MATCH
Communications in Mathematical and in Computer Chemistry

MATCH Commun. Math. Comput. Chem. **66** (2011) 347-370

ISSN 0340 - 6253

A Novel Approach for the Classical Ramsey Number Problem on DNA-Based Supercomputing

Xu Zhou [1, 2], Kenli Li[1] and Makojoa Goodman[1] Ahmed Sallam[1]
[1] *School of Computer and Communications, Hunan University, Changsha 410082*
[2] *College of Mathematics and Information Engineering, JiaXing University, JiaXing 314200*
People's Republic of China

(Received April 8, 2010)

Abstract

Ramsey number problem is a classical NP-complete problem. It takes exponential time to solve classical Ramsey number problem with traditional electronic computers. In this paper, we propose a new DNA computing algorithm based on the divide and conquer algorithm to solve the classical Ramsey number problem efficiently. The new algorithm is made up of a subspace generator, a parallel searcher, a subspace searcher and the Ramsey number generator. With the use of divide and conquer algorithm, the new DNA computing algorithm can be scaled-up to solve much greater size and harder Ramsey number problems compared to the existing algorithms. In addition, the space-efficient and error-tolerant ability can also be improved. By this approach, we have shown that DNA computing has vast potential for solving NP-complete problems.

1. Introduction

Since Dr. Leonard Adleman [1] demonstrated the possibility of using molecular biology to solve the Hamiltonian Directed Path Problem (HDPP), the DNA computing has shown a great potential of solving NP-complete problems. DNA computing makes use of DNA molecules as information storage materials and biological experiments as information processing operators [2]. DNA computing provided a massive computational parallelism that allowed us to solve hard computational problems in polynomial increasing time while a conventional Turing machine requires exponential increasing time.

It is worth noting that many algorithms have been proposed to solve different NP-complete problems. DNA computing has also been used to solve many NP problems [3-14], satisfiability problem [3,4], the set-packing problem [5], the subset product problem [6], the factoring integers problem [7], the subset-sum problem [8], the clique problem

[5,9,10], the knapsack problem [11,12], the graph isomorphism problem [13] and the dominate set problem [14]. However, most of the current DNA computing strategies are based on enumerating all the candidate solutions and these algorithms require that the size of the initial pool increases exponentially with the number of variables in the calculation so that the capacity of a DNA computer is limited. Furthermore, these enumerating algorithms make the length of the DNA strands too long to be length-efficient [15-16].

Ramsey theory is an important sub discipline of discrete mathematics. There are many interesting applications of Ramsey theory including results in number theory, algebra, geometry, topology, set theory, logic, information theory and theoretical computer science. A lot of attention has been paid in research on finding the exact value of classical Ramsey numbers but the results are still far from being satisfactory. There are only nine exact Ramsey numbers known so far [17-19]. It takes an exponential time to solve Ramsey number problem with the traditional electronic computers and it takes exponential DNA strands to solve Ramsey number with the existing DNA computing algorithms.

In this paper, we have proposed a novel DNA computing algorithm to solve the Ramsey number problem. Different from the traditional algorithm that surveys all the possible assignment sequences generated in the beginning, we make use of the solution based on the sticker-based model and the operations of Adleman-Lipton model [5-8]. The proposed algorithm is based on the divide and conquer algorithm. The proposed algorithm offers ways for speeding up the algorithm and increases the size of the Ramsey number problem without increasing the time complexity. The new method will parallelize the processes of the algorithm so that the implementation of the new algorithm can be accomplished much more quickly. Furthermore, the new method will reduce the error rate.

The rest of the paper is organized as follows: section 2 introduces the DNA computation in details. Section 3 presents the Ramsey Number Problem followed by section 4 which proposes the DNA computing algorithm to solve the Ramsey Number Problem, section 5 presents the experimental simulation results and the last part section 6 gives the discussion and conclusion of our work.

2. The improved DNA Computing Model

First, we apply the sticker-based model's solution space in our algorithm. The novel model employs only mature DNA biological operations. The basic principle operations are that the

operations of Adleman-Lipton model and *Unit* operation are selected for building this new model.

2.1 Sticker–Based Solution Space

Our algorithm is based on the solution space of sticker-based model introduced in [1, 20]. As shown in Figure 1, the model involves stickers and a memory strand. The memory strand is divided into k non-overlapping sub-strands that have m bases in a single-stranded DNA with n bases. The sticker that has m bases is complementary to one of the k sub-strands in the memory strand. During the computation process, each sub-strand is considered '1' (*on*) or '0' (*off*). If a sticker is annealed to its corresponding region on the given memory strand, then that particular region is *on* for that strand. If no sticker is annealed to a region, then that region's bit is *off*. A memory complex is defined as a memory strand where parts of the sub-strands are annealed by the matching stickers. Therefore, the computational information can be carried out in a binary format along the memory complex.

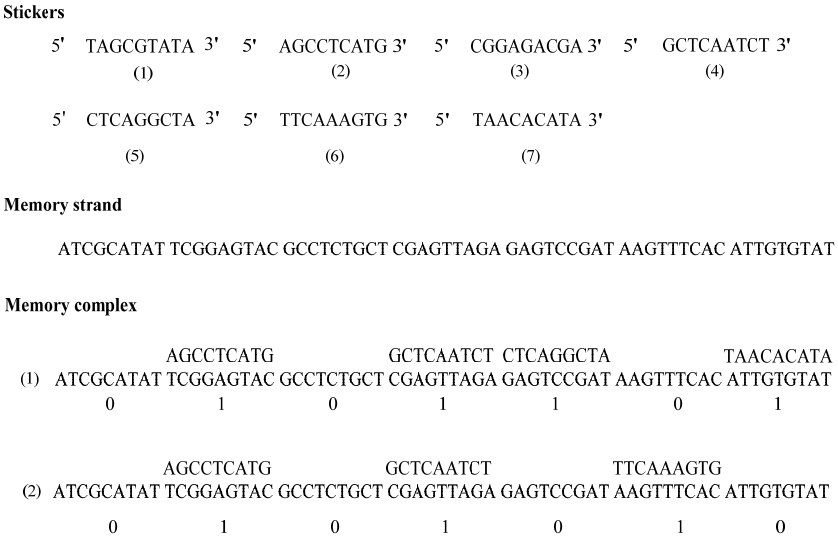

Figure 1. Illustrations of the sticker model, which are encoded 010110 and 0101010 respectively.

In the sticker-based model, the input is a test tube called initial date pool and the output is a sequence of test tubes that are called final date pool. The final date pool is read by analyzing

all the DNA strands in it.

Let us consider two integers m, n where $m \le n$, let integer $p = R(m, n)$. On the assumption that there is a graph G with p vertices, the vertex set $V(G)$ is denoted by $\{v_1, v_2, \ldots, v_p\}$ and the edge set $E(G)$ is denoted by $\{\{ v_i, v_j\}; v_i, v_j \in V(G), i \neq j\}$. In our novel algorithm, edges are represented by their binary representations using stickers. For every edge, we denoted two symbols represented by 15-base stickers to encode the information into DNA strands:

$$f(e=\{v_i, v_j\}) = x_i = \begin{cases} 0 & \text{if } e_i \text{ is not in the graph } G \\ 1 & \text{otherwise} \end{cases} \quad (1 \le i \le C_p^2) \tag{1}$$

2.2 The Operations of the DNA Computing Model

Suppose that a tube is a multi-set of DNA strands over an alphabet set $\{A, G, C, T\}$, the following biomolecular operations of the Adleman-Lipton model [5-8] will be used to construct computational space of molecules.

1. *Divide* $(T, S, (T, S)^+, (T, S)^-)$: To produce two tubes $(T, S)^+$ and $(T, S)^-$. $(T, S)^+$ consists of the DNA molecules in T which contain S as a sub-strand and $(T, S)^-$ consists of all the DNA molecules in T which do not contain S.

2. *Combine*$(T_0, T_1, T_2 \ldots T_n)$: To pour the contents of the n tubes $T_1, T_2 \ldots T_n$ into one test tube T_0. After this operation, the tubes $T_1, T_2 \ldots T_n$ will be empty.

3. *Magnify* $(T_0, T_1, T_2 \ldots T_n)$: To produce n new tubes $T_1, T_2 \ldots T_n$ which are copies of T_0 and T_0 becomes an empty tube.

4. *Attach*(T, S): To append S onto the end of every strand in T.

5. *Cast-off*(T): To discard all the DNA strands in tube T.

6. *Read-out*(T): To describe a single molecule contained in tube T.

7. *Detect-if-empty*(T): To check whether there is any DNA strand left in the tube T. If T includes at least one DNA molecule it returns 'yes,' and if T contains no DNA molecule it returns 'no'.

Besides the operations above, our new model takes a new operation: *Unit*(T_0, T_1, T_2).

8. *Unit*(T_0, T_1, T_2): To bind strands in tube T_1 with strands in tube T_2. Each strand in T_1 is ligated to every DNA strand in T_2 respectively. After the ligation, the new strands are stored in tube T_0.

The complexity of our DNA-based computing algorithm is computed by the total number of the biological operations, the total number of the test tubes used, the length of the longest DNA strands and the number of the DNA strands applied.

2.3 The Advantages of the DNA Computing Model

For the sticker-based model's solution space and the Adleman–Lipton model's operations, our new model has three advantages [1, 5-8]:

- Firstly, the proposed model has a lower rate of errors for hybridization because of having modified the program to generate good DNA sequence for constructing the solution space of stickers. Only simple and fast biological operations in the Adleman–Lipton model were employed to solve the problem.
- Secondly, the model has finished all the basic mathematical functions and the number of tubes, the longest length of DNA library strands and the number of biological operations are polynomial.
- Finally, the basic biological operations in the Adleman–Lipton model had been performed in a fully automated manner in the lab. The full automation manner is essential not only for the speedup of computation but also for error-free computation.

3. Ramsey Number Problem

Ramsey number problem is a famous optimization problem, and is also an NP-complete problem [17]. For every pair m, n of positive integers there exists at least a positive integer $k = R(m, n)$ such that for every two-coloring of the pairs of an n-element set, say $k = \{0,\ldots, k\text{-}1\}$, with colors red and blue, there exists either an m-element subset of n such that all its two-element subsets (pairs) are red (we say that there exists a red s-subset of k) or there exists a blue t-subset of k. $R(m, n)$ is called the Ramsey number of (m, n). The Ramsey number $R(m, n)$ is also defined to be the smallest number k such that, the graph with this or more nodes either contains a clique of size m or an independent set of size n [18-19].

From the definition of the Ramsey number problem, we figure out that the undirected simple graph of order $k = R(m, n)$ contains a clique of order m or an independent set of order n. Ramsey's theorem states that such a number exists for all m and n.

By symmetry, it is true that $R(m, n) = R(n, m)$. And we also have:

$$\begin{cases} R(m,n)=R(n,m) \\ R(m,2)=m \end{cases}, \quad R(m,m) \le 4R(m-2,m)+2 \tag{2}$$

From **Equation 2**, $R(1, l)=R(m, n)=1$, $R(2, n)=n$ and $R(m, 2)=m$.

Upper bounds are given by Equation 3.

$$R(m,n) \le \begin{cases} R(m-1,n)+R(m,n-1) \\ -1 \quad for\ R(m-1,n)\ and\ R(m,n-1)\ even \\ R(m-1,n)+R(m,n-1) \quad otherwise \end{cases}, R(m,n) \le C_{m+n-2}^{m-1} \tag{3}$$

In this paper, we study the Lower bounds of the Ramsey Number Problem. **Table 1** presents some results about Ramsey Number problem.

Table 1. New results of Ramsey Number Problem ($k \le 8$, $l \le 8$) [20-28]

$R(k, l)$	$k=3$	$k=4$	$k=5$	$k=6$
$l=3$	6[21]			
$l=4$	9[21]	18[21]		
$l=5$	14[21]	25[24]	43-49[24,26]	
$l=6$	18[22]	35-41[24,25]	58-87[23,27]	102-165[23,28]

4. The Novel DNA Computing Algorithm for the Ramsey Number Problem

Our novel algorithm for Ramsey Number problem is based on the new DNA computing model discussed in section 2. The detailed description of our novel algorithm is presented as follows.

4.1 The Framework of the Novel DNA Computing Algorithm

The main principle of the novel DNA algorithm for the Ramsey Number Problem is in the following:

1) Introduce the divide and conquer algorithm to our new algorithm and partition the vertex set $V(G)$ into subsets V_1, V_2,…, V_d (if p mod $n = 0$ then $d = p/n$, else $d = (p/n)+1$). For convenience, we suppose that p mod $n = 0$. Therefore, settling the Ramsey Number of R

(m, n, p) is translated into solving the Ramsey Number of R (m, n, n).

2) Produce the solution spaces for Ramsey Number problem R (m, n, n) and eliminate unfeasible solutions from the solution spaces above.

3) Based on the satisfiable solution spaces from R (m, n, n), produce the full solution spaces of the Ramsey Number problem R (m, n, p) that is equal to R (m, n) by merging the d satisfiable solution spaces of R (m, n, n).

4.2 The Divide and Conquer Algorithm

In 1974, Horowitz and Sahni [30] introduced the divide and conquer to the solution of the knapsack problem (also called the subset sum problem by some authors), and presented a two-list algorithm which can greatly reduce the time to solve the knapsack problem using enumeration algorithm. In fact the divide and conquer algorithm can solve the knapsack instance only in $O(2^{n/2})$ time and $O(2^{n/2})$ space. The divide and conquer is also introduced into our new DNA computing algorithm of Ramsey Number Problem (See **Figure 2**).

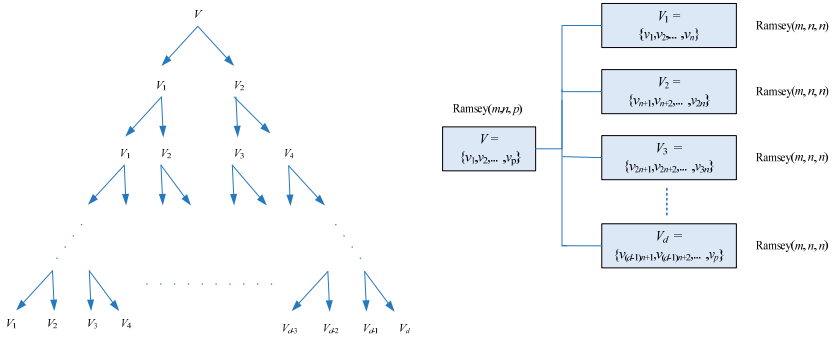

Figure 2. Partition the p elements of V into d subsets V_1,V_2,…, V_d on the assumption that $m \leq n$ and $d = p/n$.

We first partition the vertex set $V = \{v_1, v_2, \ldots, v_p\}$ into d vertices sets $V_1 = \{v_1, v_2, \ldots, v_n\}$, $V_2 = \{v_{n+1}, v_{n+2}, \ldots, v_{2n}\}$,…, and $V_d = \{v_{(d-1)n+1}, v_{(d-1)n+2}, \ldots, v_p\}$ where d is equal to p/n. Solving the Ramsey Number Problem Ramsey(m, n, p) is thentranslated into settling the Ramsey Number Problem Ramsey(m, n, n).

4.3 The Construction of a Sub–Space Generator

After partitioning the p vertices into d subsets (See **Figure 1**), the Ramsey Number problem $R(m, n)$ is translated into the Ramsey Number problem $R(m, n, n)$.

Suppose that there is k-bits binary number representing k edges of the graph with n vertices, where k is equal to C^2_n, for each bit x_a applied to represent the a^{th} edge of the graph ($1 \leq a \leq k$), there will be two different 15-base DNA sequences generated. The sequences represent '0' and '1' for x_a, respectively. For convenience, we denote x^1_a to represent the condition that x_a is equal to '1' and x^0_a to represent the condition that x_a is equal to '0'. The algorithm, Sub-Space_Generator(T_i, V_i) is designed to produce the solution space of $R(m, n, n)$.

Algorithm 1. Sub-Space_Generator(T_i, V_i)

<Input> : A tube T_i and the vertex set V_i
<Output>: The tube T_i which contains the solution space of V_i

```
1:    For j =1 to C^2_n
2:        Magnify(T_i, T_1, T_2).
3:        Attach(T_1, x^0_j).
4:        Attach(T_2, x^1_j).
5:        Combine(T_i, T_1, T_2).
6:    End For
```

Lemma 1: Algorithm Sub-Space_Generator(T_i, V_i) is applied to generate the solution space of the Ramsey Number $R(m, n, n)$.

Proof: On the assumption that T_i, T_1 and T_2 are distinct test tubes and T_1 and T_2 are empty. The Sub-Space_Generator(T_i, V_i), is implemented by the *Magnify*, *Attach* and *Combine* operations. Lines 1-6 forms a loop for generating the solution space of the vertex set V_i. There are n vertices in the set V_i. Hence, the number of edges is C^2_n. Every time, line 3 is executed to amplify tube T_i and to generate two new tubes, T_1 and T_2, which are copies of T_0, tube T_0 becomes empty. Then, line 4 is executed to append a DNA sequence (sticker), representing the value "0" for x_j, onto the end of every strand in tube T_1. Line 5 is also executed to append a DNA sequence (sticker), representing the value "1" for x_j onto the end of every strand in tube T_2. Next, line 6 is executed to pour the contents of tube T_1 and T_2 into tube T_i. This indicates that the DNA strands in tube T_0 contain the DNA sequences of $x_j = 0$ and $x_j = 1$.

After repeating the execution of lines 3-6 for C^2_n times, it finally produces tube T_i which consists of 2^k DNA sequences where $k = C^2_n$ representing 2^k possible arrays of input. Therefore, it is inferred that sticker-based solution space for 2^k ($k = C^2_n$) possible arrays of input can be constructed with the sticker.

Lemma 2: Algorithm Sub-Space_Generator(T_i, V_i), three test tubes are used to construct sticker-based solution space, an $n(n\text{-}1)/2$-bit binary number corresponds to an array of input, 2^k ($k = C^2_n$) DNA strands and it takes the following operations: $\frac{n(n-1)}{2}$ *Magnify* operations, $2n(n\text{-}1)$ *Attach* operations and $\frac{n(n-1)}{2}$ *Combine* operations.

4.4 The Construction of a Parallel Searcher

The Ramsey number $R(m, n)$ is the smallest number such that every graph with this or more nodes either contains a clique of size m or an independent set of size n. Therefore, the DNA strands which contains the clique of size m or an independent set of size n are not the solution for our Ramsey Number problem. Now, for us to eliminate the unlawful solution strands, we introduce the Parallel_Searcher algorithm as follows.

Algorithm 2. Parallel_Searcher(T_0, m, n, w)

<Input>: The tube T which contains the solution space of w vertices, the two integers in the Ramsey Number $R(m,n)$.
<Output>: The test tube T_0 containing the valid solution DNA strands.

1: **For** $i = 1$ **to** C^m_w
2: **For** $j = 1$ **to** C^2_m
3: Assume that $e_{i,j}$ is the jth edge of the ith Complete subgraph
4: *Divide*(T_0, $x^1_{i,j}$, $(T_0, x^1_{i,j})^+$, $(T_0, x^1_{i,j})^-$)
5: $T_1 = (T_0, x^1_{i,j})^+$ and $T_2 = (T_0, x^1_{i,j})^-$.
6: *Combine*(T_0 , T_0, T_1).
7: **End For**
8: *Cast-off*(T_0).
9: *Combine*(T_0 , T_0, T_2).
10: **End For**
11: **For** $i = 1$ **to** C^n_w
12: **For** $j = 1$ **to** C^2_n
13: Assume that $e_{i,j}$ is the jth element of the ith independent set
14: *Divide*(T_0, $x^1_{i,j}$, $(T_0, x^1_{i,j})^+$, $(T_0, x^1_{i,j})^-$)
15: $T_1 = (T_0, x^1_{i,j})^+$ and $T_2 = (T_0, x^1_{i,j})^-$.
16: *Combine*(T_0 , T_0, T_2).

17: **End For**
18: *Cast-off*(T_0).
19: *Combine*(T_0 , T_0, T_1).
20: **End For**

Lemma 3: Algorithm, Parallel_Searcher(T, m, n, w) is executed to search the valid solution strands from the Ramsey Number's solution space.

Proof: The Parallel_Searcher(T, m, n, w), is implemented by means of the *Divide*, *Detect-if-empty* and *Combine* operations. Line 1 is the outer loop and line 2 is the inner loop. The algorithm uses the *Divide* operation to form two test tubes: T_1 and T_2. Tube T_1 includes all the strands that have $x_{i,j} = 1$. Tube T_2 consists of all the strands that have $x_{i,j} = 0$. By executing line 6, the *Combine* operation pours the DNA strands in tube T_0 and T_1 into tube T_0 hence tube T_0 now contains the DNA strands that represent the complete subgraph. Next, on the execution of line 8, the tube T_0 is discarded. On the execution of line 9, the *Combine* operation pours the DNA strands in tube T_0 and T_2 into tube T_0. So, on the execution of lines 1-10, the complete subgraph with m vertices will be discarded. During the execution of lines 11-20, the independent set with n elements will be eliminated from the solution space. Lastly, all the legal solution strands will be contained in tube T_0.

Lemma 4: Algorithm Parallel_Searcher(T, m, n, w) uses three test tubes and takes the following operations:

1. $C^m{}_w C^2{}_m + C^n{}_w C^2{}_n = (\frac{w!}{m!(w-m)!} \times \frac{m(m-1)}{2} + \frac{w!}{n!(w-n)!} \times \frac{n(n-1)}{2})$ *Divide* operations
2. $2(C^m{}_w C^2{}_m + C^n{}_w C^2{}_n) + C^m{}_w + C^n{}_w$
   $= 2\,(\frac{w!}{m!(w-m)!} \times \frac{m(m-1)}{2} + \frac{w!}{n!(w-n)!} \times \frac{n(n-1)}{2}) + (\frac{w!}{m!(w-m)!} + \frac{w!}{n!(w-n)!})$ *Combine* operations.
3. $C^m{}_w + C^n{}_w = (\frac{w!}{m!(w-m)!} + \frac{w!}{n!(w-n)!})$ *Cast-off* operations.

4.5 The Construction of a Sub–Space Searcher

By combining **Algorithm 1** and **Algorithm 2**, we present the Sub-Space_Searcher algorithm to find the satisfiable solution space of the Ramsey Number $R(m, n, n)$. In this

algorithm, the reliable solution space of the vertex sets $V_1,...,V_d$ are produced in parallel.

Algorithm 3. Sub-Space_Searcher($\{V_1,...,V_d\}$, m, n)

<Input>: The subsets $V_1,...,V_d$ and the variables m, n of the Ramsey Number
<Output>: The tubes $T_1,...,T_d$ which contains the solution space related to the subsets $V_1,...,V_d$

```
1:    For i = 1 to d do
2:        Init_Space_Generator(V_i).
3:        Parallel_Searcher(T_i, m, n, n).
4:    End For
```

Lemma 5: Algorithm Sub-Space_Searcher ($\{V_1,...,V_d\}$, m, n) is executed to get the satisfiable solution space of the Ramsey Number $R(m, n, n)$ and the solution space related to the subsets $V_1,...,V_d$

Proof: Algorithm, Sub-Space_Searcher ($\{V_1,...,V_d\}$, m, n), is implemented based on the algorithm Init_Space_Generator(V_i) and Parallel_Searcher(T_i, m, n, n). Lines 1-4 form an outer loop. On the first execution of line 2, Init_Space_Generator(V_1) produces the solution space of the Ramsey Number $R(m, n, n)$ related to the subset V_1. Next, the execution of line 3, Parallel_Searcher(T_i, m, n, n) eliminates the unlawful DNA strands. We get the satisfiable solution space by repeating the execution of lines 2, 3.

Lemma 6: From Sub-Space_Searcher($\{V_1,...,V_d\}$, m, n) it takes d times Init_Space_Generator(V_i) and d times Parallel_Searcher(T_i, m, n, n), where $d = p/n$. As a result of Init_Space_Generator(V_i) and Parallel_Searcher(T_i, m, n, n) the Sub-Space_Searcher algorithm takes operations as follows:

1. $\frac{(n-1)pd}{2} = \frac{(n-1)p^2}{2n}$ *Magnify operations*.
2. $2(n-1)pd = \frac{2(n-1)p^2}{n}$ *Attach operations*.
3. $\frac{(n-1)pd}{2} + 2(C^m{}_nC^2{}_m + C^n{}_nC^2{}_n)d = \frac{(n-1)pd}{2} + 2(\frac{n!}{m!(n-m)!} \times \frac{m(m-1)}{2} + \frac{n(n-1)}{2})d$ *Combine operations*
4. $(C^m{}_nC^2{}_m + C^n{}_nC^2{}_n)d = (\frac{n!}{m!(n-m)!} \times \frac{m(m-1)}{2} + \frac{n(n-1)}{2})d$ *Divide operations*
5. $(C^m{}_nC^2{}_m + C^n{}_nC^2{}_n)d = (\frac{n!}{m!(n-m)!} \times \frac{m(m-1)}{2} + \frac{n(n-1)}{2})d$ *Cast-off operations*

Besides the operations above, this algorithm needs three $d=p/n$ tubes and the number of DNA strands left is $2^a - c_n^m 2^{a-b} - c_n^n (a = c_n^2 \ and \ b = c_m^2)$.

4.6 The Construction of a Ramsey Number Generator

In this section, a parallel Ramsey Number generator is proposed to combine the solution space from the algorithm Sub-Space_Searcher($\{V_1,..., V_d\}$, m, n). The process of merging the sub solution space is shown in **Figure 3**. Firstly, we merge the contents of the tubes T_1 and T_2, T_3 and T_4, ..., T_{d-1} and T_d to tubes $T_1, T_3, ..., T_{d-1}$ by the *Union* operation respectively. Then, the parallel searcher is executed to eliminate the illegal DNA strands from the tubes $T_1, T_3, ..., T_{d-1}$. After the re-labeling of tubes $T_1, T_3, ..., T_{d-1}$, we get the tubes $T_1, T_2, ..., T_{d/2}$. We repeat the above operations until all the DNA strands are stored in one tubes T_1. Finally, the test tube T_1 will contain the solution space of our problem.

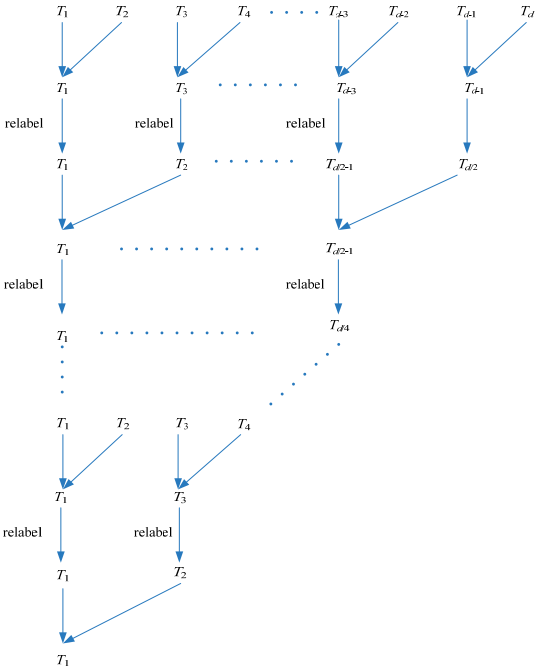

Figure 3. The process of merging the sub solution space

Algorithm 4. Ramsey_Number_Generator(T, p, m, n)

<Input>: The tube set T is $\{T_1, T_2, \ldots T_d\}$ and the parameters p, m, n of the Ramsey Number.
<Output>: The tube T_1 which contains the satisfiable solution space of the Ramsey Number.

```
1:    f = d, w = 1
2:    while f ≠ 1 do
3:        w = 2*w.
4:        If (f%2==0) then
5:            For i = 1 to f-1 Step 2 do
6:                Suppose that S_i is the number of element in V_i and S_{i+1} is the number of elements in V_{i+1}.
7:                Unit(T_i, T_i, T_{i+1}).
8:                For j =1 to S_i * S_{i+1}
9:                 Magnify(T_i, T_{i,2}, T_{i,3}).
10:                Attach(T_{i,2}, x^0_j).
11:                Attach(T_{i,3}, x^1_j).
12:                Combine(T_i, T_{i,2}, T_{i,3}).
13:                Parallel_Searcher(T_i, m, n, (w-1)n+1).
14:               End For
15:           End For
16:           Re-label all pools 1 to f/2
17:       Else
18:           For i = 1 to f-2 Step 2 do
19:               Unit(T_i, T_i, T_{i+1}).
20:               For j =1 to S_i * S_{i+1}
21:                Magnify(T_i, T_{i,2}, T_{i,3}).
22:                Attach(T_{i,2}, x^0_j).
23:                Attach(T_{i,3}, x^1_j).
24:                Combine(T_i, T_{i,2}, T_{i,3}).
25:                Parallel_Searcher(T_i, m, n, (w-1)n+1).
26:           End For
27:       End For
28:       Re-label all pools 1 to f/2
29:       Re-label the last pool to f/2 +1
30:   End If
31:   f = f/2
32:  End
33:  Parallel_Searcher(T_1, m, n, p)
```

Lemma 7. Algorithm Ramsey_Number_Generator (T, p, m, n) is executed to get the solution space of the Ramsey Number $R(m, n)$.

Proof: Ramsey_Number_Generator(T, p, m, n) is implemented by the *Magnify*, *Attach*, *Combine* ,*Unit* operations and the Parallel_Searcher(T, m, n, w) and it begins to merge the

subspace solution recursively in parallel. Lines 5-15 form an outer *for* loop on the condition that *f* is even. The algorithm merges the subspace of the Ramsey Number Problem $R(m, n, n)$ and produces the satisfiable solution space of the Ramsey Number $R(m, n, 2n)$. On the execution of line 16, we re-label the solution space from 1 to $f/2$. When *f* is odd, lines 18-24 are executed.

Lemma 8. Algorithm Ramsey_Number_Generator(*T*, *p, m, n*) takes the following operations:

1. $\lceil log_2^d \rceil$ *Unit operations*.
2. $\frac{d}{2}n^2 + \frac{d}{2^2}(2n)^2 + \frac{d}{2^3}(4n)^2 + ... + \frac{d}{2^k}(\frac{d}{2}n)^2$ $(k = log_2^d)$ *Magnify* operations.
3. $2(\frac{d}{2}n^2 + \frac{d}{2^2}(2n)^2 + \frac{d}{2^3}(4n)^2 + ... + \frac{d}{2^k}(\frac{d}{2}n)^2)$ $(k = \log_2^d)$ *Attach* operations.
4. $\frac{d}{2}n^2 + \frac{d}{2^2}(2n)^2 + \frac{d}{2^3}(4n)^2 + ... + \frac{d}{2^k}(\frac{d}{2}n)^2$ $(k = log_2^d)$ *Combine* operations.

Besides the operations above, this algorithm takes $\lceil log_2^d \rceil$ times the algorithm Parallel_Searcher(T_i, *m*, *n*, *wn*) and it requires $d+2 = p/n+2$ test tubes.

4.7 The Complete Algorithm for Ramsey Number Problem

The following DNA algorithm is applied to solve the Ramsey Number Problem $R(m, n, p)$

Algorithm 5. Ramsey_Number(*m*, *n*, *p*)

<Input>: The parameters *p*, *m*, *n* of the Ramsey Number
<Output>: The Ramsey Number $R(m, n)$

```
1:    Sub-Space_Searcher({V1,..., Vd}, m, n).
2:    Ramsey_Number_Generator(T1, p, m, n).
3:    If ( Detect-if-empty(T1) = 'yes') then
4:        Ramsey_Number(m, n, p+1).
5:    Else
6:        R (m, n) = p
7:    End If
```

Theorem 9: From the steps in Algorithm 1, the Ramsey Number problem $R(m, n)$ can be found.

Proof: On the execution of line 1, Sub-Space_ Searcher({V_1,..., V_d }, *m*, *n*), the satisfiable subsolution space of the Ramsey Number $R(m, n, n)$ is produced. The algorithm,

Ramsey_Number_Generator(T, p, m, n), is mainly used to construct a full solution space and search the solution of Ramsey Number $R(m, n)$. Line 3 detects whether there is a DNA strand in tube T_1. If it returns 'yes', line 4, the algorithm Ramsey_ Number(m, n, p+1) will be executed to find the Ramsey problem (m, n, p+1). If on the execution of line 3, it returns 'no', line 6 will be executed, and the solution of the Ramsey Number R (m, n) is equal to p.

Theorem 10: The Ramsey Number problem R (m, n) can be solved with $O(p^2)$ biological operations, $O(p/n)$ tubes , $O(2^s - C_p^m 2^{s-y} - C_p^n 2^{s-z})$ $(s = C_p^2, y = C_m^2 \text{ and } z = C_n^2)$ strands, and the longest library strand $O(p^2)$.

Proof:

1: Algorithm 5 includes seven main lines. From the algorithm, Sub-Space_ Searcher($\{V_1,...,V_d\}$, m, n), line 1, is executed to produce the satisfiable solution space for the Ramsey Number problem $R(m, n, n)$. It is obvious that it takes $(n\text{-}1)p^2/2n$ *Magnify* operations, $2(n\text{-}1)p^2/n$ *Attach operations*, $((n\text{-}1)p/2)+ 2(C^m{}_nC^2{}_m+C^n{}_nC^2{}_n)d$ *Combine operations*, $(C^m{}_nC^2{}_m+C^n{}_nC^2{}_n)d$ *Divide operations*, and $(C^m{}_nC^2{}_m+C^n{}_nC^2{}_n)d$ *Cast-off operations*. The number of the operations is:

$$
\begin{aligned}
&(\frac{(n-1)p^2}{2n})+(\frac{2(n-1)p^2}{n})+(\frac{(n-1)p}{2})\times d+2\times(\frac{n(n-1)...(n-m+1)}{2((m-2)!)}+\frac{n(n-1)}{2})\times d+(\frac{n(n-1)...(n-m+1)}{2((m-2)!)}+\frac{n(n-1)}{2})\times d\\
&+(\frac{n(n-1)...(n-m+1)}{2((m-2)!)}+\frac{n(n-1)}{2})\times d\\
&=(\frac{3(n-1)p^2}{n})+4\times(\frac{n(n-1)...(n-m+1)}{2((m-2)!)}+\frac{n(n-1)}{2})\times d\\
&=O(p^2)
\end{aligned}
$$

Line 2 produces the solution space for the Ramsey Number Problem $R(m, n, p)$ and it takes the following operations:

1. $\lceil log_2^d \rceil$ *Unit operations*,
2. $\frac{d}{2}n^2+\frac{d}{2^2}(2n)^2+\frac{d}{2^3}(4n)^2+.....+\frac{d}{2^k}(\frac{d}{2}n)^2$ $(k=log_2^d)$ *Magnify* operations,
3. $2\times(\frac{d}{2}n^2+\frac{d}{2^2}(2n)^2+\frac{d}{2^3}(4n)^2+.....+\frac{d}{2^k}(\frac{d}{2}n)^2)$ $(k = \log_2^d)$ *Attach* operations,
4. $\frac{d}{2}n^2+\frac{d}{2^2}(2n)^2+\frac{d}{2^3}(4n)^2+.....+\frac{d}{2^k}(\frac{d}{2}n)^2$ $(k=log_2^d)$ *Combine* operations.

So the total number of the operations taken in line 2 is:

$$\lceil \log_2^d \rceil + (\frac{d}{2}n^2 + \frac{d}{2^2}(2n)^2 + \frac{d}{2^3}(4n)^2 + ... + \frac{d}{2^k}(\frac{d}{2}n)^2) + 2\times(\frac{d}{2}n^2 + \frac{d}{2^2}(2n)^2 + \frac{d}{2^3}(4n)^2 + ... + \frac{d}{2^k}(\frac{d}{2}n)^2$$
$$+(\frac{d}{2}n^2 + \frac{d}{2^2}(2n)^2 + \frac{d}{2^3}(4n)^2 + ... + \frac{d}{2^k}(\frac{d}{2}n)^2))$$
$$= \lceil \log_2^d \rceil + 4\times(\frac{d}{2}n^2 + \frac{d}{2^2}(2n)^2 + \frac{d}{2^3}(4n)^2 + ... + \frac{d}{2^k}(\frac{d}{2}n)^2)$$
$$= \lceil \log_2^{p/n} \rceil + 4\times((\frac{1}{2} + 1 + 2 + ...)pn + p^2))$$
$$= O(p^2 + pn)$$

So the number of the biological operations In Algorithm 5 is $O(p^2)$.

2. In new algorithm, the illegal strands will be removed in the process of producing the solution space. Finally, the number of the DNA strands is $2^s - C_p^m 2^{s-y} - C_p^n 2^{s-z}$ which is $O(2^s - C_p^m 2^{s-y} - C_p^n 2^{s-z})$ $(s = C_p^2,\ y = C_m^2\ and\ z = C_n^2)$.

3. From **Algorithm 1** to **Algorithm 5**, the number of test tubes needed is $d = p/n$ and the number of tubes is $O(p/n)$.

4. In the Ramsey Number Problem $R(m, n, p)$, there are $C_P^2 = p(p\text{-}1)/2$ edges in the graph with p vertices. Hence, we need $p(p\text{-}1)/2$ bits to represent the $p(p\text{-}1)/2$ edges and the number of the longest DNA strands is $O(p^2)$.

5. Experimental Results by Simulated DNA Computing

For the purposes of simulating our new algorithm, reference is made to Michael at 2005[5-8] and the article [29].We consider the Ramsey Number Problem $R(3, 3) = 6$ as an example. In our example, we must find the Ramsey graph from the graph with six vertices. There are C_6^2 edges of the graph with five vertices, so it needs $C_6^2 = 15$ bits to represent the edges.

5.1 DNA Code

For each bit used in our new algorithm, there are two distinct 15 bases value sequences designed for our experiment. One represents the value one for x_i^1 and the other one represents the value zero for x_i^0. DNA sequences and the energy used generated by the modified Adleman program are shown in **Table 2**.

Table 2. DNA Sequences chosen to represent the 15 bits (blocks) for settling the Ramsey Number Problem $R(3,3) = 6$

bit	5'→3'DNA Sequence	Entropy energy (*S*)	Enthalpy energy (*H*)	Free energy (*G*))
x^0_1	*AATTAACAATCATCT*	273.0	104.3	24.1
x^1_1	*AATTCCCATTCCCTA*	273.0	108.5	27.8
x^0_2	*CCACCCTCATCCTAT*	266.6	101.3	21.5
x^1_2	*AATTCACAAACAATT*	299.4	114.4	25.0
x^0_3	*ATTCACTTCTTTAAT*	283.5	107.8	23.0
x^1_3	*CCTTTCTAACCTTCA*	272.6	103.8	28.2
x^0_4	*TCTCTCTCTAATCAT*	270.5	105.2	28.2
x^1_4	*TTTACCCTCATTACT*	255.8	97.6	20.9
x^0_5	*CTTACAATCTTACCT*	234.9	106.7	23.0
x^1_5	*AACATACCCCTAATC*	291.2	112.6	25.6
x^0_6	*ATTCTAACTCTACCT*	277.1	105.2	25.0
x^1_6	*AATTCATCATCAATT*	270.8	105.0	24.1
x^0_7	*TCTCCCTATTTATTT*	278.6	107.8	24.3
x^1_7	*TAATTCCATAACCTA*	288.2	110.2	25.9
x^0_8	*CCAATTCCAATAATC*	290.6	113.0	24.5
x^1_8	*AAAACTCACCCTCCT*	288.7	113.7	22.3
x^0_9	*AAATTAATACATTAA*	294.0	115.9	28.2
x^1_9	*TCTAATATAATTACT*	283.7	104.8	22.4
x^0_{10}	*CCATCATCTACCTTA*	286.3	108.4	22.6
x^1_{10}	*TTACTCTTAACATCT*	282.8	112.1	24.4
x^0_{11}	*TATCTTTCTTTATCA*	285.7	111.3	25.9
x^1_{11}	*TTAATCAAATCCCTA*	266.0	102.1	22.6
x^0_{12}	*AATTCACTTTCTATC*	275.5	105.3	22.9
x^1_{12}	*CTTCTCCACTATACT*	288.3	111.1	27.8
x^0_{13}	*CTCTTAATCTCATTC*	291.5	112.4	25.3
x^1_{13}	*CCTTATCATCCAATC*	284.9	112.8	24.3
x^0_{14}	*AAACTCTACATACAC*	285.5	109.9	27.0
x^1_{14}	*TTTCAATAACACCTC*	271.6	105.6	24.3
x^0_{15}	*CCTAAATCTCCAATA*	266.0	101.9	22.4
x^1_{15}	*AAATCTATCTAATTC*	283.6	106.6	21.9

5.2 Solving Process of the Novel Algorithm for the Ramsey Number Problem

For the purposes of simulating **algorithm 5**, we make use of our example ($R(3, 3) = 6$). The vertex set is $V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$. For convenience, we let i to represent the vertex v_i ($1 \le i \le 6$) and the vertex set $V = \{1, 2, 3, 4, 5, 6\}$. From the algorithm, we first partition the six vertices of V into two parts $V_1 = \{1, 2, 3\}$ and $V_2 = \{4, 5, 6\}$. Now, solving the Ramsey Number Problem $R(3,3,6)$, we present the solution by solving the Ramsey Number Problem R(3,3,3).

By executing line 1of algorithm 5, the algorithm produces the subspace of the Ramsey

Number $R(3,3,3)$ and we get two tubes T_1 and T_2. In tube T_1 we consider the edges only to refer to $V_1 = \{1, 2, 3\}$ and get the DNA Strands {000, 001, 010, 011, 100, 101, 110, 111} that represent the eight graphs in **Figure 4** respectively.

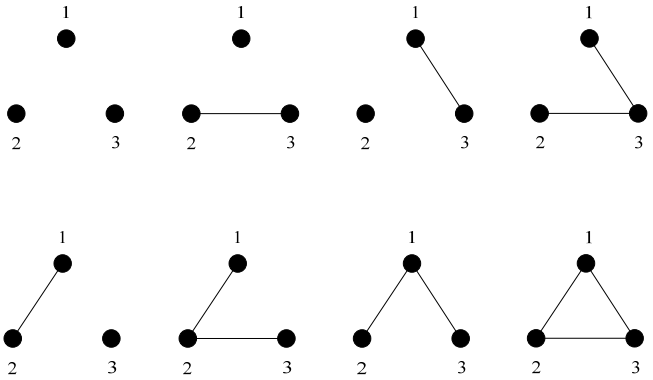

Figure 4. Eight graphs with the vertexes $\{v_1, v_2, v_3\}$

In tube T_2 we consider the edges only to refer to $V_1 = \{1, 2, 3\}$ and get the DNA Strands {000, 001, 010, 011, 100, 101, 110, 111} that represent the eight graphs in **Figure 5** respectively.

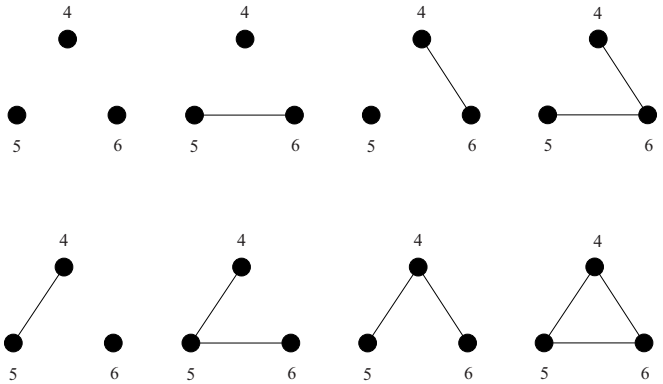

Figure 5. Eight graphs with the vertexes $\{v_4, v_5, v_6\}$

On the execution of Parallel_Searcher, the illegal strands {000,111} in tubes T_1 and T_2 are removed. Therefore, the strands left in T_1 are {001, 010, 011, 100, 101, 110}. The strands left in T_2 are {001, 010, 011, 100, 101, 110}.

After the execution of line 1, we get the satisfiable solution space of the Ramsey Number Problem $R(3,3,3)$. To produce the solution space for our problem $R(3,3,6)$, line 2 is executed. On the execution of line 2, the *Unit operation* binds strands in tube T_1 with strands in tube T_2. After the *Unit operation*, the DNA strands of the solution space are{ 001001, 010001, 011001, 100001, 101001, 110001, 001010, 010010, 011010, 100010, 101010, 110010, 001011, 010011, 011011, 100011, 101011, 110011, 001100, 010100, 011100, 100100,101100, 110100, 001101, 010101, 011101, 100101, 101101,110101, 001110, 010110, 011110, 100110, 101110, 110110 }.

The edges refer to both V_1 and V_2 such as $e(1,4)$, $e(1,5)$, $e(1,6)$, $e(2,4)$, $e(2,5)$, $e(2,6)$, $e(3,4)$, $e(3,5)$, $e(3,6)$ will be considered present. The full solution space is shown parting into four parts in **Table 3**. If we let x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10} x_{11}, x_{12}, x_{13}, x_{14} and x_{15} to represent *the edge* $e_{1,2}$, $e_{1,3}$, $e_{2,3}$, $e_{4,5}$, $e_{4,6}$, $e_{5,6}$, $e_{1,4}$, $e_{2,4}$, $e_{3,4}$, $e_{1,5}$, $e_{2,5}$, $e_{3,5}$, $e_{1,6}$,$e_{2,6}$,$e_{3,6}$, respectively, we get the solutions from the column one of **Table 3(See Table 4).**

Table 3. The satisfiable solution space of the Ramsey Number Problem R (3, 3)

The solution space of the edge $e_{1,2}$ $e_{1,3}$ $e_{2,3}$ $e_{4,5}$ $e_{4,6}$ $e_{5,6}$	*The solution of the edge* $e_{1,4}$ $e_{2,4}$ $e_{3,4}$	*The solution of the edge* $e_{1,5}$ $e_{2,5}$ $e_{3,5}$	*The solution of the edge* $e_{1,6}$ $e_{2,6}$ $e_{3,6}$
001001	011, 100, 101, 110, 111	011, 100, 101, 110, 111	011, 100, 101, 110, 111
010001	010, 011, 100, 110	010, 011, 100, 110	010, 011, 100, 110
011001	010, 011, 100, 110	010, 011, 100, 110	010, 011, 100, 110
100001	001, 011, 100, 101	001, 011, 100, 101	001, 011, 100, 101
101001	001, 011, 100, 101	001, 011, 100, 101	001, 011, 100, 101
110001	000, 001, 010, 011, 100	000, 001, 010, 011, 100	000, 001, 010, 011, 100
001010	011, 100, 101, 110, 111	011, 100, 101, 110, 111	011, 100, 101, 110, 111
010010	010, 011, 100, 110	010, 011, 100, 110	010, 011, 100, 110
011010	010, 011, 100, 110	010, 011, 100, 110	010, 011, 100, 110
100010	001, 011, 100, 101	001, 011, 100, 101	001, 011, 100, 101
101010	001, 011, 100, 101	001, 011, 100, 101	001, 011, 100, 101
110010	000, 001, 010, 011, 100	000, 001, 010, 011, 100	000, 001, 010, 011, 100
001011	011, 100, 101, 110, 111	011, 100, 101, 110, 111	011, 100, 101, 110, 111
010011	010, 011, 100, 110	010, 011, 100, 110	010, 011, 100, 110
011011	010, 011, 100, 110	010, 011, 100, 110	010, 011, 100, 110
100011	001, 011, 100, 101	001, 011, 100, 101	001, 011, 100, 101

101011	001, 011, 100, 101	001, 011, 100, 101	001, 011, 100, 101
110011	000, 001, 010, 011, 100	000, 001, 010, 011, 100	000, 001, 010, 011, 100
001100	011, 100, 101, 110, 111	011, 100, 101, 110, 111	011, 100, 101, 110, 111
010100	010, 011, 100, 110	010, 011, 100, 110	010, 011, 100, 110
011100	010, 011, 100, 110	010, 011, 100, 110	010, 011, 100, 110
100100	001, 011, 100, 101	001, 011, 100, 101	001, 011, 100, 101
101100	001, 011, 100, 101	001, 011, 100, 101	001, 011, 100, 101
110100	000, 001, 010, 011, 100	000, 001, 010, 011, 100	000, 001, 010, 011, 100
001101	011, 100, 101, 110, 111	011, 100, 101, 110, 111	011, 100, 101, 110, 111
010101	010, 011, 100, 110	010, 011, 100, 110	010, 011, 100, 110
011101	010, 011, 100, 110	010, 011, 100, 110	010, 011, 100, 110
100101	001, 011, 100, 101	001, 011, 100, 101	001, 011, 100, 101
101101	001, 011, 100, 101	001, 011, 100, 101	001, 011, 100, 101
110101	000, 001, 010, 011, 100	000, 001, 010, 011, 100	000, 001, 010, 011, 100
001110	011, 100, 101, 110, 111	011, 100, 101, 110, 111	011, 100, 101, 110, 111
010110	010, 011, 100, 110	010, 011, 100, 110	010, 011, 100, 110
011110	010, 011, 100, 110	010, 011, 100, 110	010, 011, 100, 110
100110	001, 011, 100, 101	001, 011, 100, 101	001, 011, 100, 101
101110	001, 011, 100, 101	001, 011, 100, 101	001, 011, 100, 101
110110	000, 001, 010, 011, 100	000, 001, 010, 011, 100	000, 001, 010, 011, 100

Table 4. The satisfiable solution space of from the column one in **Table 3**

$x_1\ x_2\ x_3\ x_4\ x_5\ x_6\ x_7\ x_8\ x_9$ $x_{10}\ x_{11}\ x_{12}\ x_{13}\ x_{14}\ x_{15}$	$x_1\ x_2\ x_3\ x_4\ x_5\ x_6\ x_7\ x_8\ x_9$ $x_{10}\ x_{11}\ x_{12}\ x_{13}\ x_{14}\ x_{15}$	$x_1\ x_2\ x_3\ x_4\ x_5\ x_6\ x_7\ x_8\ x_9$ $x_{10}\ x_{11}\ x_{12}\ x_{13}\ x_{14}\ x_{15}$	$x_1\ x_2\ x_3\ x_4\ x_5\ x_6\ x_7\ x_8\ x_9$ $x_{10}\ x_{11}\ x_{12}\ x_{13}\ x_{14}\ x_{15}$
001001011011011	001001100011011	001001101011011	001001110011011
001001111011011	001001011100011	001001100100011	001001101100011
001001110100011	001001111100011	001001011101011	001001100101011
001001101101011	001001110101011	001001111101011	001001011110011
001001100110011	001001101110011	001001110110011	001001111110011
001001011111011	001001100111011	001001101111011	001001110111011
001001111111011	001001011011100	001001100011100	001001101011100
001001110011100	001001111011100	001001011100100	001001100100100
001001101100100	001001110100100	0010011111 00100	001001011101100
001001100101100	001001101101100	001001110101100	001001111101100
001001011110100	001001100110100	001001101110100	001001110110100
001001111110100	001001011111100	001001100111100	001001101111100
001001110111100	001001111111100	001001011011101	001001100011101
001001101011101	001001110011101	001001111011101	001001011100101
001001100100101	001001101100101	001001110100101	001001111100101
001001011101101	001001100101101	001001101101101	001001110101101
001001111101101	001001011110101	001001100110101	001001101110101
001001110110101	001001111110101	001001011111101	001001100111101
001001101111101	001001110111101	001001111111101	001001011011110
001001100011110	001001101011110	001001110011110	001001111110111
001001011111111	001001100111111	001001101111111	001001110111111
001001101100110	001001110100110	001001111100110	001001011101110
001001111011110	001001011100110	001001100100110	001001100101110

001001101101110	001001110101110	001001111101110	001001011110110
001001100110110	001001101110110	001001110110110	001001111110110
001001011111110	001001100111110	001001101111110	001001110111110
001001111111110	001001011011111	001001100011111	001001101011111
001001110011111	001001111011111	001001011100111	001001100100111
001001101100111	001001110100111	001001111100111	001001011101111
001001100101111	001001101101111	001001110101111	001001111101111
001001011110111	001001100110111	001001101110111	001001110110111
001001111111111			

For example, the DNA strand 001001011011011 represents the graph shown in **Figure 6.** By the parallel searcher, we can see that there will be a 3-clique containing the vertex 2, 3 and 4 in Figure 6. Hence, the DNA strand 001001011011011 is no part of the solution and will be removed from the solution space. Applying the parallel searcher to the DNA strand above, we find out that no DNA strand from column one of **Table 3** is left.

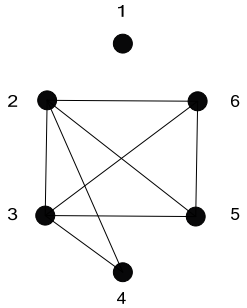

Figure 6. The graph represented by the DNA strand '001001011011011'

After searching all the DNA strands in **Table 3** with the parallel searcher, there is no DNA strand left in the final data pool and detecting the test tube *T* returns 'no'. So the solution to the Ramsey Number Problem *R*(3,3) is 6.

6. Discussion and Conclusion

As it has been shown, DNA computing has the advantage of massive computational parallelism [31]. However, from [32], we can see that DNA computer's power is limited by the volume of DNA that can be manipulated in the lab. At present, the DNA volume's exponential explosion problem has been the critical factor that constraint the development of

the DNA computing. In this paper, we demonstrated the power of DNA-based supercomputing by showing that the Ramsey number problem can be solved under this computation model efficiently. For the purposes of decreasing the DNA volume complexity of the Ramsey Number problem, the divide and conquer algorithm is taken into the DNA-based computing and a novel DNA computing algorithm is proposed. Compared to enumerating DNA computing algorithm for Ramsey Number problem, our new method is more volume-efficient and has a lower error rate.

In the future, molecular computing may be a good choice for massively parallel computations. For the objective to reach a free stage in using DNA computers just as using classical digital computers, many technical difficulties have to overcome before this can be a reality such as real time updating a solution when the initial condition of a problem changes, finding out the exact answer quickly and efficiently or when the size of the initial data pool increases exponentially with the number of variables. We hope that our study will make a contribution in DNA computing and can also clarify that DNA-based computing is a technology that is worth pursuing.

Acknowledgement

This research is supported by the Projects of National Natural Science Foundation of China under grants (60603053, 90715029), the Ministry of Education for New Century Excellent Talent Support Program, Natural Science Foundation of Zhejiang province under grant Y1090264, Natural Science Foundation of Hunan province under grant 07JJ6109.

References

[1] K. H. Zimmermann, Efficient DNA sticker algorithms for NP–complete graph problems, *Comput. Phys. Commun.* **144** (2002) 297–309.

[2] W. L. Chang, M. Ho, M. Guo, X. Jiang, J. Xue, M. Li, Fast parallel DNA–based algorithms for molecular computation: Determining a prime number, *Proceedings of the Third International Conference on Information Technology and Applications*, 2005, pp. 447–452.

[3] W. L. Chang, T. T. Ren, J. Luo, M. Feng, M. Guo, K. W. Lin, Quantum algorithms for biomolecular solutions of the satisfiability problem on a quantum machine, *IEEE Trans. Nanobiosci.* **7** (2008) 215–222.

[4] D. F. Li, X. R. Li, H. T. Huang, Scalability of the surface–based DNA algorithm for 3–SAT, *BioSystems* **85** (2006) 95–98.

[5] M. Ho, W. L. Chang, M. Guo, Fast parallel solution for set–packing and clique problems by DNA–based computing, *IEICE Trans. Inf. Syst.* **E87-D(7)** (2004) 1782–1788.

[6] M. Ho, Fast parallel molecular solutions for DNA–based supercomputing: The subset–product problem, *BioSystems* **80** (2005) 233–250.

[7] W. L. Chang, M. Guo, M. Ho, Fast parallel molecular algorithms for DNA–based computation, *IEEE Trans. Nanobiosci.* **4** (2005) 133–163.

[8] W. L. Chang, M. Guo, Molecular solutions for the subset–sum problem on DNA–based supercomputing, *BioSystems* **73** (2004) 117–130.

[9] Y. Li, C. Fang, Q. Ouyang, Genetic algorithm in DNA Computing: A solution to the maximal clique problem, *Chinese Sci. Bull.* **49** (2004) 967–971.

[10] Q. Huiqin, L. Mingming, Z. Hong, Solve maximum clique problem by sticker model in DNA computing, *Prog. Nat. Sci.* **14** (2004) 1116–1121.

[11] K. L. Li, F. J. Yao, J. Xu, Improved molecular solutions for the knapsack problem on DNA–based supercomputing, *Chinese J. Comput. Res. Develop.* **44** (2007) 1063–1070.

[12] E. Horowitz, S. Sahni, Computing partitions with applications to the knapsack problem, *J. ACM* **21**(1974) 277–292.

[13] S. Y. Hsieh, M. Y. Chen. A DNA–based graph encoding scheme with its applications to graph isomorphism problem, *Appl. Math. Comput.* **203** (2008) 502–512.

[14] M. Guo, M. Ho, W. L. Chang. Fast parallel molecular solution to the dominating–set problem on massively parallel biocomputing, *Parallel Comput.* **30** (2004) 1109–1125.

[15] B. Fu, R. Beigel, Length bounded molecular computing, *BioSystems* **52** (1999) 155–163.

[16] B. Fu. *Volume Bounded Molecular Computation*, Ph.D. Thesis, Dep. Comput. Sci., Yale Univ., 1997

[17] M. R. Garey, D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP–Completeness*, Freeman, San Francisco, 1979.

[18] R. Graham, B. Rothschild, J. Spence, Ramsey theory, *Elec. J. Comb. Dyn. Surv.* **11** (2006) 3–36.

[19] B. D. Mckay, K M Zhang, The value of the Ramsey number R(3,8), *J. Graph Theor.* **16** (1992) 99–105.

[20] I. M. Martínez–Pérez, K. H. Zimmermann, Parallel bioinspired algorithms for NP complete graph problems, *J. Parallel Distrib. Comput.* **69** (2009) 221–229.

[21] R. E. Greennwood, A M Cleason, Combinatorial relations and chromatic graphs, *Canadian J. Math.* **7** (1955) l–7.

[22] J. E. Graver, J. Yackel, Some graph theoretic results associated with Ramsey's theorem, *J. Comb. Theor.* **4** (1968) 125–175.

[23] J. G. Kalbfleisch, *Chromatic Graphs and Ramsey's Theorem*, Waterloo, Canada: Univ. Waterloo, 1966.

[24] B. D. Mckay, S. P. Radziszowski, *R*(4, 5)=25, *J. Graph Theor.* **19** (1995) 309–322.

[25] G. Exoo, Announcement: on the Ramsey numbers R(4, 6), R(5, 6) and R(3, 12), *Ars Combin.* **35** (1993) 85–85.

[26] G. Exoo, Alowerboundfor *R*(5, 5), *J. Graph Theor*. **13** (1989) 97–98.

[27] T. Spenced, Upper bounds for Ramsey numbers via linear programming [EB/OL], (1993) [2007]. http://www. mathworld.wolfram.com/RamseyNumbor.html.

[28] X. D. Xu, Z. Xie, G. Exoo, S. P. Radziszowski, Constructive lower bounds on classical multicolor Ramsey numbers, *Elec. J. Comb.* **11** (2004) 35–58.

[29] S. D. Lu, *The Experimentation of Molecular Biology,* Peking Union Medical College Press.1999

[30] E. Horowitz, S. Sahni, Computing partitions with applications to the knapsack problem, *J. ACM* **21** (1974) 277–292.

[31] K. L. Li, S. T. Zhou, J. Xu, Fast parallel molecular algorithms for DNA–based computation: Solving the elliptic curve discrete logarithm problem over $GF(2^n)$, *J. Biomed. Biotech.* **1** (2008) 1–10.

[32] B. Richard, F. Bin, Solving intractable problems with DNA computing, *Proceedings of the Thirteenth Annual IEEE Conference on Computational Complexity*, 1998, pp. 154–168.