A new fast algorithm for solving the minimum spanning tree problem based on DNA molecules computation

Zhaocai Wang\textsuperscript{a,}\textsuperscript{*}, Dongmei Huang\textsuperscript{a}, Huajun Meng\textsuperscript{a}, Chengpei Tang\textsuperscript{b}

\textsuperscript{a} College of Information, Shanghai Ocean University, Shanghai 201306, PR China
\textsuperscript{b} School of Engineering, Sun Yat-sen University, Guangzhou 510006, PR China

\textbf{A R T I C L E   I N F O}

\textbf{Article history:}
Received 2 February 2013
Received in revised form 23 April 2013
Accepted 8 July 2013

\textbf{Keywords:}
DNA computation
The minimum spanning tree problem
Adleman–Lipton model
NP-complete problem

\textbf{A B S T R A C T}

The minimum spanning tree (MST) problem is to find minimum edge connected subsets containing all the vertex of a given undirected graph. It is a vitally important NP-complete problem in graph theory and applied mathematics, having numerous real life applications. Moreover in previous studies, DNA molecular operations usually were used to solve NP-complete head-to-tail path search problems, rarely for NP-hard problems with multi-lateral path solutions result, such as the minimum spanning tree problem. In this paper, we present a new fast DNA algorithm for solving the MST problem using DNA molecular operations. For an undirected graph with \( n \) vertex and \( m \) edges, we reasonably design flexible length DNA strands representing the vertex and edges, take appropriate steps and get the solutions of the MST problem in proper length range and \( \mathcal{O}(3m + n) \) time complexity. We extend the application of DNA molecular operations and simultaneously simplify the complexity of the computation. Results of computer simulation experiments show that the proposed method updates some of the best known values with very short time and that the proposed method provides a better performance with solution accuracy over existing algorithms.

\textcopyright 2013 The Authors. Published by Elsevier Ireland Ltd. Open access under CC BY license.

1. Introduction

DNA computing is a newly emerging crossdisciplinary science that uses DNA molecular biotechnologies to solve conundrum problems of computer science and computational mathematics. Adleman (1994) presented that DNA molecule can be used to solve the directed Hamiltonian path problem of size \( n \) in \( \mathcal{O}(n) \) steps, and also proved the potential parallel power of DNA computation. The advantage implied that we can utilize DNA molecule 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. Lipton (1995) demonstrated that Adleman’s experiment could be used to figure out the NP-complete satisfiability (SAT) problem. In recent years, DNA computation has received considerable interest from researchers. There are three major advantages of the DNA computing: massive parallelism, enormous memory storage and very low energy consumption. Some typical DNA computing models, such as Adleman–Lipton model (Adleman, 1994; Lipton, 1995), the sticker model (Roweis et al., 1998), the restriction enzyme model (Qiyang et al., 1997), the self-assembly model (Winfree et al., 1998), the hairpin model (Sakamoto et al., 2000) and the surface-based model (Smith et al., 1998), have already been established. Based on these models, Lots of papers have occurred for designing DNA procedures and algorithms to solve various NP-complete problems (Li et al., 2006; Xiao et al., 2006; Wang et al., 2008, 2012; Lee et al., 2004; Guo et al., 2005; Chang, 2007; Chang et al., 2008, 2012; Han, 2008; Liu et al., 2010; Narayanan et al., 1998). In order to totally understand the power of biological computation, it is worthwhile to try to solve more kinds of computationally intractable problems with the aid of DNA operations. Moreover, many previous research works are about optimal path search problems or set division problems (Li et al., 2006; Xiao et al., 2006; Wang et al., 2008, 2012; Lee et al., 2004; Guo et al., 2005; Chang et al., 2008, 2012; Chang, 2007; Han, 2008; Liu et al., 2005, 2010; Narayanan et al., 1998; Carey and Johnson, 1979; Jonoskas, 1998; Zimmermann et al., 2008; Han et al., 2008; Braich et al., 2001, 2002; Zhang and Liu, 2011; Majid, 2011; Alberto et al., 2009; Bakar et al., 2008; Bondy, 1976; Yao et al., 2008; Chen and Zhang, 2000; Han and Zhu, 2006; Yamamura et al., 2002). For example, Lee et al. (2004) first designs different length’s strands representing paths values and cities, takes molecular operations to generate strands standing for all possible paths, then uses biochemical techniques, such as denaturation temperature gradient polymerase chain reaction and temperature gradient gel, to get the
optimum solutions of the traveling salesman problem. To solve the shortest path problem, Narayanan et al. (1998) respectively carries out DNA reaction to get the strands for a list of series paths, then chooses the shortest length strands as the solution through DNA biotechnologies. The previous researches have some insufficient factors. One is that the strands for the possible paths are usually very long, while too long DNA strands can lead to error-prone in annealing and separation procedures using modern biotechniques. The other is that many previous disquisitive NP-complete problems based on DNA computation are path search problems, which the optimum solutions are vertex hail-to-end paths, such as the Traveling Salesman Problem, Shortest Route Problem, Hamilton Path Problem and so on. While in solution space paths of the minimum spanning tree problem, one vertex may not just point to one vertex but maybe more than one. So expressing vertex one-to-many paths by DNA strands is an important issue towards extending the capability of DNA computing to solve many optimization problems.

The minimum spanning tree problem can be described: Given an undirected and no-loop graph $G = (V, E, C)$ with a vertex set $V = \{v_1, v_2, \ldots, v_n\}$ and edges set $E = \{e_{ij}\}_{1 \leq i, j \leq n}$. Let $|V| = n$, $|E| = m$, $c_{ij} \in C$ is the weight of edge $e_{ij}$. For $T = (V, E) (E \subseteq E)$ is a spanning tree if and only if $T$ is a connected graph with all the vertex of $G$ and $(n - 1)$ edges, and the total weight is equal to $\sum_{e_{ij} \in E} c_{ij}$.

The minimum spanning tree problem is to find a spanning tree $T$ of graph $G$ such that $T$ has the minimum weight. For instance, the undirected graph $G$ in Fig. 1 defines such a problem. It is not difficult to find that the edges subset $\{e_{12}, e_{24}, e_{25}, e_{34}\}$ is the solution to the minimum spanning tree problem for graph $G$ in Fig. 1. Garey and Johnson (1979) has shown that the minimum spanning tree problem is NP-complete. The minimum spanning tree problem is a problem of central importance in graph theory and computational Sciences and also plays an important role in parallel processing. As a result, various heuristic algorithms have been devised for the MST problem. Now it can be solved by Prim algorithm in $O(n^2)$ time and Kruskal algorithm in $O(m \log m)$ time. But with the scale expansion of graph, it is intractable to solve. Liu et al. (2005) use 3-dimensional (3D) DNA structures to represent the different vertex in the graph, design number of the hydrogen bonds in the edges strands in connection with edges weights. For melting temperature of DNA sequences can be influenced by the G/C contents (the more G/C contents in DNA sequences, the lower melting temperature), and the melting temperature influence Polymerase Chain Reaction (PCR). They design the amount G/C contents of DNA sequences related to edges weight (edges with small weights have more G/C contents). In the solution space sequences, the DNA sequences with maximum quantity sum of G/C contents is the result of minimum spanning tree problem. The method has two disadvantages: One is that it cannot be used complex graph with vertex degree more 3 for 3D DNA structures is unstable and can not easy to generate (Jonoskas, 1998). The other is that it is hard to meet the weight value strictly corresponding to the number of hydrogen bonds in the experiment strands. Taking the Table 4 in Liu et al. (2005) for example, weight values 20 ($v_{12}, v_{13}, v_{14}$) can be denoted by different number Hydrogen bonds (86 and 88), there is inevitable having small error to affect accuracy of computing. Zimmermann et al. (2008) and Han et al. (2008) attempted using the DNA strands with actual weights length to represent the edges information. But in their algorithms, they ignored that edge subsets having a loop with minimum weight sum may be improperly chosen as the optimum solution. Such as the disconnected edges subset $\{e_{12}, e_{15}, e_{25}, e_{34}\}$ in the Fig. 1 of author’s paper is inaccurate choice for it is not a spanning tree. But it is not can be distinguished in above algorithms for lacking of judgement on loop. In this paper, based on a combination of Adleman–Lipton model and the DNA molecule sticker model is introduced for figuring out solutions of the minimum spanning tree problem.

The rest of this paper is organized as follows. In Section 2, the Adleman–Lipton model is introduced in detail. Section 3 uses a DNA molecular algorithm for solving the minimum spanning tree problem. Section 4 proved DNA algorithm complexity and feasibility. In Section 5, We use computer to simulate the DNA experiment and get correct solution of the Fig. 1, furthermore, a relatively complex example of the minimum spanning tree problem was given and corresponding simulant results were described. We get conclusions in Section 6.

2. The Adleman–Lipton model

Deoxyribonucleic acid or DNA plays the role of memory in nature. DNA is the genetic material containing the whole information of an organism to be copied into the next generation of the nature. DNA-based computing, or more generally molecular computing, is a computational paradigm that uses synthetic DNA molecules as information storage media. Bio-molecular computers work at the molecular level. Because biological and mathematical operations have some similarities, DNA, the genetic material that encodes for living organisms, is stable and predictable in its reactions and can be used to encode information for mathematical systems.

The DNA is a long polymer formed by units called nucleotides that connect among themselves by four different types of molecules called bases: adenine (A), cytosine (C), guanine (G) and thymine (T). To the context of this work, a nucleotide and its corresponding base are considered as the same element. To form the DNA sequences, the nucleotides are joined among them by phosphate groups C bonds C that are asymmetric with respect of the geometry of each other, and they are referred as the 3’ and 5’ ends. The DNA double helix structure comes as a result of the annealing of complementary bases (A with T and C with G). The reverse process C melting C separate the double helix into two bases sequences. Moreover, a sequence of pieces of DNA is composed by genes, e.g., the single strands $5’TGACGTACCG3’$ and $3’GACGTACCG5’$ can form a double strand. We also call the strand $3’GACGTACCG5’$ as the complementary strand of $5’TGACGTACCG3’$ and simply denote $3’GACGTACCG5’$ by $5’TGACGTACCG3’$. The length of a single stranded DNA is the number of nucleotides comprising the single strand. Thus, if a single stranded DNA includes 15 nucleotides, it is called a 15 mer. The length of a double stranded DNA is counted in the number of base pairs. Thus, if we make a double stranded DNA from a single strand 15 mer, then the length of the double stranded DNA is 15 base pairs, also written as 15 bp.
The DNA operations proposed by Adleman (1994) and Lipton (1995) are described below. These operations will be used for figuring out solutions of the minimum spanning tree problem in this paper. 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 $T_1, T_2$, 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 chooses the shortest length strands in the tube $T_2$, the longest strands in $T_3$ and the remaining strands in $T_1$;
7. **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 strand containing $[\gamma_0 \gamma_1]$ in $T$ into different strands as follows:
   \[
   [\cdots \alpha \gamma_0 \gamma_1 \beta \gamma_0 \gamma_1 \delta \cdots] \Rightarrow [\cdots \alpha \gamma_0 \gamma_1 \beta \gamma_0 \gamma_1 \delta \cdots];
   \]
8. **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;
9. **Denaturation** ($T$): for a given test tube $T$, it dissociates each double strand in $T$ into two single strands;
10. **Ligation** ($T$): for a given test tube $T$, the operation is used to ligate together the strands in $T$;
11. **Discard** ($T$): for a given test tube $T$, it discards the tube $T$;
12. **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;
13. **Append-head** ($T, Z$): for a given test tube $T$ and a given DNA single strand $Z$ it appends $Z$ onto the head of every strand in the tube $T$;
14. **Append-tail** ($T, Z$): for a given test tube $T$ and a given DNA single strand $Z$ it appends $Z$ onto the end of every strand in the tube $T$.

Since these fourteen manipulations are implemented with a constant number of biological steps for DNA strands (Shin et al., 1999), we assume that the complexity of each manipulation is in $O(1)$ time steps.

3. DNA algorithm for the minimum spanning tree problem

For a given undirected graph $G = (V, E)$, $V = \{v_k \mid k = 1, 2, \ldots, n\}$ is vertex set, $E = \{e_{ij} \mid 1 \leq i, j \leq n\}$ is edges set and $|E| = m$. Some vertex $v_i$ and $v_j$ can be connected by the edge $e_{ij}$ in graph $G$ with the positive integer weight $c_{ij}$. At the same time, the graph processed in this paper has no self-loops.

In the following, the symbols $s, e, A_k (k = 1, 2, \ldots, n)$ denote distinct DNA single strands with same length, say $t mer$ ($t$ is a positive integer). 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 (Zimmermann et al., 2008). Meanwhile we use the symbols $w_{ij}$, $w_{ij}^{-1}$ to denote the edge $e_{ij}$ and $||w_{ij}|| = c_{ij}$. Then in the below operations, we use the distinct DNA single strands symbols $sA_k e$, $sA_k e (1 \leq i, j \leq n)$ to denote the edge $e_{ij}$ without weight information. Simultaneity the symbol $s, e$ is the signal of different edges division. Let

\[ R = \{\varnothing\} \]

For a graph with $n$ vertex and $m$ edges, every possible subset of the edges subset $E$ can be expressed by a list of DNA strands. DNA strands with $sA_k e$ or $sA_k e$ represent the edge $e_{ij}$ in the subset, and without $sA_k e$ and $sA_k e$ represent the edge $e_{ij}$ out of the subset. For example in Fig. 1, the edges subset $\{e_{1,2}, e_{2,4}, e_{3,4}, e_{4,5}\}$ can be expressed by the DNA strands

\[ (sA_1 A_2 e sA_3 e A_4 A_5 e A_2 e A_4 e A_3 e A_5 e) \]

In this way, we transform all possible edges subsets of $E$ for different DNA strands. We call this the data pool.

1. We choose all possible edges subsets of graph $G$. For $k = 1$ to $k = |E| = m$
   \[ (1-1) \text{Copy}(R, T_1) \]
   \[ (1-2) \text{Append} - \text{tail}(T_1, sA_k e) \]
   \[ (1-3) \text{Merge}(R, T_1) \]
   \[ (1-4) \text{Discard}(T_1) \]

   End for After the above steps of manipulations, the single strands in tube $R$ will encode all possible subsets of edges. For example, for the graph in Fig. 1, we have single strands:

\[ (sA_1 A_2 e sA_3 e A_4 e A_5 e A_2 e A_4 e A_3 e A_5 e) \in R \]

which denote the subset of edges $\{e_{1,2}, e_{1,5}, e_{2,3}, e_{2,4}, e_{4,5}\}$. The number of edges in the subset is $m$, so this step operation can be finished in $O(m)$ time steps since each manipulation above works in $O(1)$ steps.

2. Each single strand in tube $R$ denotes one possible edges subset. The minimum spanning tree problem is firstly required the solutions that all vertex of the graph should be included in the edges subset. So we should check all the edges subsets whether to satisfy the above condition. If $e_k \in E$ in graph, we should discard the strands which don’t contain the symbol $A_k$. For example in Fig. 1, the single strands

\[ (sA_1 A_2 e sA_4 e A_2 e A_3 e A_2 A_4 e A_3 e A_4 e A_5 e) \in R \]

(representing the subset of edges $\{e_{1,2}, e_{1,5}, e_{2,3}, e_{2,5}\}$) should be discarded for not including the vertex $v_4$ in graph $G$. We choose all possible subset strands as below:

For $k = 1$ to $k = n$

\[ (2-1) \text{Separation}(R, A_k, T_2) \]
\[ (2-2) \text{Discard}(R) \]
\[ (2-3) \text{Copy}(T_3, R) \]
\[ (2-4) \text{Discard}(T_3) \]

End for After the above operations, the single strands in tube $R$ are edges subsets containing all the vertex of graph. Meanwhile we use one “For” clauses, thus this operation can be finished in $O(n)$ time steps since each single manipulation above works in $O(1)$ steps.

3. The solution to minimum spanning tree problem must be a edge-connected subset. So the solution to MST problem is at least having $(n - 1)$ edges for a graph with $n$ vertex. Meanwhile it must have circuit if the number of edges in the subset is more than $(n - 1)$. Therefore the solution of the minimum spanning tree problem is one and only having $(n - 1)$ edges in the subset, that cannot be the optimum solution to the problem. We
should discard the inappropriate strands. For example, for the graph in Fig. 1, the singled strands
\[(s_{A_1}A_2es_{A_2}A_1es_{A_2}A_3es_{A_4}A_5es_{A_4}A_6e) \in R\]
in R represent containing the 5 edges \(\{e_{1,2}, e_{2,3}, e_{2,5}, e_{3,4}, e_{4,5}\}\) should be discarded for having a circuit loop \(v_2 \rightarrow v_3 \rightarrow v_4 \rightarrow v_5 \rightarrow v_2\). Owing to at first we let \(|s| = |e| = |A_0| = t\) mer, then the DNA strands length with \((n - 1)\) edges is \((4n - 4)t\). This is done by the following manipulations:
1. Selection \(R, (4n - 4)t, T_3\);
2. Discard \(R\);
3. Copy \(T_2, R\).
In the above operation, this operation can be finished in \(O(1)\) time steps since each single manipulation above works in \(O(1)\) steps.

(4) Through above manipulations, most of the strands denote spanning trees. But there still exits a class non-proper DNA strands in tube \(R\) which contain all the vertex symbol with \((n - 1)\) edges. For example, for the graph in Fig. 1, the singled strands
\[(s_{A_1}A_2es_{A_2}A_1es_{A_2}A_3es_{A_4}A_5e) \in R\]
represent edges subset \(\{e_{1,2}, e_{1,5}, e_{2,5}, e_{3,4}\}\) should be discarded for that the edges subset is not connected and having a loop path. Nevertheless solutions to minimum spanning tree problem cannot have a edges-loop and must be a edges-connected graph. We should choose the strands containing edges-loop of graph \(G\) to discard. In previous DNA computing study (Liu et al., 2005; Jonoskas, 1998), they overlooked the existence of this situation. If we find a loop in the graph, the longest strands in the loop is not be selected as solution to the minimum spanning tree problem. So we automatically generate all possible loop path with weight−length DNA strands. We let
\[P = s_{A_1}w_{12}A_2e, s_{A_2}w_{21}A_1e, s_{A_1}w_{15}A_5e, \ldots,\]
\[x s_{A_4}w_{25}A_5e, s_{A_5}w_{54}A_4e,\]
\[Q = [s_{A_1}w_{12}A_2e, s_{A_2}w_{21}A_1e, s_{A_1}w_{15}A_5e, \ldots, s_{A_4}w_{25}A_5e, s_{A_5}w_{54}A_4e]\]
For \(k = 1\) to \(k = |E| = m\)
1. Merge \(P, Q\);
2. Annealing \(P\);
3. Ligation \(P\);
4. Denaturation \(P\);
5. Separation \(P, (3 - s_{A_i}w_{ij}), T_4\);
6. Separation \(T_3, (A_i e - 5'), T_5\);
7. Separation \(T_2, (w_{ij}A_i), T_6\);
8. Discard \(T_2\);
9. Discard \(T_3\);
10. Discard \(T_2, T_5, T_3\);
11. Discard \(T_6, w_{ij}A_i, T_2\);
12. Sort \(T_2, T_3, A_3, T_5, T_1\);
13. Separation \(R, (A_i A_j T_{12} = e_{ij}), T_{12}\);
14. Discard \(T_2\);
15. Discard \(T_3\);
16. Discard \(T_1\).
End for in the above operation, we use one “For” clauses, we discard non-proper strands at most \(m\), thus this operation can be finished less in \(O(m)\) time steps since each single manipulation above works in \(O(1)\) steps.

(5) The minimum spanning tree set problem should be a smallest weight edges subset which satisfy the above condition. So we append the weight value \(w_{ij}\) at the end of previous strands containing the edge \(e_{ij}\). For example, for the graph in Fig. 1, the singled strands
\[(s_{A_1}A_2es_{A_2}A_1es_{A_2}A_3es_{A_4}A_5e) \in R\]
represent edges subset \(\{e_{1,2}, e_{1,5}, e_{2,5}, e_{3,4}\}\), we append strands \(w_{1,2}, w_{1,5}, w_{2,3}, w_{4,5}\) at the above-mentioned strands to
\[(s_{A_1}A_2es_{A_2}A_1es_{A_2}A_3es_{A_4}A_5ew_{1,2}w_{1,5}w_{2,3}w_{4,5})\]
This is done by the following manipulations:
1. Separation \(R, A_i A_j, T_{12}\);
2. Append − tail \(T_{13}, w_{1,2}\);
3. Merge \(R, T_{13}\);
4. Discard \(T_{13}\);
End for in the above operation, we use one “For” clauses, thus this operation can be finished in less \(O(m)\) time steps since each single manipulation above works in \(O(1)\) step.

(6) We take out those singled strands in \(R\) with shortest length, which give the solutions to minimum spanning tree problem. For example, for the graph in Fig. 1, those singled strands in \(R\) with shortest length are
\[(s_{A_1}A_2es_{A_2}A_4es_{A_5}A_3es_{A_4}ew_{1,2}w_{2,3})\]
Therefore, solutions to minimum spanning tree problem for the graph in Fig. 1 are \(\{e_{1,2}, e_{2,4}, e_{2,5}, e_{3,4}\}\) with the weight sum 8.

1. Sort \(R, T_{14}, T_{15}\);
2. Read \(T_{14}\);
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 minimum spanning tree problem.

4. The complexity and feasibility of the proposed DNA algorithm
The following theorems tell that the algorithm proposed above really can get solutions of the minimum spanning tree problem in \(O(3m + n)\) steps using DNA molecules.

Theorem 1. The solutions of minimum spanning tree problems for a graph with \(n\) vertex and \(m\) edges can be obtained by the above DNA operations.

Proof. We first get all combinations of the edges in the data pool after the first step. Because the spanning tree should be traversal all the vertex of graph \(G\), we discard strands without some vertex information of graph at step (2). Simultaneous the spanning tree should have \((n - 1)\) edges in the subset, we select the satisfactory strands at step (3). Furthermore we find false spanning tree strands at step (4) and use basic biological operations to remove illegal solution strands. In order to find the minimum solution, we append the edge weight strands at the end of previous strands at step (5). The shortest strands in the pool \(R\) means the solution to minimum spanning tree problem, and we can “read” the answer at the final step.

Theorem 2. The solutions of minimum spanning tree problems for a graph with \(n\) vertex and \(m\) edges can be figured out in \(O(3m + n)\) time steps using DNA molecules. Proof. The manipulates of algorithm can be entirely finished in finite operations. Such as step (3), (6) in \(O(1)\) time, step (1), (4), (5) in \(O(m)\) time, Simultaneity step (2) in \(O(n)\) time. In conclusion, We
can get the solution of minimum spanning tree problems with $n$ vertex and $m$ edges in $O(3m+n)$ time complexity.

**Theorem 3.** The solutions strands of minimum spanning tree problems for a graph with $n$ vertex and $m$ edges can be founded in finite length range.

**Proof.** After the operations of four step, we discard the non-proper DNA strands for the minimum spanning tree problem. The singled strands in tube $R$ denote all possible spanning tree this moment. Then strands can be described:

$$sA_{i_1}A_{j_1}esA_{i_2}A_{j_2}...esA_{i_{n-1}}A_{j_{n-1}}e \quad 1 \leq i_1,j_1 \leq n$$

In the beginning we reasonably design the length of $s,e,A_k$. For

$$||S|| = ||A_k|| = ||e|| = t \quad \text{mer}$$

In order to choose the minimum spanning tree, we append the weight strand $w_{ij}$ at the end of previous strands with the edge $e_{ij}$ information. And we let $||w_{ij}|| = c_{ij} \quad \text{mer}$ and $\max ||w_{ij}|| = m \quad \text{mer}$. Then $R$ can be described:

$$sA_{i_1}A_{j_1}esA_{i_2}A_{j_2}...esA_{i_{n-1}}A_{j_{n-1}}ew_{i_1j_1}w_{j_1j_2}...w_{i_{n-1}j_{n-1}}$$

So the length range of DNA strands in tube $R$ is:

$$||S|| = ||S|| + ||A_{i_1}|| + ||A_{j_1}|| + ||e|| + ||S|| + ||A_{i_2}|| + ||A_{j_2}|| + ||e|| + \cdots + ||S|| + ||A_{i_{n-1}}|| + ||A_{j_{n-1}}|| + ||e|| + ||w_{i_1j_1}|| + ||w_{j_1j_2}|| + \cdots + ||w_{i_{n-1}j_{n-1}}||$$

$$= \sum_{i=1}^{n-1} ||S|| + \sum_{k=1}^{n-1} ||A_{i_k}|| + \sum_{l=1}^{n-1} ||A_{j_l}|| + \sum_{k,l=1}^{n-1} ||e|| + \sum_{k,l=1}^{n-1} ||w_{k,l}||$$

$$= (4n-4)t + \sum_{k,l=1}^{n-1} ||w_{k,l}||$$

Hence, $0 \leq ||w_{k,l}|| \leq m$

Hence, $(4n-4)t \leq ||R|| \leq (4n-4)t + m(n-1)$

So the length of strands in $R$ tube must be between $(4n-4)t$ and $(4n-4)t + m(n-1)$. Accordingly we can get the solution at step (6) in appropriate length range.

5. Experimental results of simulated DNA computing

5.1. Simple example of the minimum spanning tree problem

DNA-based computing counts on the biochemical operations of DNA molecules and may cause error when applying these biochemical operations. So sequence design is an important issue to make DNA-based computing more reliable. To have a better performance in hybridization reactions, we adapt the sequence design from Braich et al. (2001), such as Library sequences contain only Ts, and Cs; No probe sequence has a run of more than 7 matches with any 8 base alignment of any library sequence; and so on.

In this paper, We use BioPython, a python tool for computational molecular biology, as our developing platform for generating good DNA sequences which are suitable for executing our algorithms on laboratory. The Braich’s program (Braich et al., 2001) and other simulations are running on a Windows XP machine, with an Intel Core-2 XP CPU and 4-GB main memory, and the compiler is Visual C++. The coded program is used to generate DNA sequences to solve the minimum spanning tree Problem and to construct the DNA sequences for every bit of the library. For the graph in Fig. 1, the program generates 5-base random sequences, consisting of $s,e,A_k$ and whether the library strands satisfy the above constraints when the new DNA sequences are added (Braich et al., 2001). If the generated DNA sequence fails to pass any of the constraints, the program will regenerated a new DNA sequence. If the constraints are satisfied, the new DNA sequences are accepted. If all the DNA strands satisfy the constraints, the program has then succeeded and these sequences would be the outputs.

Consider the graph in Fig. 1. The graph includes five vertex: $v_1$, $v_2$, $v_3$, $v_4$ and $v_5$. DNA vertex sequences generated by the Braich program modified were shown in Table 1 and the edges sequence in Table 2. Braich’s program is also used to calculate the enthalpy, entropy, and free energy for binding of each probe to its corresponding region on a library strand, while the energy used is shown in Table 3.

Our program also figured out the average and standard deviation for the enthalpy, entropy, and free energy over all probe/library strand interaction. The energy levels are shown as in Table 4 and Table 5.

| Table 1 | Sequences chosen to represent $s,e$ and $A_k$ in the example for Fig. 1. |
|---------|--------------------------------------------------|
| Bit     | $3’-5’$DNA sequence | Bit | $3’-5’$DNA sequence |
| $s$     | TTCTT | $e$ | TATCC |
| $A_1$   | CACTC | $A_2$ | AACC |
| $A_2$   | CTTAC | $A_3$ | AACC |
| $A_3$   | ATATT | $w_{2,2}$ | CTAAT |
| $w_{2,2}$ | TAAT | $w_{2,3}$ | TATCA |
| $w_{2,3}$ | TACTC | $w_{2,5}$ | TACA |
| $w_{2,5}$ | CACT | $w_{4,5}$ | TCAC |

| Table 2 | Sequences chosen to represent the edges $sA_k se$ and $sA_k se$ in the example for Fig. 1. |
|---------|--------------------------------------------------|
| $e_{11}$ | 3’-5’ DNA sequence |
| $sA_k se$ | TTCTTCACTCACTATCC |
| $sA_k se$ | TTCTTCACTCACTATCC |
| $sA_k se$ | TTCTTCACTCACTATCC |
| $sA_k se$ | TTCTTCACTCACTATCC |
| $sA_k se$ | TTCTTCACTCACTATCC |
| $sA_k se$ | TTCTTCACTCACTATCC |
| $sA_k se$ | TTCTTCACTCACTATCC |
| $sA_k se$ | TTCTTCACTCACTATCC |
| $sA_k se$ | TTCTTCACTCACTATCC |
| $sA_k se$ | TTCTTCACTCACTATCC |

| Table 3 | The energies for of binding each probe to its corresponding region on a library strand. |
|---------|--------------------------------------------------|
| Vertex  | Enthalpy energy $H$ | Entropy energy $S$ | Free energy $G$ |
| $sA_k se$ | 1 | 2 | 3 | 4 |
| $sA_k se$ | 2 | 3 | 4 | 5 |
| $sA_k se$ | 3 | 4 | 5 | 6 |
| $sA_k se$ | 4 | 5 | 6 | 7 |
| $sA_k se$ | 5 | 6 | 7 | 8 |

| Table 4 | The energies over all probe/library strand interactions. |
|---------|--------------------------------------------------|
| Enthalpy energy $H$ | 106.279 |
| Entropy energy $S$ | 274.057 |
| Free energy $G$ | 24.143 |

| Table 5 | Average and standard deviation for the enthalpy, entropy, and free energy over all probe/library strand interaction. |
|---------|--------------------------------------------------|
| Average | Standard deviation |
| 106.279 | 4.70246 |
| 274.057 | 13.84603 |
| 24.143 | 1.77793 |
Table 5
DNA sequences chosen to represent the answer of the minimum spanning tree problem.

| \( e_{1,1}, e_{2,4}, e_{2,5}, e_{3,6} \) | \( 3',-TTCTCCTACCCTACCTCTCCCTTA \\ CCACTTCCCTACCTCTCCCTACCCATAAA \\ TTATCCCTTCTCTACCAACTCCCTATCC-5' |

Table 6
Sequences chosen to represent \( s \) and \( A_k \) \((k = 1, 2, \ldots, n)\) in the example for Fig. 2.

| Bit | 3'−5 DNA sequence | Bit | 3'−5 DNA sequence |
|-----|--------------------|-----|--------------------|
| s   | ATTTCC             | e   | TCAC               |
| A1  | TCCTT              | A2  | ATCC               |
| A3  | CACAT              | A4  | TCAC               |
| A5  | ACTCA              | A6  | CTCA               |
| A7  | TCATA              | A8  | ACTTA              |
| w1,2| TACAC              | w1,6| ACAT               |
| w1,6| TCCAT              | w2,3| ACTTA              |
| w2,6| TCCTCT             | w3,6| ACTT               |
| w3,5| TTCTC              | w4,5| TACT               |
| w4,7| ATACAA             | w5,6| ACTAA              |

Table 7
Sequences chosen to represent the edges \( sA_k e \) and \( sA_k e \) in the example for Fig. 2.

| e_i  | 3'−5 DNA sequence |
|------|--------------------|
| sA1 e| ATTTCTCCCCCTCTTCAAC |
| sA3 e| ATTTCTCCCCCTCTTCAAC |
| sA1 e| ATTTCTCCCCCTCTTCAAC |
| sA2 e| ATTTCTCCCCCTCTTCAAC |
| sA3 e| ATTTCTCCCCCTCTTCAAC |
| sA1 e| ATTTCTCCCCCTCTTCAAC |
| sA2 e| ATTTCTCCCCCTCTTCAAC |
| sA3 e| ATTTCTCCCCCTCTTCAAC |
| sA4 e| ATTTCTCCCCCTCTTCAAC |
| sA5 e| ATTTCTCCCCCTCTTCAAC |
| sA6 e| ATTTCTCCCCCTCTTCAAC |
| sA7 e| ATTTCTCCCCCTCTTCAAC |
| sA8 e| ATTTCTCCCCCTCTTCAAC |
| sA9 e| ATTTCTCCCCCTCTTCAAC |
| sA10 e| ATTTCTCCCCCTCTTCAAC |
| sA11 e| ATTTCTCCCCCTCTTCAAC |
| sA12 e| ATTTCTCCCCCTCTTCAAC |
| sA13 e| ATTTCTCCCCCTCTTCAAC |
| sA14 e| ATTTCTCCCCCTCTTCAAC |
| sA15 e| ATTTCTCCCCCTCTTCAAC |
| sA16 e| ATTTCTCCCCCTCTTCAAC |
| sA17 e| ATTTCTCCCCCTCTTCAAC |
| sA18 e| ATTTCTCCCCCTCTTCAAC |
| sA19 e| ATTTCTCCCCCTCTTCAAC |
| sA20 e| ATTTCTCCCCCTCTTCAAC |

Table 8
The energies for of binding each probe to its corresponding region on a library strand.

| Vertex | Enthalpy energy H | Entropy energy S | Free energy G |
|--------|-------------------|------------------|---------------|
| sA1 e  | 106.2             | 280.5            | 24.1          |
| sA2 e  | 98.6              | 251.9            | 23.6          |
| sA3 e  | 109.5             | 286.2            | 24.6          |
| sA4 e  | 102.3             | 260.8            | 24.2          |
| sA5 e  | 105.3             | 277.1            | 24.1          |
| sA6 e  | 102.6             | 272.3            | 23.4          |
| sA7 e  | 103.8             | 273.1            | 23.9          |
| sA8 e  | 113.3             | 295.8            | 25.9          |
| sA9 e  | 104.4             | 266.9            | 24.3          |
| sA10 e | 103.7             | 266.1            | 24            |
| sA11 e | 115.1             | 299.7            | 25.3          |
| sA12 e | 111.5             | 292.3            | 24.9          |
| sA13 e | 101.1             | 260.8            | 23.7          |
| sA14 e | 107.5             | 277.1            | 24.4          |
| sA15 e | 104.5             | 272.5            | 23.6          |
| sA16 e | 105.7             | 275.8            | 24.3          |
| sA17 e | 96.9              | 240.9            | 23.1          |
| sA18 e | 108.7             | 276.5            | 24.9          |
| sA19 e | 103.5             | 269.4            | 23.8          |
| sA20 e | 112               | 291.4            | 24.7          |
| sA21 e | 111.4             | 290.1            | 24.6          |
| sA22 e | 99.2              | 256.1            | 23.8          |
| sA23 e | 104.6             | 279.5            | 24.2          |
| sA24 e | 102.1             | 259.4            | 23.6          |

Table 9
The energies over all probe/library strand interactions.

|                        | Enthalpy energy H | Entropy energy S | Free energy G |
|------------------------|-------------------|------------------|---------------|
| Average                | 105.563           | 273.633          | 24.208        |
| Standard deviation     | 4.75832           | 14.56532         | 0.63788       |

Table 10
DNA sequences chosen to represent the answer of the minimum spanning tree problem.

| \( e_{1,2}, e_{2,4}, e_{2,5}, e_{3,6}, e_{5,7} \) | \( 3',-TTCTCCTACCCTACCTCTCTTCAAC \\ CCACTTCCCTACCTCTTCTCTCCCTACCCATAAA \\ TTATCCCTTCTCTACCAACTCCCTATCC-5' |

6. Conclusions

In this paper, we present DNA algorithms for solving the minimum spanning tree problem based on biological operations in the Adleman–Lipton model. Because electronic computers have 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 minimum
spanning tree problem. Secondly, Kruskal algorithm and Prim algorithm are considered efficient methods for the classic minimum spanning tree problems in fifty years [Bondy (1976)], respectively with $O(m \log m)$ and $O(n^2)$ time complexity. Meanwhile we find that Yao et al. (2008) proposed a polynomial time algorithm for the minimum degree spanning tree problem in directed acyclic graphs. The algorithm terminates in $O(m \log m)$ time, where $m$ and $n$ are the number of edges and vertex of the graph, respectively. In addition, Chen and Zhang (2000) also gave a $O(n^2)$ time algorithm for finding solutions of the minimum spanning tree problem. In our paper, the proposed algorithm requires a time cost that is linearly proportional to the instance size. It can finish in $O(3m+n)$ time for the minimum spanning tree problem of an undirected graph with $n$ vertex and $m$ edges, faster and with less computational complexity than other previous algorithms. Thirdly, At step (4), we automatic generate the possible edges-loop with easier construction of the solution space. Especially some algorithms ignored the instance existing edges-loop with non-connected edges to get the false conclusion [Zimmermann et al., 2008; Han et al., 2008], such as the edges subset $\{e_{12}, e_{15}, e_{25}, e_{34}\}$ in the Fig. 1. Besides 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 minimum spanning tree 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.

Acknowledgements

This work was supported by Doctor fund of Shanghai Ocean University (A-2400-12-0000351) and Shanghai Ocean university 085 Engineering. The project was also supported by CNSF (Grant number: 61272008) and the Specialized Research Fund for the Doctoral Program of Higher Education (SRFDP, 20100171120004).

References

Adleman, L.M., 1994. Molecular computation of solution to combinatorial problems. Science 266, 1021–1024.

Alonso Sanches, C.A., Soma, N.Y., 2009. A polynomial-time DNA computing solution for the Bin-Packing Problem. Appl. Math. Comput. 215, 2055–2062.

Baker, R.B.A., Watada, J., Pedrycz, W., 2008. DNA approach to solve clustering problem based on a mutual order. Biosystems 91, 1–12.

Bondy, J.A., 1976. Murty USR. Graph Theory with Applications. The Macmillan Press.

Braich, R.S., Johnson, C., Rothemund, P.W.K., Hwang, D., Chelyapov, N., Adleman, L.M., 2001. Solution of a satisfiability problem on a gel-based DNA computer. In: Proceedings of the Sixth International Conference on DNA Computation (DNA 2000). Lecture Notes in Computer Science 2054, 27–42.

Braich, R.S., Johnson, C., Rothemund, P.W.K., Chelyapov, N., Adleman, L.M., 2002. Solution of a 20-variable 3-SAT problem on a DNA computer. Science 298, 499–502.

Chang, W.L., Ren, T.T., Luo, J., et al., 2008. Quantum Algorithms for Bio-molecular Solution of the Satisfiability Problem on a Quantum Machine. IEEE Trans. Nanobiosci. 7, 215–222.

Chang, W.L., Lin, K.W., Chen, J.C., et al., 2012. Molecular Solutions of the RSA Public-key Cryptosystem on a DNA-based Computer. J. Supercomput. 61, 642–672.

Chang, W.L., 2007. Fast Parallel DNA-based Algorithms for Molecular Computation: the Set-Partition Problem. IEEE Trans. Nanobiosci. 6, 346–353.

Chen, G.T., Zhang, G.C., 2000. A constrained minimum spanning tree problem. Comput. Oper. Res. 27, 867–875.

Garey, M.R., Johnson, D.S., 1979. Computers and Intractability: A Guide to the Theory of NP-completeness. W. H. Freeman and Company.

Guo, M.Y., Chang, W.L., Ho, M., Lu, J., Cao, J.N., 2005. Is optimal solution of every NP-complete or NP-hard problem determined from its characteristic for DNA-based computing. BioSystems 80, 71–82.

Han, A.L., Zhu, D.M., 2006. A New DNA Encoding Method for Traveling Salesman Problem. Lecture Notes in Computer Science 4115, 328–335.

Han, A.L., Zhu, D.M., et al., 2008. DNA Solution Based on Sequence Alignment to the Minimum Spanning Tree Problem. J. Bioinform. Res. Appl. 2, 188–200.

Han, A., 2008. An improved DNA solution to the vertex cover problem. In: Proceedings of International Conference on Natural Computation.

Jonoskas, N., Saito, K.M., 1998. In: Pauw, G.H. (Ed.), Graph Structures in DNA Computing in computing with Molecules-T Theory and Experiments. Springer-Verlag, pp. 93–110.

Lee, J.Y., Shin, S.Y., Park, T.H., Zhang, B.T., 2004. Solving traveling salesman problems with DNA molecules encoding numerical values. BioSystems 78, 39–47.

Li, W.X., Xiao, D.M., He, L., 2006. DNA ternary addition. Appl. Math. Comput. 182, 977–989.

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

Liu, X., Li, Y., Xu, J., March 2005. Solving minimum spanning tree problem with DNA computing. J. Electron. (China) 22 (2), 112–117.

Liu, X.C., Yang, X.F., Li, S.L., Ding, Y., 2010. Solving the minimum bisection problem using a biologically inspired computational model. Theor. Comput. Sci. 411, 888–896.

Majid, Darehmiraki, 2009. A New Solution for Maximal Clique Problem based Sticker Model. BioSystems 95, 145–149.

Narayanan, A., Zorbalas, S., et al., 1998. DNA algorithms for computing shortest paths. in: J.R. Koza (Ed.), Proceedings of the Genetic Programming, 718–723.

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

Roweis, S., Winfree, E., Burgoyne, R., Chelyapov, N.V., Goodman, M.F., Rothemund, P.W.K., Adleman, L.M., 1998. A sticker based model for DNA computation. J. Comput. Biol. 5, 615–629.

Sakamoto, K., Gouzu, H., Komiya, K., Kiga, D., Yokoyama, S., Yokomori, T., Hagiya, M., 2000. Molecular computation by DNA hairpin formation. Science 288, 1223–1226.

Shin, S.Y., Zhang, B.T., Jun, S.S., et al., 1999. Solving traveling salesman problems using molecular programming. in: P.J. Angeline (Ed.), Proceedings of the Congress on Evolutionary Computation. 994–1000.

Smith, L.M., Corn, R.M., Condon, A.E., Lagally, M.G., Frutos, A.G., Liu, Q., Thiell, A.J., 1998. A surface-based approach to DNA computation. J. Comput. Biol. 5, 255–267.

Wang, X.L., Bao, Z.M., Hu, J.J., Wang, S., Zhan, A., 2008. Solving the SAT problem using a DNA computing algorithm based on ligase chain reaction. BioSystems 91, 117–125.

Wang, Z.C., Zhang, Y.M., Zhou, W.H., Liu, H.F., 2012. Solving traveling salesman problem in the Adleman-Lipton model. Appl. Math. Comput. 219, 2267–2270.

Winfree, E., Liu, F., Wenzler, L.A., Seeman, N.C., 1998. Design and self-assembly of two dimensional DNA crystals. Nature 394, 539–544.

Xiao, D.M., Li, W.X., Yu, J., Zhang, X.D., Zhang, Z.Z., He, L., 2006. Procedures for a dynamical system on (0,1)$^n$ with DNA molecules. BioSystems 84, 207–216.

Yamamura, M., Hiroto, Y., Matoba, T., 2002. Solutions of shortest path problems by concentration control. Lecture Notes Computer Science 2340, 211–240.

Yao, G.H., Zhu, D.M., Li, H.W., Ma, S.H., 2008. A polynomial algorithm to compute the minimum degree spanning trees of directed acyclic graphs with applications to the broadcast problem. Discrete Math. 308, 3951–3959.

Zhang, H.Y., Liu, X.Y., 2011. A CLIQUE algorithm using DNA computing techniques based on closed-circle DNA sequences. BioSystems 105, 73–82.

Zimmermann, K.H., Ignatova, Z., Israel, M.P., 2008. DNA Computing Models. Springer-Verlag, pp. 146–147.