assignment problem in this paper. In the Adleman-Lipton 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 and 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) *Sort*(T_1,T_2,T_3): for a given test tube T_1, it choose the shortest length strands in the tube T_2, the longest strands in T_3 and the remaining strands in T_1;

(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) *Ligation*(T): for a given tube T, the operation is used to ligate together the strands in T;

(10) *Discard*(T): for a given test tube T, it discards the tube T;

(11) *Read*(*T*): for a given 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;

(12) *Append-tail*(*T*,*Z*): for a given test tube *T* and a given DNA singled strand, it appends *Z* onto the end of every strand in the tube *T*.

Since these twelve manipulations are implemented with a constant number of biological steps for DNA strands [24,25], we assume that the complexity of each manipulation is in $O(1)$ time steps.

2.2. DNA Algorithm for the Unbalanced Assignment Problem

2.2.1. Thinking Process

The initial idea to solve the unbalanced assignment problem is as followed: generate strands corresponding to all possible job allocation schemes in a data pool, then, filter out inappropriate job allocation. Next, append the cost-weighted length strands in order to identify the schemes' pros and cons. Finally, obtain the optimal solutions of the unbalanced assignment problem by using the corresponding DNA operations. Concretely, the proposed algorithm has four steps.

Step 1: Construct set *T* of all possible m^n solutions for unbalanced assignment problem;

Step 2: For all possible solutions, eliminate inappropriate allocation, such as one job distribution to multiple individuals, one job without been assigned.

Step 3: Append time weight chain at the corresponding qualified strands in order to find the optimal solution.

Step 4: Get the shortest strands as the answer to the problem and identify the specific distribution.

2.2.2. Detailed DNA Algorithm

Given a set of n jobs and m individuals ($m < n$), the unbalanced assignment problem requires that each job should be allocated to only one individual that the total cost, which is defined as the sum of the cost between each pair job and individual, is minimized. Consider a problem which consists of a set of *m* individuals $I = \{i_1,i_2,\dots,i_m\}$. A set of n jobs $J = \{j_1,j_2,\dots,j_n\}$ is considered which are to be assigned for execution by *m* available individuals.

We suppose $m < n$. The execution cost of each job by all the individuals is known and mentioned in the matrix, namely $A = [c_{ij}]_{m \times n}$ where c_{ij} is the cost between job *j* and individual *i*. The objective is to determine the optimal assignment cost. A method is devised to obtain the said costs in such a way that all the jobs are to be allotted on the available individuals. In the following, the symbols s, e, A_i, B_j ($i = 1,2,\dots,m$, $j = 1,2,\dots,n$) denote distinct DNA single strands with the same length, say *t mer* (*t* is a positive integer, *mer* is monomer unit length). Obviously the length t of the DNA single strands greatly depends on the size of the problem involved in order to distinguish all above symbols [25]. Meanwhile we use the symbols w_{ij} to denote the cost c_{ij} and $||w_{ij}|| = c_{ij}$. Then, in the below operations, we use the distinct DNA single strand symbols sA_ieB_j ($i = 1,2,\dots,m$, $j = 1,2,\dots,n$) to denote that the *j*-th job is assigned to the *i*-th individual without cost information. Simultaneity the symbols s, e are the signal of different edges division. Let

$$P = \{sA_1e, sA_2e, \cdots, sA_me\}$$
$$Q = \{\overline{eB_1s}, \overline{eB_2s}, \cdots, \overline{eB_ns}, B_1, B_2, \cdots, B_n\}$$

For a cost matrix A = $[c_{ij}]_{m \times n}$, the unbalanced assignment problem is firstly on the relation between *n* jobs and *m* individuals. Every possible assignment can be expressed by a list of DNA strands. DNA strands sA_ieB_j represent that the *j*-th job is assigned to the *i*-th individual. For example in Table 1, the DNA strands $\{sA_1eB_2sA_2eB_3sA_3eB_4sA_2eB_1sA_1eB_5\}$ represent the 2 and 5-job to 1-individual, 1 and 3-job to 2-individual and 4-job to 3-individual. In this way, we can get DNA strands representing all possible *n* jobs to *m* individuals allocation relation.

(1) We choose all possible DNA strands denoting *n* jobs to *m* individuals allocation relation.

```
(1-1) Merge(P,Q);
(1-2) Annealing(P);
(1-3) Ligation(P);
(1-4) Denaturation(P);
(1-5) Separation(P,{s},$T_1$);
(1-6) Selection($T_1$,4nt,$T_2$).
```

This step operation can be finished in *O*(1) time steps since each manipulation above works in *O*(1).

(2) The unbalanced assignment problem requires that each job is assigned to a unique individual. So we check above DNA strands whether they satisfy the condition or not. The above assignment DNA strands should contain job strands B_k ($1 \leq k \leq n$) one time only. For example in Table 1, the single strands $\{sA_1eB_3sA_3eB_3sA_2eB_1sA_1eB_2sA_2eB_4\} \in T_2$ should be discarded for the 3-job simultaneously assigned to 1 and 3-individuals and not assigned the 5-job. We get all feasible assignment strands as follow:

```
For k = 1 to k = n
(2-1) Separation($T_2$,{$eB_ks$},$T_3$);
(2-2) Discard($T_2$);
(2-3) Copy($T_3$,$T_2$);
(2-4) Discard($T_3$).
End for
```

In the above operations, we use one "For" clauses, thus this operation can be finished in *O*(*n*) time steps.

(3) In addition, the unbalanced assignment problem require that every individual get at least one job. So the above assignment DNA strands should contain individual strands A_i ($1 \leq i \leq m$) at least one time. For example in Table 1, the single strands $\{sA_1eB_3sA_2eB_4sA_2eB_1sA_1eB_2sA_2eB_5\} \in T_2$ should be discarded for not including the 3-individual. We get all feasible assignment strands as follow:

```
For k = 1 to k = n
(3-1) Separation($T_2$,{$sA_ke$},$T_4$);
(3-2) Discard($T_2$);
(3-3) Copy($T_4$,$T_2$);
(3-4) Discard($T_4$).
End for
```

In the above operations, we use one "For" clause; thus, this operation can be finished in *O*(*m*) time steps.

(4) The solutions of unbalanced assignment problem must be with the minimum cost. In order to find the optimal results, we append the cost information strands at the end of above strands. For example,

for the Table 1, the singled strands $\{sA_1eB_3sA_3eB_2sA_3eB_4sA_2eB_1sA_2eB_3\} \in T_2$ representing the allocation: $\{j_3\rightarrow i_1, j_2\rightarrow i_3, j_4\rightarrow i_3, j_1\rightarrow i_2, j_5\rightarrow i_2\}$, we append strands $\{w_{14}, w_{21}, w_{32}, w_{43}\}$ at the above-mentioned strands to $\{sA_1eB_3sA_3eB_2sA_3eB_4sA_2eB_1sA_2eB_3w_{13}w_{32}w_{34}w_{21}w_{25}\}$.

This is done by the following manipulations:

```
For $i = 1$ to $i = m$
    For $j = 1$ to $j = n$
(4-1) Separation($T_2$,{$sA_ieB_j$},$T_5$);
(4-2) If (Detect($T_5$))
    Then execute (4-3) to (4-5)
(4-3) Append-tail($T_5$,$w_{ij}$);
(4-4) Merge($T_2$,$T_5$);
(4-5) Discard($T_5$).
    End for
End for
```

In the above operation, this operation can be finished in $O(mn)$ time steps since we use two "For" clauses with m and n circulation.

(5) We take out those single strands in T_2 with the shortest length, which give the solutions to the unbalanced assignment problem. For example in Table 1, those single strands in T2with shortest length are $\{sA_1eB_3sA_1eB_4sA_2eB_2sA_3eB_1sA_3eB_5w_{13}w_{14}w_{22}w_{31}w_{35}\}$.

Therefore, solutions to unbalanced assignment problem for Table 1 are $\{j_3\rightarrow i_1, j_4\rightarrow i_1, j_2\rightarrow i_2, j_1\rightarrow i_3, j_5\rightarrow i_3\}$ with the weight sum 17.

```
(5-1) Sort($T_2$,$T_5$,$T_6$);
(5-2) Read($T_6$).
```

In the above operation, this operation can be finished in $O(1)$ time steps since each single manipulation above works in $O(1)$ steps. Finally the "Read" operation is applied to giving the exact solutions to the unbalanced assignment problem.

2.3. A Simple Example

We take a simple example in Table 2 to walk through the entire DNA algorithm.

Table 2. Cost matrix A = $[c_{ij}]_{2\times 3}$.

Cost	j_1	j_2	j_3
i_1	2	4	1
i_2	1	2	3

After Step (1), all possible job assignment projects are shown in Table 3. For the unbalanced assignment problem, every job should be assigned to someone. The strands' symbols after Step (2) are shown in Table 4. Table 5 displays the strand symbols, which mean every individual gets at least one job. Subsequently, the corresponding weight strands are attached to the original ones as shown in Table 6 and we get the solution strands in Table 7.

Table 3. DNA Sequences symbols after Step (1) for Table 2.

3′–5′ DNA Sequence	3′–5′ DNA Sequence	3′–5′ DNA Sequence	3′–5′ DNA Sequence
$sA_1eB_1sA_1eB_1sA_1eB_1$	$sA_1eB_1sA_1eB_1sA_1eB_2$	$sA_1eB_1sA_1eB_1sA_1eB_3$	$sA_1eB_1sA_1eB_1sA_2eB_1$
$sA_1eB_1sA_1eB_1sA_2eB_2$	$sA_1eB_1sA_1eB_1sA_2eB_3$	$sA_1eB_1sA_1eB_2sA_1eB_1$	$sA_1eB_1sA_1eB_2sA_1eB_2$
$sA_1eB_1sA_1eB_2sA_1eB_3$	$sA_1eB_1sA_1eB_2sA_2eB_1$	$sA_1eB_1sA_1eB_2sA_2eB_2$	$sA_1eB_1sA_1eB_2sA_2eB_3$
$sA_1eB_1sA_1eB_3sA_1eB_1$	$sA_1eB_1sA_1eB_3sA_1eB_2$	$sA_1eB_1sA_1eB_3sA_1eB_3$	$sA_1eB_1sA_1eB_3sA_2eB_1$
$sA_1eB_1sA_1eB_3sA_2eB_2$	$sA_1eB_1sA_1eB_3sA_2eB_3$	$sA_1eB_1sA_2eB_1sA_1eB_1$	$sA_1eB_1sA_2eB_1sA_1eB_2$
$sA_1eB_1sA_2eB_1sA_1eB_3$	$sA_1eB_1sA_2eB_1sA_2eB_1$	$sA_1eB_1sA_2eB_1sA_2eB_2$	$sA_1eB_1sA_2eB_1sA_2eB_3$
$sA_1eB_1sA_2eB_2sA_1eB_1$	$sA_1eB_1sA_2eB_2sA_1eB_2$	$sA_1eB_1sA_2eB_2sA_1eB_3$	$sA_1eB_1sA_2eB_2sA_2eB_1$
$sA_1eB_1sA_2eB_2sA_2eB_2$	$sA_1eB_1sA_2eB_2sA_2eB_3$	$sA_1eB_1sA_2eB_3sA_1eB_1$	$sA_1eB_1sA_2eB_3sA_1eB_2$
$sA_1eB_1sA_2eB_3sA_1eB_3$	$sA_1eB_1sA_2eB_3sA_2eB_1$	$sA_1eB_1sA_2eB_3sA_2eB_2$	$sA_1eB_1sA_2eB_3sA_2eB_3$
$sA_1eB_2sA_1eB_1sA_1eB_1$	$sA_1eB_2sA_1eB_1sA_1eB_2$	$sA_1eB_2sA_1eB_1sA_1eB_3$	$sA_1eB_2sA_1eB_1sA_2eB_1$
$sA_1eB_2sA_1eB_1sA_2eB_2$	$sA_1eB_2sA_1eB_1sA_2eB_3$	$sA_1eB_2sA_1eB_2sA_1eB_1$	$sA_1eB_2sA_1eB_2sA_1eB_2$
$sA_1eB_2sA_1eB_2sA_1eB_3$	$sA_1eB_2sA_1eB_2sA_2eB_1$	$sA_1eB_2sA_1eB_2sA_2eB_2$	$sA_1eB_2sA_1eB_2sA_2eB_3$
$sA_1eB_2sA_1eB_3sA_1eB_1$	$sA_1eB_2sA_1eB_3sA_1eB_2$	$sA_1eB_2sA_1eB_3sA_1eB_3$	$sA_1eB_2sA_1eB_3sA_2eB_1$
$sA_1eB_2sA_1eB_3sA_2eB_2$	$sA_1eB_2sA_1eB_3sA_2eB_3$	$sA_1eB_2sA_2eB_1sA_1eB_1$	$sA_1eB_2sA_2eB_1sA_1eB_2$
$sA_1eB_2sA_2eB_1sA_1eB_3$	$sA_1eB_2sA_2eB_1sA_2eB_1$	$sA_1eB_2sA_2eB_1sA_2eB_2$	$sA_1eB_2sA_2eB_1sA_2eB_3$
$sA_1eB_2sA_2eB_2sA_1eB_1$	$sA_1eB_2sA_2eB_2sA_1eB_2$	$sA_1eB_2sA_2eB_2sA_1eB_3$	$sA_1eB_2sA_2eB_2sA_2eB_1$
$sA_1eB_2sA_2eB_2sA_2eB_2$	$sA_1eB_2sA_2eB_2sA_2eB_3$	$sA_1eB_2sA_2eB_3sA_1eB_1$	$sA_1eB_2sA_2eB_3sA_1eB_2$
$sA_1eB_2sA_2eB_3sA_1eB_3$	$sA_1eB_2sA_2eB_3sA_2eB_1$	$sA_1eB_2sA_2eB_3sA_2eB_2$	$sA_1eB_2sA_2eB_3sA_2eB_3$
$sA_1eB_3sA_1eB_1sA_1eB_1$	$sA_1eB_3sA_1eB_1sA_1eB_2$	$sA_1eB_3sA_1eB_1sA_1eB_3$	$sA_1eB_3sA_1eB_1sA_2eB_1$
$sA_1eB_3sA_1eB_1sA_2eB_2$	$sA_1eB_3sA_1eB_1sA_2eB_3$	$sA_1eB_3sA_1eB_2sA_1eB_1$	$sA_1eB_3sA_1eB_2sA_1eB_2$
$sA_1eB_3sA_1eB_2sA_1eB_3$	$sA_1eB_3sA_1eB_2sA_2eB_1$	$sA_1eB_3sA_1eB_2sA_2eB_2$	$sA_1eB_3sA_1eB_2sA_2eB_3$
$sA_1eB_3sA_1eB_3sA_1eB_1$	$sA_1eB_3sA_1eB_3sA_1eB_2$	$sA_1eB_3sA_1eB_3sA_1eB_3$	$sA_1eB_3sA_1eB_3sA_2eB_1$
$sA_1eB_3sA_1eB_3sA_2eB_2$	$sA_1eB_3sA_1eB_3sA_2eB_3$	$sA_1eB_3sA_2eB_1sA_1eB_1$	$sA_1eB_3sA_2eB_1sA_1eB_2$
$sA_1eB_3sA_2eB_1sA_1eB_3$	$sA_1eB_3sA_2eB_1sA_2eB_1$	$sA_1eB_3sA_2eB_1sA_2eB_2$	$sA_1eB_3sA_2eB_1sA_2eB_3$
$sA_1eB_3sA_2eB_2sA_1eB_1$	$sA_1eB_3sA_2eB_2sA_1eB_2$	$sA_1eB_3sA_2eB_2sA_1eB_3$	$sA_1eB_3sA_2eB_2sA_2eB_1$
$sA_1eB_3sA_2eB_2sA_2eB_2$	$sA_1eB_3sA_2eB_2sA_2eB_3$	$sA_1eB_3sA_2eB_3sA_1eB_1$	$sA_1eB_3sA_2eB_3sA_1eB_2$
$sA_1eB_3sA_2eB_3sA_1eB_3$	$sA_1eB_3sA_2eB_3sA_2eB_1$	$sA_1eB_3sA_2eB_3sA_2eB_2$	$sA_1eB_3sA_2eB_3sA_2eB_3$
$sA_2eB_1sA_1eB_1sA_1eB_1$	$sA_2eB_1sA_1eB_1sA_1eB_2$	$sA_2eB_1sA_1eB_1sA_1eB_3$	$sA_2eB_1sA_1eB_1sA_2eB_1$
$sA_2eB_1sA_1eB_1sA_2eB_2$	$sA_2eB_1sA_1eB_1sA_2eB_3$	$sA_2eB_1sA_1eB_2sA_1eB_1$	$sA_2eB_1sA_1eB_2sA_1eB_2$
$sA_2eB_1sA_1eB_2sA_1eB_3$	$sA_2eB_1sA_1eB_2sA_2eB_1$	$sA_2eB_1sA_1eB_2sA_2eB_2$	$sA_2eB_1sA_1eB_2sA_2eB_3$
$sA_2eB_1sA_1eB_3sA_1eB_1$	$sA_2eB_1sA_1eB_3sA_1eB_2$	$sA_2eB_1sA_1eB_3sA_1eB_3$	$sA_2eB_1sA_1eB_3sA_2eB_1$
$sA_2eB_1sA_1eB_3sA_2eB_2$	$sA_2eB_1sA_1eB_3sA_2eB_3$	$sA_2eB_1sA_2eB_1sA_1eB_1$	$sA_2eB_1sA_2eB_1sA_1eB_2$
$sA_2eB_1sA_2eB_1sA_1eB_3$	$sA_2eB_1sA_2eB_1sA_2eB_1$	$sA_2eB_1sA_2eB_1sA_2eB_2$	$sA_2eB_1sA_2eB_1sA_2eB_3$
$sA_2eB_1sA_2eB_2sA_1eB_1$	$sA_2eB_1sA_2eB_2sA_1eB_2$	$sA_2eB_1sA_2eB_2sA_1eB_3$	$sA_2eB_1sA_2eB_2sA_2eB_1$
$sA_2eB_1sA_2eB_2sA_2eB_2$	$sA_2eB_1sA_2eB_2sA_2eB_3$	$sA_2eB_1sA_2eB_3sA_1eB_1$	$sA_2eB_1sA_2eB_3sA_1eB_2$
$sA_2eB_1sA_2eB_3sA_1eB_3$	$sA_2eB_1sA_2eB_3sA_2eB_1$	$sA_2eB_1sA_2eB_3sA_2eB_2$	$sA_2eB_1sA_2eB_3sA_2eB_3$
$sA_2eB_2sA_1eB_1sA_1eB_1$	$sA_2eB_2sA_1eB_1sA_1eB_2$	$sA_2eB_2sA_1eB_1sA_1eB_3$	$sA_2eB_2sA_1eB_1sA_2eB_1$
$sA_2eB_2sA_1eB_1sA_2eB_2$	$sA_2eB_2sA_1eB_1sA_2eB_3$	$sA_2eB_2sA_1eB_2sA_1eB_1$	$sA_2eB_2sA_1eB_2sA_1eB_2$
$sA_2eB_2sA_1eB_2sA_1eB_3$	$sA_2eB_2sA_1eB_2sA_2eB_1$	$sA_2eB_2sA_1eB_2sA_2eB_2$	$sA_2eB_2sA_1eB_2sA_2eB_3$
$sA_2eB_2sA_1eB_3sA_1eB_1$	$sA_2eB_2sA_1eB_3sA_1eB_2$	$sA_2eB_2sA_1eB_3sA_1eB_3$	$sA_2eB_2sA_1eB_3sA_2eB_1$
$sA_2eB_2sA_1eB_3sA_2eB_2$	$sA_2eB_2sA_1eB_3sA_2eB_3$	$sA_2eB_2sA_2eB_1sA_1eB_1$	$sA_2eB_2sA_2eB_1sA_1eB_2$
$sA_2eB_2sA_2eB_1sA_1eB_3$	$sA_2eB_2sA_2eB_1sA_2eB_1$	$sA_2eB_2sA_2eB_1sA_2eB_2$	$sA_2eB_2sA_2eB_1sA_2eB_3$
$sA_2eB_2sA_2eB_2sA_1eB_1$	$sA_2eB_2sA_2eB_2sA_1eB_2$	$sA_2eB_2sA_2eB_2sA_1eB_3$	$sA_2eB_2sA_2eB_2sA_2eB_1$

Table 3. *Cont.*

3′–5′ DNA Sequence	**3′–5′ DNA Sequence**	**3′–5′ DNA Sequence**	**3′–5′ DNA Sequence**
$sA_2eB_2sA_2eB_2sA_2eB_2$	$sA_2eB_2sA_2eB_2sA_2eB_3$	$sA_2eB_2sA_2eB_3sA_1eB_1$	$sA_2eB_2sA_2eB_3sA_1eB_2$
$sA_2eB_2sA_2eB_3sA_1eB_3$	$sA_2eB_2sA_2eB_3sA_2eB_1$	$sA_2eB_2sA_2eB_3sA_2eB_2$	$sA_2eB_2sA_2eB_3sA_2eB_3$
$sA_2eB_3sA_1eB_1sA_1eB_1$	$sA_2eB_3sA_1eB_1sA_1eB_2$	$sA_2eB_3sA_1eB_1sA_1eB_3$	$sA_2eB_3sA_1eB_1sA_2eB_1$
$sA_2eB_3sA_1eB_1sA_2eB_2$	$sA_2eB_3sA_1eB_1sA_2eB_3$	$sA_2eB_3sA_1eB_2sA_1eB_1$	$sA_2eB_3sA_1eB_2sA_1eB_2$
$sA_2eB_3sA_1eB_2sA_1eB_3$	$sA_2eB_3sA_1eB_2sA_2eB_1$	$sA_2eB_3sA_1eB_2sA_2eB_2$	$sA_2eB_3sA_1eB_2sA_2eB_3$
$sA_2eB_3sA_1eB_3sA_1eB_1$	$sA_2eB_3sA_1eB_3sA_1eB_2$	$sA_2eB_3sA_1eB_3sA_1eB_3$	$sA_2eB_3sA_1eB_3sA_2eB_1$
$sA_2eB_3sA_1eB_3sA_2eB_2$	$sA_2eB_3sA_1eB_3sA_2eB_3$	$sA_2eB_3sA_2eB_1sA_1eB_1$	$sA_2eB_3sA_2eB_1sA_1eB_2$
$sA_2eB_3sA_2eB_1sA_1eB_3$	$sA_2eB_3sA_2eB_1sA_2eB_1$	$sA_2eB_3sA_2eB_1sA_2eB_2$	$sA_2eB_3sA_2eB_1sA_2eB_3$
$sA_2eB_3sA_2eB_2sA_1eB_1$	$sA_2eB_3sA_2eB_2sA_1eB_2$	$sA_2eB_3sA_2eB_2sA_1eB_3$	$sA_2eB_3sA_2eB_2sA_2eB_1$
$sA_2eB_3sA_2eB_2sA_2eB_2$	$sA_2eB_3sA_2eB_2sA_2eB_3$	$sA_2eB_3sA_2eB_3sA_1eB_1$	$sA_2eB_3sA_2eB_3sA_1eB_2$
$sA_2eB_3sA_2eB_3sA_1eB_3$	$sA_2eB_3sA_2eB_3sA_2eB_1$	$sA_2eB_3sA_2eB_3sA_2eB_2$	$sA_2eB_3sA_2eB_3sA_2eB_3$

Table 4. DNA Sequences symbols after Step (2) for Table 2.

3′–5′ DNA Sequence	**3′–5′ DNA Sequence**	**3′–5′ DNA Sequence**	**3′–5′ DNA Sequence**
$sA_1eB_1sA_1eB_1sA_2eB_1$	$sA_1eB_1sA_1eB_1sA_2eB_2$	$sA_1eB_1sA_1eB_1sA_2eB_3$	$sA_1eB_1sA_1eB_2sA_1eB_1$
$sA_1eB_1sA_1eB_2sA_1eB_2$	$sA_1eB_1sA_1eB_2sA_1eB_3$	$sA_1eB_1sA_1eB_2sA_2eB_1$	$sA_1eB_1sA_1eB_2sA_2eB_2$
$sA_1eB_1sA_1eB_2sA_2eB_3$	$sA_1eB_1sA_1eB_3sA_1eB_1$	$sA_1eB_1sA_1eB_3sA_1eB_2$	$sA_1eB_1sA_1eB_3sA_1eB_3$
$sA_1eB_1sA_1eB_3sA_2eB_1$	$sA_1eB_1sA_1eB_3sA_2eB_2$	$sA_1eB_1sA_1eB_3sA_2eB_3$	$sA_1eB_1sA_2eB_1sA_1eB_1$
$sA_1eB_1sA_2eB_1sA_1eB_2$	$sA_1eB_1sA_2eB_1sA_1eB_3$	$sA_1eB_1sA_2eB_1sA_2eB_1$	$sA_1eB_1sA_2eB_1sA_2eB_2$
$sA_1eB_1sA_2eB_1sA_2eB_3$	$sA_1eB_1sA_2eB_2sA_1eB_1$	$sA_1eB_1sA_2eB_2sA_1eB_2$	$sA_1eB_1sA_2eB_2sA_1eB_3$
$sA_1eB_1sA_2eB_2sA_2eB_1$	$sA_1eB_1sA_2eB_2sA_2eB_2$	$sA_1eB_1sA_2eB_2sA_2eB_3$	$sA_1eB_1sA_2eB_3sA_1eB_1$
$sA_1eB_1sA_2eB_3sA_1eB_2$	$sA_1eB_1sA_2eB_3sA_1eB_3$	$sA_1eB_1sA_2eB_3sA_2eB_1$	$sA_1eB_1sA_2eB_3sA_2eB_2$
$sA_1eB_1sA_2eB_3sA_2eB_3$	$sA_1eB_2sA_1eB_1sA_1eB_1$	$sA_1eB_2sA_1eB_1sA_1eB_2$	$sA_1eB_2sA_1eB_1sA_1eB_3$
$sA_1eB_2sA_1eB_1sA_2eB_1$	$sA_1eB_2sA_1eB_1sA_2eB_2$	$sA_1eB_2sA_1eB_1sA_2eB_3$	$sA_1eB_2sA_1eB_2sA_1eB_1$
$sA_1eB_2sA_1eB_2sA_1eB_2$	$sA_1eB_2sA_1eB_2sA_1eB_3$	$sA_1eB_2sA_1eB_2sA_2eB_1$	$sA_1eB_2sA_1eB_2sA_2eB_2$
$sA_1eB_2sA_1eB_2sA_2eB_3$	$sA_1eB_2sA_1eB_3sA_1eB_1$	$sA_1eB_2sA_1eB_3sA_1eB_2$	$sA_1eB_2sA_1eB_3sA_1eB_3$
$sA_1eB_2sA_1eB_3sA_2eB_1$	$sA_1eB_2sA_1eB_3sA_2eB_2$	$sA_1eB_2sA_1eB_3sA_2eB_3$	$sA_1eB_2sA_2eB_1sA_1eB_1$
$sA_1eB_2sA_2eB_1sA_1eB_2$	$sA_1eB_2sA_2eB_1sA_1eB_3$	$sA_1eB_2sA_2eB_1sA_2eB_1$	$sA_1eB_2sA_2eB_1sA_2eB_2$
$sA_1eB_2sA_2eB_1sA_2eB_3$	$sA_1eB_2sA_2eB_2sA_1eB_1$	$sA_1eB_2sA_2eB_2sA_1eB_2$	$sA_1eB_2sA_2eB_2sA_1eB_3$
$sA_1eB_2sA_2eB_2sA_2eB_1$	$sA_1eB_2sA_2eB_2sA_2eB_2$	$sA_1eB_2sA_2eB_2sA_2eB_3$	$sA_1eB_2sA_2eB_3sA_1eB_1$
$sA_1eB_2sA_2eB_3sA_1eB_2$	$sA_1eB_2sA_2eB_3sA_1eB_3$	$sA_1eB_2sA_2eB_3sA_2eB_1$	$sA_1eB_2sA_2eB_3sA_2eB_2$
$sA_1eB_2sA_2eB_3sA_2eB_3$	$sA_1eB_3sA_1eB_1sA_1eB_1$	$sA_1eB_3sA_1eB_1sA_1eB_2$	$sA_1eB_3sA_1eB_1sA_1eB_3$
$sA_1eB_3sA_1eB_1sA_2eB_1$	$sA_1eB_3sA_1eB_1sA_2eB_2$	$sA_1eB_3sA_1eB_1sA_2eB_3$	$sA_1eB_3sA_1eB_2sA_1eB_1$
$sA_1eB_3sA_1eB_2sA_1eB_2$	$sA_1eB_3sA_1eB_2sA_1eB_3$	$sA_1eB_3sA_1eB_2sA_2eB_1$	$sA_1eB_3sA_1eB_2sA_2eB_2$
$sA_1eB_3sA_1eB_2sA_2eB_3$	$sA_1eB_3sA_1eB_3sA_1eB_1$	$sA_1eB_3sA_1eB_3sA_1eB_2$	$sA_1eB_3sA_1eB_3sA_1eB_3$
$sA_1eB_3sA_1eB_3sA_2eB_1$	$sA_1eB_3sA_1eB_3sA_2eB_2$	$sA_1eB_3sA_1eB_3sA_2eB_3$	$sA_1eB_3sA_2eB_1sA_1eB_1$
$sA_1eB_3sA_2eB_1sA_1eB_2$	$sA_1eB_3sA_2eB_1sA_1eB_3$	$sA_1eB_3sA_2eB_1sA_2eB_1$	$sA_1eB_3sA_2eB_1sA_2eB_2$
$sA_1eB_3sA_2eB_1sA_2eB_3$	$sA_1eB_3sA_2eB_2sA_1eB_1$	$sA_1eB_3sA_2eB_2sA_1eB_2$	$sA_1eB_3sA_2eB_2sA_1eB_3$
$sA_1eB_3sA_2eB_2sA_2eB_1$	$sA_1eB_3sA_2eB_2sA_2eB_2$	$sA_1eB_3sA_2eB_2sA_2eB_3$	$sA_1eB_3sA_2eB_3sA_1eB_1$
$sA_1eB_3sA_2eB_3sA_1eB_2$	$sA_1eB_3sA_2eB_3sA_1eB_3$	$sA_1eB_3sA_2eB_3sA_2eB_1$	$sA_1eB_3sA_2eB_3sA_2eB_2$
$sA_1eB_3sA_2eB_3sA_2eB_3$	$sA_2eB_1sA_1eB_1sA_1eB_1$	$sA_2eB_1sA_1eB_1sA_1eB_2$	$sA_2eB_1sA_1eB_1sA_1eB_3$

Table 5. DNA Sequences symbols after Step (1) for Table 2.

3′–5′ DNA Sequence	**3′–5′ DNA Sequence**	**3′–5′ DNA Sequence**	**3′–5′ DNA Sequence**
$sA_1eB_1sA_1eB_2sA_2eB_3$	$sA_1eB_1sA_1eB_3sA_2eB_2$	$sA_1eB_1sA_2eB_2sA_1eB_3$	$sA_1eB_1sA_2eB_2sA_2eB_3$
$sA_1eB_1sA_2eB_3sA_1eB_2$	$sA_1eB_1sA_2eB_3sA_2eB_2$	$sA_1eB_2sA_1eB_1sA_2eB_3$	$sA_1eB_2sA_1eB_3sA_2eB_1$
$sA_1eB_2sA_2eB_1sA_1eB_3$	$sA_1eB_2sA_2eB_1sA_2eB_3$	$sA_1eB_2sA_2eB_3sA_1eB_1$	$sA_1eB_2sA_2eB_3sA_2eB_1$
$sA_1eB_3sA_1eB_1sA_2eB_2$	$sA_1eB_3sA_1eB_2sA_2eB_1$	$sA_1eB_3sA_2eB_1sA_1eB_2$	$sA_1eB_3sA_2eB_1sA_2eB_2$
$sA_1eB_3sA_2eB_2sA_1eB_1$	$sA_1eB_3sA_2eB_2sA_2eB_1$	$sA_2eB_1sA_1eB_2sA_1eB_3$	$sA_2eB_1sA_1eB_2sA_2eB_3$
$sA_2eB_1sA_1eB_3sA_1eB_2$	$sA_2eB_1sA_1eB_3sA_2eB_2$	$sA_2eB_1sA_2eB_2sA_1eB_3$	$sA_2eB_1sA_2eB_3sA_1eB_2$
$sA_2eB_2sA_1eB_1sA_1eB_3$	$sA_2eB_2sA_1eB_1sA_2eB_3$	$sA_2eB_2sA_1eB_3sA_1eB_1$	$sA_2eB_2sA_1eB_3sA_2eB_1$
$sA_2eB_2sA_2eB_1sA_1eB_3$	$sA_2eB_2sA_2eB_3sA_1eB_1$	$sA_2eB_3sA_1eB_1sA_1eB_2$	$sA_2eB_3sA_1eB_1sA_2eB_2$
$sA_2eB_3sA_1eB_2sA_1eB_1$	$sA_2eB_3sA_1eB_2sA_2eB_1$	$sA_2eB_3sA_2eB_1sA_1eB_2$	$sA_2eB_3sA_2eB_2sA_1eB_1$

Table 6. DNA Sequences symbols after Step (4) for Table 2.

3′–5′ DNA Sequence	**3′–5′ DNA Sequence**	**3′–5′ DNA Sequence**
$sA_1eB_1sA_1eB_2sA_2eB_3w_{11}w_{12}w_{23}$	$sA_1eB_1sA_1eB_3sA_2eB_2w_{11}w_{13}w_{22}$	$sA_1eB_1sA_2eB_2sA_1eB_3w_{11}w_{13}w_{22}$
$sA_1eB_1sA_2eB_2sA_2eB_3w_{11}w_{22}w_{23}$	$sA_1eB_1sA_2eB_3sA_1eB_2w_{11}w_{12}w_{23}$	$sA_1eB_1sA_2eB_3sA_2eB_2w_{11}w_{22}w_{23}$
$sA_1eB_2sA_1eB_1sA_2eB_3w_{11}w_{12}w_{23}$	$sA_1eB_2sA_1eB_3sA_2eB_1w_{12}w_{13}w_{21}$	$sA_1eB_2sA_2eB_1sA_1eB_3w_{12}w_{13}w_{21}$
$sA_1eB_2sA_2eB_1sA_2eB_3w_{12}w_{21}w_{23}$	$sA_1eB_2sA_2eB_3sA_1eB_1w_{11}w_{12}w_{23}$	$sA_1eB_2sA_2eB_3sA_2eB_1w_{12}w_{21}w_{23}$
$sA_1eB_3sA_1eB_1sA_2eB_2w_{11}w_{13}w_{22}$	$sA_1eB_3sA_1eB_2sA_2eB_1w_{12}w_{13}w_{21}$	$sA_1eB_3sA_2eB_1sA_1eB_2w_{12}w_{13}w_{21}$
$sA_1eB_3sA_2eB_1sA_2eB_2w_{13}w_{21}w_{22}$	$sA_1eB_3sA_2eB_2sA_1eB_1w_{11}w_{13}w_{22}$	$sA_1eB_3sA_2eB_2sA_2eB_1w_{13}w_{21}w_{22}$
$sA_2eB_1sA_1eB_2sA_1eB_3w_{12}w_{13}w_{21}$	$sA_2eB_1sA_1eB_2sA_2eB_3w_{12}w_{21}w_{23}$	$sA_2eB_1sA_1eB_3sA_1eB_2w_{12}w_{13}w_{21}$
$sA_2eB_1sA_1eB_3sA_2eB_2w_{13}w_{21}w_{22}$	$sA_2eB_1sA_2eB_2sA_1eB_3w_{13}w_{21}w_{22}$	$sA_2eB_1sA_2eB_3sA_1eB_2w_{12}w_{21}w_{23}$
$sA_2eB_2sA_1eB_1sA_1eB_3w_{11}w_{13}w_{22}$	$sA_2eB_2sA_1eB_1sA_2eB_3w_{11}w_{22}w_{23}$	$sA_2eB_2sA_1eB_3sA_1eB_1w_{11}w_{13}w_{22}$
$sA_2eB_2sA_1eB_3sA_2eB_1w_{13}w_{21}w_{22}$	$sA_2eB_2sA_2eB_1sA_1eB_3w_{13}w_{21}w_{22}$	$sA_2eB_2sA_2eB_3sA_1eB_1w_{11}w_{22}w_{23}$
$sA_2eB_3sA_1eB_1sA_1eB_2w_{11}w_{12}w_{23}$	$sA_2eB_3sA_1eB_1sA_2eB_2w_{11}w_{22}w_{23}$	$sA_2eB_3sA_1eB_2sA_1eB_1w_{11}w_{12}w_{23}$
$sA_2eB_3sA_1eB_2sA_2eB_1w_{12}w_{21}w_{23}$	$sA_2eB_3sA_2eB_1sA_1eB_2w_{12}w_{21}w_{23}$	$sA_2eB_3sA_2eB_2sA_1eB_1w_{11}w_{22}w_{23}$

Table 7. DNA Sequences symbols after Step (5) for Table 2.

3′–5′ DNA Sequence	**3′–5′ DNA Sequence**	**3′–5′ DNA Sequence**
$sA_1eB_3sA_2eB_1sA_2eB_2w_{13}w_{21}w_{22}$	$sA_1eB_3sA_2eB_2sA_2eB_1w_{13}w_{21}w_{22}$	$sA_2eB_1sA_1eB_3sA_2eB_2w_{13}w_{21}w_{22}$
$sA_2eB_1sA_2eB_2sA_1eB_3w_{13}w_{21}w_{22}$	$sA_2eB_2sA_1eB_3sA_2eB_1w_{13}w_{21}w_{22}$	$sA_2eB_2sA_2eB_1sA_1eB_3w_{13}w_{21}w_{22}$

2.4. The Complexity and Feasibility of the Proposed DNA Algorithm

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

Theorem 1. *The solutions of the unbalanced assignment problem with n jobs and m individuals can be obtained by the above DNA operations.*

Proof. We first get all combinations of the *n* jobs assignment in the data pool after the first step. Since the unbalanced assignment problem requires every job be assigned, we discard DNA strands without some jobs information at step (2). Additionally, each individual should get at least one job in the problem, thus we abandon DNA strands without some jobs information at step (3). Simultaneity is required in order to find the minimum solution, thus we append the corresponding cost strands at the end of previous strands at step (4). The shortest stands in the tube T_6 mean the solutions to the unbalanced assignment problem, and we can "read" the answer at the last step.

Theorem 2. *The solutions of the unbalanced assignment problem with n jobs and m individuals can be figured out in O(mn) time steps using DNA molecules computation.*

Proof. We get that the complexity of every biological operation is in $O(1)$ time [14], the manipulation of the total algorithm can be entirely finished in a finite time range, such as step (1), (5) in $O(1)$ time, step (2) in $O(n)$, step (3) in $O(m)$, and step (4) in $O(mn)$. The time complexity T of the total algorithm is as follows:

$T(\text{Step (1)}) = O(1)$;
$T(\text{Step (2)}) = O(n)$;
$T(\text{Step (3)}) = O(m)$;
$T(\text{Step (4)}) = O(mn)$;
$T(\text{Step (4)}) = O(1)$;
$T = T(\text{Step (1)}) + T(\text{Step (2)}) + T(\text{Step (3)}) + T(\text{Step (4)}) + T(\text{Step (5)})$;
$= O(1) + O(n) + O(m) + O(mn) + O(1)$;
$= O(mn)$.

Theorem 3. *The solutions strands of the unbalanced assignment problem with n jobs and m individuals can be found in the finite length range.*

Proof. After the second step, we pick up the DNA strands for all possible assignment choices with n jobs and m individuals. Here the single strands in tube T_2 at step (3) can be described:

$$sA_{i_1}eB_{j_1}sA_{i_2}eB_{j_2}\cdots sA_{i_p}eB_{j_p}\cdots sA_{i_n}eB_{j_n}\ i_p \in \{1,2,\cdots,m\},\ j_p \in \{1,2,\cdots,n\}$$

In the beginning we reasonably design the length of s, e, A_k, B_k, for

$$\|s\| = \|A_k\| = \|B_k\| = \|e\| = t\ mer$$

In order to choose the minimum cost assignment, we append the cost strands w_{ij} at the end of the previous strands denoting j-job to i-individual at (4) step. And we let $\|w_{ij}\| = c_{ij}\ mer$ and $\max \|w_{ij}\| = y\ mer$. Then T_2 can be described:

$$sA_{i_1}eB_{j_1}sA_{i_2}eB_{j_2}\cdots sA_{i_p}eB_{j_p}\cdots sA_{i_n}eB_{j_n}w_{i_1j_1}w_{i_2j_2}\cdots w_{i_pj_p}\cdots w_{i_nj_n}$$

So the length range of DNA strands in tube T_2 is:

$$\begin{aligned}\|S\| &= \|s\| + \|A_{i_1}\| + \|e\| + \|B_{j_1}\| + \|s\| + \|A_{i_2}\| + \|e\| + \|B_{j_2}\| + \cdots \\ &\quad + \|s\| + \|A_{i_n}\| + \|e\| + \|B_{j_n}\| + \|w_{i_1j_1}\| + \|w_{i_2j_2}\| + \cdots \|w_{i_nj_n}\| \\ &= \sum_{i=1}^{n}\|s\| + \sum_{k=1}^{n}\|A_{i_k}\| + \sum_{i=1}^{n}\|e\| + \sum_{k=1}^{n}\|B_{j_k}\| + \sum_{k=1}^{n}\|w_{i_kj_k}\| \\ &= 4nt + \sum_{k=1}^{n}\|w_{i_kj_k}\|\end{aligned}$$

$\because\ 0 \le w_{i_kj_k} \le y$

$\therefore\ 4tn \le \|T_2\| \le (4t + y)n$

The length of strands in T_2 tube must be between $4tn$ and $(4t + y)n$. So we can get the solution in step (4) in the appropriate length range.

3. Experimental Section

It is shown that errors in the separation of the library strands are errors in the computation [26–32]. This means that it needs a lower rate of primary errors of hybridization in the computation. DNA sequences should be generated to ensure library strands have little secondary structure that might inhibit intended probe-library hybridization. The design must also exclude DNA sequences that might encourage unintended probe-library hybridization. In order to realize the goals, we need to follow some DNA sequence constraints. For instance, library strands should include only A, T, and C, which has a lower secondary structure than those conditions and has higher chance to bind probes; through the restriction every library and probe sequence should have no runs of more than four A, four T, four C or four G, long homopolymer tracts which may have unusual secondary structures that inhibit the binding from probes to library strands and the melting temperatures of probe and library strand hybridization will be more similar, if they do not have any long enough homopolymer tracts. Restrictions that each probe should have four, five, or six G in its sequence are aimed to insure that intended probe-library pairings have uniform melting temperatures, and so on.

In this paper, the Adleman procedures [29] are simulated working on an electronic computer. The coded algorithms are used to generate DNA sequences to solve the unbalanced assignment problem and construct the 15-base DNA sequences for every bit of the library. For Table 1, the algorithm generates 5-base random sequences, consisting of A_k, B_k, s, e, and confirms whether the library strands satisfy above-mentioned restrictions when a new DNA sequence is gained [30]. If the restrictions are satisfied, the new DNA sequence is "greedily" taken up. If the restrictions are not satisfied, then mutations would be imported individually, in succession, into the new block until either the restrictions are satisfied and then the new DNA sequence is accepted, or a threshold for the number of mutations is reached, and the algorithm has finished and so it quits, printing the DNA sequence found so far. If the whole string of bits satisfy the restrictions, the algorithm has succeeded and these DNA sequences would be the product.

Consider the example in Table 1, the problem includes jobs $\{j_1, j_2, j_3, j_4, j_5\}$,and individuals $\{i_1, i_2, i_3\}$. Basic DNA sequences are generated by Adleman's algorithms and modified, shown in Table 8. Table 9 shows DNA sequences representing the job assignment schemes sA_ieB_j ($1 \le i \le m$, $1 \le j \le n$). Adleman's algorithms are also used to calculate the enthalpy, entropy, and free energy for binding of each probe to its corresponding region on a library strand; meanwhile, the energy used is shown in Table 10.

Table 8. Sequences chosen to represent s, e, A_k, B_k, and w_{ij} in the example for Table 1.

Bit	3′–5′ DNA Sequence	Bit	3′–5′ DNA Sequence	Bit	3′–5′ DNA Sequence	Bit	3′–5′ DNA Sequence
s	CTATC	e	AACTC	A_1	TAAAA	B_1	AATTA
A_2	CTTTT	B_2	CATTA	A_3	TTCAA	B_3	ATCTA
B_4	CAAAC	B_5	ATCCA	w_{11}	CCCAT	w_{12}	ATATA
w_{13}	TTACA	w_{14}	TACCC	w_{15}	TTCTT	w_{21}	TTTCA
w_{22}	ATAAT	w_{23}	CTACC	w_{24}	TTACA	w_{25}	CATAC
w_{31}	CCTTC	w_{32}	ACTCA	w_{33}	TCACT	w_{34}	ACCCT
w_{35}	TTAAC						

Table 9. Sequences chosen to represent the job assignment schemes sA_ieB_j ($1 \leq i \leq m$, $1 \leq j \leq n$) in the example for Table 1.

Job Assignment	3′–5′ DNA Sequence	Job Assignment	3′–5′ DNA Sequence
sA_1eB_1	CTATCTAAAAAACTCAATTA	sA_1eB_2	CTATCTAAAAAACTCCATTA
$\mathrm{sA_1eB_3}$	CTATCTAAAAAACTCATCTA	sA_1eB_4	CTATCTAAAAAACTCCAAAC
sA_1eB_5	CTATCTAAAAAACTCATCCA	sA_2eB_1	CTATCCTTTTAACTCAATTA
sA_2eB_2	CTATCCTTTTAACTCCATTA	sA_2eB_3	CTATCCTTTTAACTCATCTA
sA_2eB_4	CTATCCTTTTAACTCCAAAC	sA_2eB_5	CTATCCTTTTAACTCATCCA
sA_3eB_1	CTATCTTCAAAACTCAATTA	sA_3eB_2	CTATCTTCAAAACTCCATTA
sA_3eB_3	CTATCTTCAAAACTCATCTA	sA_3eB_4	CTATCTTCAAAACTCCAAAC
sA_3eB_5	CTATCTTCAAAACTCATCCA		

Table 10. The energies for of binding each probe to its corresponding region on a library strand.

Job Assignment	Enthalpy Energy *H*	Entropy Energy *S*	Free Energy *G*	Job Assignment	Enthalpy Energy *H*	Entropy Energy *S*	Free Energy *G*
sA_1eB_1	110.5	287.1	24.7	sA_1eB_2	101.2	256.7	22.9
sA_1eB_3	111.6	294.4	25.9	sA_1eB_4	107.4	281.3	24.9
sA_1eB_5	108.3	277.7	25.2	sA_2eB_1	104.2	271.3	24.9
sA_2eB_2	99.8	248.2	22.9	sA_2eB_3	105.4	270.3	24.8
sA_2eB_4	104.5	269.7	24.7	sA_2eB_5	97.7	243.1	23.3
sA_3eB_1	107.6	275.3	25.2	sA_3eB_2	111.2	289.5	25.8
sA_3eB_3	103.3	262.4	24.6	sA_3eB_4	114.7	292.1	26.3
$sA_3e\mathrm{B}_5$	112.1	288.5	26.1				

Our program also figures out the average and standard deviation for the enthalpy, entropy, and free energy over all probe/library strand interactions. The energy levels are shown as in Table 11. Table 12 presents the library strands and the solution $\{j_3 \rightarrow i_1, j_4 \rightarrow i_1, j_2 \rightarrow i_2, j_1 \rightarrow i_3, j_5 \rightarrow i_3\}$of the unbalanced assignment problem.

Table 11. The energies over all probe/library strand interactions.

Average	106.463	273.613	24.794
Standard Deviation	4.8018	15.3631	1.0363

Table 12. DNA sequences chosen to represent the answer of the unbalanced assignment problem.

Solutions	DNA Strands Denoting Solutions
$\{j_3 \rightarrow i_1, j_4 \rightarrow i_1, j_2 \rightarrow i_2, j_1 \rightarrow i_3, j_5 \rightarrow i_3\}$	3′-CTATCTAAAAAACTCATCTACTATCTAAAAAACTCCAAAC CTATCCTTTTAACTCCATTACTATCTTCAAAACTCAATTACTAT CTTCAAAACTCATCCATTACATACCCATAATCCTTCTTAAC-5′

4. Conclusions

In this paper, we present DNA algorithms for solving the unbalanced assignment problem based on biological operations in the Adleman-Lipton model. Due to electronic computers having obvious limits in storage, speed, intelligence, and miniaturization, the methods of DNA computation have arisen, especially for their efficient parallelism. The present algorithm has the following advantages compared with previous algorithms: firstly, the proposed algorithm actually has a lower rate of errors for hybridization because we develop a computer program to generate good DNA sequences for generating the solution space of the unbalanced assignment problem. Secondly, the proposed algorithm requires a time cost and a DNA strand length that are linearly proportional to the instance size. It can finish in $O(mn)$ the unbalanced assignment problem with n jobs and m individuals, having certain advantages comparing with existing algorithms, such as algorithm of paper [33] with $O(n\log n(m + n\log n))$ complexity. Additionally, the proposed algorithms can be easily performed in a fully automated manner in a laboratory. The full automation manner is essential not only for the speedup of computation but also for error-free computation. Meanwhile we simulated the DNA experiment to solve the unbalanced assignment problem. The ability to perform complex operations in solution might help us learn more about the nature of computation and lead to the development of better DNA-based computation, capable of solving a wide range of complex problems. We hope that, in future studies, more highly-effective DNA operations will be exploited to derive a DNA computing model with time efficiency and is complete for NP-hard problems.

Acknowledgments

The project was also supported by National Natural Science Foundation of China (No. 41201400 and No. 41201399).

Author Contributions

Zhaocai Wang designed the study, analyzed the data, and wrote the manuscript. Jun Pu carried out theoretical proof and gave the instructions for modification. Liling Cao performed most of the experiments. Jian Tan provided manuscript instructions and supervised the study.

Conflicts of Interest

The authors declare no conflict of interest.

References

1. Adleman, L.M. Molecular computation of solution to combinatorial problems. *Science* **1994**, *266*, 1021–1024.
2. Lipton, R.J. DNA solution of HARD computational problems. *Science* **1995**, *268*, 542–545.
3. Roweis, S.; Winfree, E.; Burgoyne, R.; Chelyapov, N.V.;Goodman, M.F.; Rothemund, P.W.K.; Adleman, L.M. A sticker based model for DNA computation. *J. Comput. Biol.* **1998**, *5*, 615–629.
4. Ouyang, Q.; Kaplan, P.D.; Liu, S.; Libchaber, A. DNA solution of the maximal clique problem. *Science* **1997**, *278*, 446–449.
5. Winfree, E.; Liu, F.; Wenzler, L.A.; Seeman, N.C. Design and self-assembly of two dimensional DNA crystals. *Nature* **1998**, *394*, 539–544.
6. Sakamoto, K.; Gouzu, H.; Komiya, K.; Kiga, D.; Yokoyama, S.; Yokomori, T.; Hagiya, M. Molecular computation by DNA hairpin formation. *Science* **2000**, *288*, 1223–1226.

7. Smith, L.M.; Corn, R.M.; Condon, A.E.; Lagally, M.G.; Frutos, A.G.; Liu, Q.; Thiel, A.J. A surface-based approach to DNA computation. *J. Comput. Biol.* **1998**, *5*, 255–267.
8. Li, W.X.; Xiao, D.M.; He, L. DNA ternary addition.*Appl. Math. Comput.* **2006**, *182*, 977–986.
9. Xiao, D.M.; Li, W.X.; Yu, J.; Zhang, X.D.; Zhang, Z.Z.; He, L. Procedures for a dynamical system on $\{0,1\}^n$ with DNA molecules. *BioSystems* **2006**, *84*, 207–216.
10. Wang, Z.C.; Huang, D.M.; Meng, H.J.; Tang, C.P. A new fast algorithm for solving the minimum spanning tree problem based on DNA molecules computation. *BioSystems* **2013**, *114*, 1–7.
11. Lee, J.Y.; Shin, S.Y.; Park, T.H.; Zhang, B.T. Solving traveling salesman problems with DNA molecules encoding numerical values. *BioSystems* **2004**, *78*, 39–47.
12. Wang, Z.C.; Huang, D.M.; Tan, J.; Liu, T.G.; Zhao, K.; Li, L. A parallel algorithm for solving the n-queens problem based on inspired computational model. *BioSystems* **2015**, *131*, 22–29.
13. Chang, W.L.; Lin, K.W.; Chen, J.C.; Wang, C.-C.; Lu, L.C.; Guo, M.; Ho, M. Molecular Solutions of the RSA Public-key Cryptosystem on a DNA-based Computer. *J. Supercomput.* **2012**, *61*, 642–672.
14. Chang, W.L.; Ren, T.T.; Luo, J.; Feng, M.; Guo, M. Quantum Algorithms for Biomolecular Solutions of the Satisfiability Problem on a Quantum Machine.*IEEE Trans. Nanobiosci.* **2008**, *7*, 215–222.
15. Wang, Z.C.; Tan, J.; Huang, D.M.; Ren, Y.C.; Ji, Z.W. A biological algorithm to solve the assignment problem based on DNA molecules computation. *Appl. Math. Comput.* **2014**, *244*, 183–190.
16. Han, A. An improved DNA solution to the vertex cover problem. In Proceedings of the Fourth International Conference on Natural Computation (ICNC'08), Jinan, China, 18–20 October 2008.
17. Liu, X.C.; Yang, X.F.; Li, S.L.; Ding, Y. Solving the minimum bisection problem using a biologically inspired computational model. *Theor. Comput. Sci.* **2010**, *411*, 888–896.
18. Wang, Z.C.; Zhang, Y.M.; Zhou, W.H.; Liu, H.F. Solving traveling salesman problem in the Adleman-Lipton model. *Appl. Math. Comput.* **2012**, *219*, 2267–2270.
19. Castellanos-Garzn, J.A.; Garca, C.A.; Novais, P.; Díaz, F. A visual analytics framework for cluster analysis of DNA microarray data. *Expert Syst. Appl.* **2013**, *40*, 758–774.
20. Chang, W.-L.;Ho, M.; Guo, M. Fast Parallel Molecular Algorithms for DNA-based Computation: Factoring Integers. *IEEE Trans. Nanobiosci.* **2005**, *4*, 149–163.
21. Chang, W.L.; Ren, T.T.; Feng, M. Quantum Algorithms and Mathematical Formulations of Biomolecular Solutions of the Vertex Cover Problem in the Finite-Dimensional Hilbert Space. *IEEE Trans. Nanobiosci.* **2014**, *14*, 121–128.
22. Garey, M.R.; Johnson, D.S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*; W.H. Freeman and Company: New York, NY, USA, 1979.
23. Zimmermann, K.H.; Ignatova, Z.; Israel, M.P. *DNA Computing Models*; Springer: New York, NY, USA, **2008**, 146–147.
24. Han, A.; Zhu, D.; Pan, J. DNA Solution Based on Sequence Alignment to the Minimum Spanning Tree problem. *J. Bioinform. Res. Appl.* **2008**, *2*, 188–200.
25. Yamamura, M.; Hiroto, Y.; Matoba, T. Solutions of shortest path problems by concentration control. *Lect. Notes Comput. Sci.* **2002**, *2340*, 231–240.
26. Zhang, Y.; Chu, C.H.; Chen, Y.; Zha, H.; Ji, X. Splice site prediction using support vector machines with a Bayes kernel. *Expert Syst. Appl.* **2006**, *30*, 73–81.

27. Braich, R.S.; Johnson, C.; Rothemund, P.W.K.; Hwang, D.; Chelyapov, N.; Adleman, L.M. Solution of a satisfiability problem on a gel-based DNA computer, in: Proceedings of the Sixth International Conference on DNA Computation (DNA 2000). *Lect. Notes Comput. Sci.* **2001**, *2054*, 27–42.
28. Zhang, H.Y.; Liu, X.Y. A CLIQUE algorithm using DNA computing techniques based on closed-circle DNA sequences. *Biosystems* **2011**, *105*, 73–82.
29. Darehmiraki, M. A New Solution for Maximal Clique Problem based Sticker Model. *Biosystems* **2009**, *95*, 145–149.
30. Braich, R.S.; Johnson, C.; Rothemund, P.W.K.; Chelyapov, N.; Adleman, L.M. Solution of a 20-variable 3-SAT problem on a DNA computer. *Science* **2002**, *296*, 499–502.
31. Alonso Sanches, C.A.; Soma, N.Y. A polynomial-time DNA computing solution for the Bin-Packing Problem.*Appl. Math. Comput.* **2009**, *215*, 2055–2062.
32. Ting, C.J.; Wu, K.C.; Chou, H. Particle swarm optimization algorithm for the berth allocation problem. *Expert Syst. Appl.* **2014**, *41*, 1543–1550.
33. Balachandran, V. Faster strongly polynomial algorithms for the unbalanced transportation problem and assignment problem with monge costs. *Networks* **2013**, *62*, 136–148.