Proc. of the Intl. Conf. on Advances In Bio-Informatics And Environmental Engineering - ICABEE 2014.
Copyright © Institute of Research Engineers and Doctors. All rights reserved.
ISBN: 978-1-63248-019-4 doi: 10.15224/ 978-1-63248-019-4-02

Constructing a Fast Bioinformatics Algorithm to Solve Cancer Genome Assembly Using Enhanced Euler Path on Constructed De Bruijn Bioinformatics Graph

Michael Shan-Hui Ho, Kun-Yu Hung, Paul Pin-Shuo Huang, Jack Dao-Jie Li and Dio Feng-Yu Chung

Abstract—Cancer is defined as a disease that involves changes or mutations in the cell genome. Cancer genome sequencing has been recognized as a NP problem. Cancer genome sequencing includes cancer genome assembly and cancer genome alignment is through early detection improving survival opportunity of cancer patients. In this research, a bioinformatics approach uses a proposed modified Euler path on a constructed De Bruijn cancer genome graph for solving cancer genome assembly. This fast DNA algorithm fully utilizes parallelism to conquer time complexity bottleneck, and improves any cancer genome Assembly more efficient. The experimental results of cancer genome reassemble is estimated in O(n^3) polynomial bound.

***Keywords*—Cancer Genome Assembly; Cancer Genome Alignment; De Bruijn Graphs; Euler Path**

I. Introduction

Cancer is an important public health concern around the world. Cancer is defined as a disease that involves changes or mutations in the cell genome. These changes (mutations) produce proteins that disrupt the delicate cellular balance between cell division and quiescence, resulting in cells that keep dividing to form cancers. The underlying cause of mutations leading to cancer is DNA damage.

DNA damage In human cells, the estimated average number of DNA damages occurring per hour is about 800, and the number per day is about 19,200[1]. Under normal circumstances, healthy cells repair virtually all of these damages. Damages that are not repaired are termed mutations. When a single cell acquires enough mutations in the DNA sequence of relevant 'cancer gene(s)' it begins to behave in an abnormal way characteristic of cancer.

Michael Shan-Hui Ho
Department of Electrical Engineering, NTPU,
New Taipei City, Taiwan, ROC
Kun-Yu Hung
Department of Information Management, MCU
Taoyuan Country, Taiwan, ROC
Paul Pin-Shuo Huang
Department of Electrical Engineering, NTPU,
New Taipei City, Taiwan, ROC
Jack Dao-Jie Li
Department of Electrical Engineering, NTPU,
New Taipei City, Taiwan, ROC
Dio Feng-Yu Chung
Department of Electrical Engineering, NTPU,
New Taipei City, Taiwan, ROC

Cancer genome sequencing includes cancer genome assembly and cancer genome alignment is through early detection improving survival opportunity of cancer patients.

A. *Cancer Genome Assembly*

For the last 30 years, fragment assembly followed the ''overlap–layout–consensus'' paradigm[2]. Although this approach proved to be useful in assembling clones, it faces difficulties in genomic shotgun assembly: the algorithms often unable to resolve repeats even in prokaryotic genomes. So, in the past decade, there has been a new approach: instead of ''overlap–layout–consensus'' paradigm, the new algorithm is based on the notion of the De Bruijn graph and transforms the cancer genome assembly problem into an Euler super path problem [3].

B. *The Euler Path Problem*

The Euler path problem can be traced back 300 years, to the Prussian city of Königsberg (Kaliningrad, Russia). In Königsberg city, seven bridges crossed four parts of the city. Residents enjoyed strolling through their city, and they wondered if every part of the city could be visited by walking across each of the seven bridges exactly once and returning to one's starting location.

A mathematician, Leonhard Euler [4], proposed a solution in 1735; it made a conceptual breakthrough that would solve this "Bridges of Königsberg" problem. First, Euler assumes each landmass represents a point (also called vertex) and each bridge denotes a line segment (also called edge) connecting two points/ vertices. This creates a network graph of vertices connected by edges. By describing a procedure for determining whether an arbitrary graph contains a path that visits every edge exactly once. This solution, called the Euler path, is based on the concept of an undirected graph. The De Bruijn graph is a kind of digraph, for a digraph, Euler path not only to traverse all the edges, but also follows the correct direction. In a more rigorous description, there exists a path which can traverse the graph without repeating all the edges and visits them by directed edges along the direction.

C. *De Bruijn Graph*

Dutch mathematician Nicolaas De Bruijn finds a cyclic sequence of letters taken from a given alphabet for which

Proc. of the Intl. Conf. on Advances In Bio-Informatics And Environmental Engineering - ICABEE 2014.
Copyright © Institute of Research Engineers and Doctors. All rights reserved.
ISBN: 978-1-63248-019-4 doi: 10.15224/ 978-1-63248-019-4-02

every possible word of a certain length (k) appears as a string of consecutive characters in the cyclic sequence exactly once.[4][5]

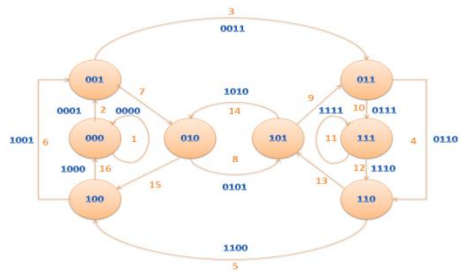


Figure 1: De Bruijn graph.

There exist $n^k$$k$-mers in an alphabet containing n symbols. If our alphabet is instead 0 and 1, then all possible 3-mers are simply given by all eight 3-digit binary numbers: 000, 001, 010, 011, 100, 101, 110and 111. The circular superstring 0001110100 not only contains all 3-mers but also is as short as possible, as it contains each 3-mer exactly once shown in Figure 1.

II. Sticker-Based Model

The sticker-based model employs two basic groups of single-stranded DNA molecules in its representation of a bit string. Consider a memory strand*N*bases in length subdivided into K non-overlapping regions each M bases long (thus,$N \geq M * K$). Each region is identified with exactly one bit position (or equivalently one Boolean variable) during the course of the computation. Each memory strand along with its annealed stickers (if any) represents one bit string shown in Figure 2.

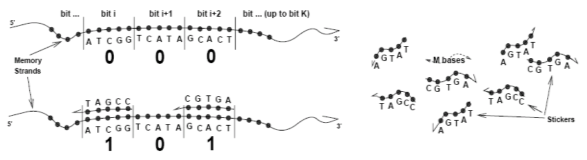


Figure 2:Memory strands of the sticker model

In Table 1, a two-bit sticker ($s_{m,1}$and $s_{m,2}$) model is used to represent letters A, G, C, T.

Table 1:Two-bit sticker-based model

$s_{m,1}$	$s_{m,2}$	Letter of m^{th} site
0	0	A
0	1	G
1	0	C
1	1	T

III. DNA Manipulations

DNA Manipulations is also called Adleman-Lipton model. A test tube is a set of molecules of DNA (a multi-set of finite strings over the alphabet {A,C,G,T}. In this subsection, DNA Model of computation has eight biological operations, shown as following:

1. ***Extract***: Given a tube *T* and a short single strand of DNA, *S*. This operation produces two new tubes +(*T, S*) and –(*T, S*). In tube +(*T, S*), all of the molecules contain strand *S* as sub-strand, on the contrary, all molecules in tube –(*T, S*) do not contain strand *S*.

2. ***Merge***: The representation also can be symbol ∪. Given tubes T_1 to T_x, yield ∪□(T_1, T_2, …, T_x), where ∪□(T_1, T_2, …, T_x) = T_1 ∪ T_2 ∪ … ∪ T_x. This operation pours contents of tubes T_1 to T_x into one tube without any change in the individual strands.

3. ***Detect***: Given a tube *T*, if *T* includes at least one DNA molecule then return a response "YES", if *T* contains no DNA molecules then get a response "NO".

4. ***Discard***: Given a tube *T*, this operation will discard tube *T*.

5. ***Amplify***: Given a tube *T*, Amplify (*T*, T_1, T_2) operation will produce two new tubes T_1 and T_2. Tube T_1 and T_2 are totally copy from tube *T* (T_1 and T_2 are now identical) and tube *T* becomes an empty tube.

6. ***Append***: Given a tube *T* containing a short strand of DNA, *S*. This operation will append *S* onto the end of every strand in *T*.

7. ***Append-head***: Given a tube *T* containing a short strand of DNA, *S*. This operation will append *S* onto the head of every strand in *T*.

8. ***Read***: Given a tube *T*, this operation describes every single molecule contained in 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.

IV. Basic Bioinformatics Circuitry

We use logic truth tables to optimize and complete logic bio-circuit operations that can construct most basic DNA logic circuits. These DNA logic circuits (gates) work in test tubes to implement basic logic operations. These gates are AND, OR, XOR.

A. AND Operation on Bioinformatics Computing

The AND operation of a bit with two input Boolean variables U and V generates a result of 1 or 0. The logic circuitry of parallel AND on one bit is shown in Figure 3. The corresponding truth table of the one-bit AND is shown in Table 2.

Table 2: The truth table of the one-bit AND

Input		Output
U_k	V_k	$AND_k = U_k \wedge V_k$
0	0	0
0	1	0

Proc. of the Intl. Conf. on Advances In Bio-Informatics And Environmental Engineering - ICABEE 2014.
Copyright © Institute of Research Engineers and Doctors. All rights reserved.
ISBN: 978-1-63248-019-4 doi: 10.15224/ 978-1-63248-019-4-02

1	**0**	**0**
1	**1**	**1**

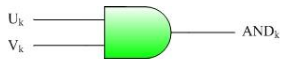


Figure 3: Logic circuitry of parallel AND on one bit

```
**ParallelOneBitAND**$(T_0, U_k, V_k, AND_k)$
$T_1^{U=1} = +(T_0, U_k^1)$ and $T_1^{U=0} = -(T_0, U_k^1)$.
$T_2^{U=1,V=1} = +(T_1^{U=1}, V_k^1)$ and $T_2^{U=1,V=0} = -(T_1^{U=1}, V_k^1)$
$T_2^{U=0,V=1} = +(T_1^{U=0}, V_k^1)$ and $T_2^{U=0,V=0} = -(T_1^{U=0}, V_k^1)$
If ($Detect(T_2^{U=1,V=1})$ = = “yes”) then
Append-head($T_2^{U=1,V=1}$, $AND_k^1$)EndIf
If ($Detect(T_2^{U=1,V=0})$ = = “yes”) then
Append-head($T_2^{U=1,V=0}$, $AND_k^0$)EndIf
If ($Detect(T_2^{U=0,V=1})$ = = “yes”) then
    Append-head($T_2^{U=0,V=1}$, $AND_k^0$)EndIf
If ($Detect(T_2^{U=0,V=0})$ = = “yes”) then
    Append-head($T_2^{U=0,V=0}$, $AND_k^0$)EndIf
$T_0 = \cup(T_2^{U=1,V=1}, T_2^{U=1,V=0}, T_2^{U=0,V=1}, T_2^{U=0,V=0})$
EndAlgorithm
```

Figure 4: Parallel AND operation on one bit algorithm

B. *OR Operation on Bioinformatics Computing*

The OR operation of a bit with two input Boolean variables *U* and *V* produces a result of 1 or 0. The logic circuitry of parallel OR on one bit is shown in Figure 5. The corresponding truth table of the one-bit OR is shown in Table 3.

Table 3: The truth table of the one-bit OR

Input		Output
U_k	V_k	$OR_k = U_k \vee V_k$
0	0	0
0	1	1
1	0	1
1	1	1

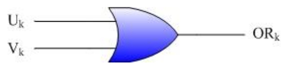


Figure 5: Logic circuitry of parallel OR on one bit

```
**ParallelOneBitOR**$(T_0, U_k, V_k, OR_k)$
$T_1^{U=1} = +(T_0, U_k^1)$ and $T_1^{U=0} = -(T_0, U_k^1)$.
$T_2^{U=1,V=1} = +(T_1^{U=1}, V_k^1)$ and $T_2^{U=1,V=0} = -(T_1^{U=1}, V_k^1)$
$T_2^{U=0,V=1} = +(T_1^{U=0}, V_k^1)$ and $T_2^{U=0,V=0} = -(T_1^{U=0}, V_k^1)$
If ($Detect(T_2^{U=1,V=1})$ = = “yes”) then
Append-head($T_2^{U=1,V=1}$, $OR_k^1$)EndIf
If ($Detect(T_2^{U=1,V=0})$ = = “yes”) then
    Append-head($T_2^{U=1,V=0}$, $OR_k^1$)EndIf
If ($Detect(T_2^{U=0,V=1})$ = = “yes”) then
    Append-head($T_2^{U=0,V=1}$, $OR_k^1$)EndIf
If ($Detect(T_2^{U=0,V=0})$ = = “yes”) then
    Append-head($T_2^{U=0,V=0}$, $OR_k^0$)EndIf
$T_0 = \cup(T_2^{U=1,V=1}, T_2^{U=1,V=0}, T_2^{U=0,V=1}, T_2^{U=0,V=0})$
EndAlgorithm
```

Figure 6: Parallel OR operation on one bit algorithm

C. *XOR Operation on Bioinformatics Computing*

The Exclusive-OR (XOR) operation of a bit which can generate an output of 1 or 0.The logic circuitry of parallel XOR on one bit is shown in Figure 7. The corresponding truth table of the one-bit XOR is shown in Table 4:

Table 4: The truth table of the one-bit XOR

Input		Output
U_k	V_k	$XOR_k = U_k \oplus V_k$
0	0	0
0	1	1
1	0	1
1	1	0

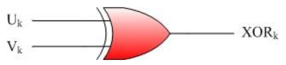


Figure 7: Logic circuitry of Parallel XOR on one bit

```
**ParallelOneBitXOR**$(T_0, U_k, V_k, XOR_k)$
$T_1^{U=1} = +(T_0, U_k^1)$ and $T_1^{U=0} = -(T_0, U_k^1)$.
$T_2^{U=1,V=1} = +(T_1^{U=1}, V_k^1)$ and $T_2^{U=1,V=0} = -(T_1^{U=1}, V_k^1)$
$T_2^{U=0,V=1} = +(T_1^{U=0}, V_k^1)$ and $T_2^{U=0,V=0} = -(T_1^{U=0}, V_k^1)$
If ($Detect(T_2^{U=1,V=1})$ = = “yes”) then
Append-head($T_2^{U=1,V=1}$, $XOR_k^0$)EndIf
If ($Detect(T_2^{U=1,V=0})$ = = “yes”) then
  Append-head($T_2^{U=1,V=0}$, $XOR_k^1$)EndIf
If ($Detect(T_2^{U=0,V=1})$ = = “yes”) then
  Append-head($T_2^{U=0,V=1}$, $XOR_k^1$)EndIf
If ($Detect(T_2^{U=0,V=0})$ = = “yes”) then
  Append-head($T_2^{U=0,V=0}$, $XOR_k^0$)EndIf
$T_0 = \cup(T_2^{U=1,V=1}, T_2^{U=1,V=0}, T_2^{U=0,V=1}, T_2^{U=0,V=0})$.
EndAlgorithm
```

Figure 8: Parallel XOR operation on one bit algorithm

D. *Bio-arithmetic Parallel Comparator on n Bits*

The following algorithm, ParallelComparator (T_0, T_0 *overlay*, T_a, T_b, *m*, *n*, *g*, *b*), is an n-bit comparator. Algorithm for parallel execution is shown in Figure 11.

```
**ParallelComparator**$(T_0, T_0^{overlay}, T_a, T_b, m, n, g, b)$
For $d = 0$ to Min($n-m, b-g$)
    For $p=n$ downto $m$
        OneBitComparator$(T_0^{=}, T_a, T_b, p, g+d)$
        If ($Detect(T_0^{=})$=“yes”) then
            $Append(T_0^{overlay}, O_{p,g+d}^{\ 1})$
            $Discard(T_0^{=})$  EndIf
    EndFor
EndFor
If ($Detect(T_0^{overlay})$=“yes”) then
    $T_0 = \cup(T_0, T_0^{overlay})$  EndIf
$Discard(T_0^{overlay})$
EndAlgorithm
```

Figure 9: Parallel comparator on n bits

V. Fast Bioinformatics Parallel Algorithms for Solving Cancer Genome Sequencing Assembly

In this research, the entire bioinformatics approach for solving cancer genome sequencing assembly is accomplished by algorithms I, II, and III. They are algorithms

Proc. of the Intl. Conf. on Advances In Bio-Informatics And Environmental Engineering - ICABEE 2014.
Copyright © Institute of Research Engineers and Doctors. All rights reserved.
ISBN: 978-1-63248-019-4 doi: 10.15224/ 978-1-63248-019-4-02

ConstructDeBruijnGraphs, **ParallelStickerAppended**, and **ModifiedEulerPathInDeBruijngraphs**.

```
Algorithm : Solving cancer genome  sequencing assembly
   (a)Algorithm I : ConstructDeBruijngraphs
   (b) Algorithm II :ParallelStickerAppended
   (c)Algorithm III: ModifiedEulerPathInDeBruijngraphs
EndAlgorithm
```

Figure 10: Proposed algorithms to construct a De Bruijn bioinformatics graph and using the modified Euler path to solve that De Bruijn bioinformatics graph

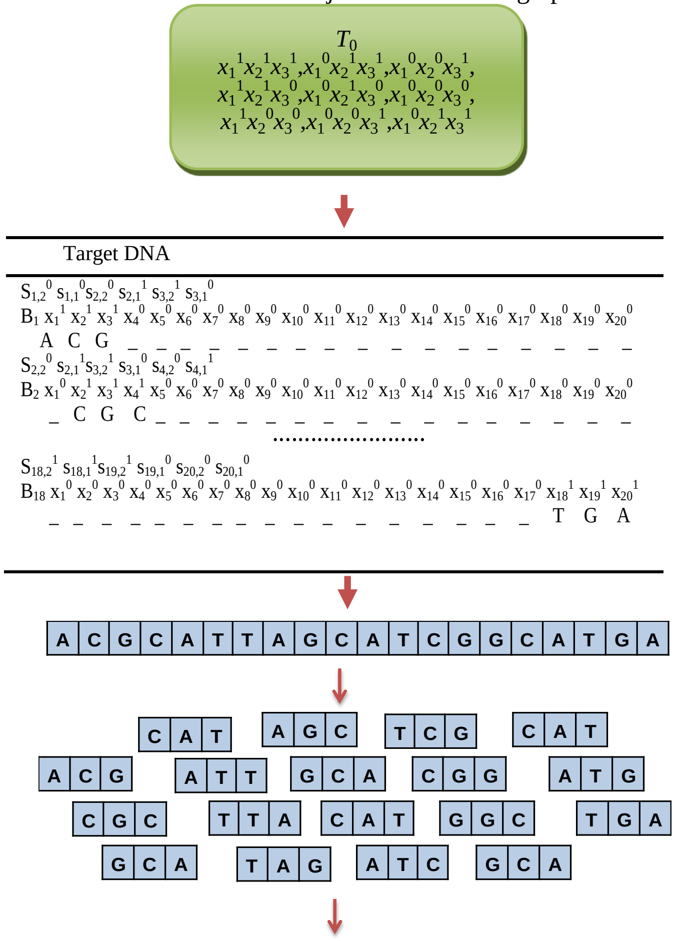

T_0
$x_1^1x_2^1x_3^1, x_1^0x_2^1x_3^1, x_1^0x_2^0x_3^1,$
$x_1^1x_2^1x_3^0, x_1^0x_2^1x_3^0, x_1^0x_2^0x_3^0,$
$x_1^1x_2^0x_3^0, x_1^0x_2^0x_3^1, x_1^0x_2^1x_3^1$
Target DNA
$S_{1,2}^0\ S_{1,1}^0\ S_{2,2}^0\ S_{2,1}^1\ S_{3,2}^1\ S_{3,1}^0$
$B_1\ x_1^1\ x_2^1\ x_3^1\ x_4^0\ x_5^0\ x_6^0\ x_7^0\ x_8^0\ x_9^0\ x_{10}^0\ x_{11}^0\ x_{12}^0\ x_{13}^0\ x_{14}^0\ x_{15}^0\ x_{16}^0\ x_{17}^0\ x_{18}^0\ x_{19}^0\ x_{20}^0$
A C G
$S_{2,2}^0\ S_{2,1}^1\ S_{3,2}^1\ S_{3,1}^0\ S_{4,2}^0\ S_{4,1}^1$
$B_2\ x_1^0\ x_2^1\ x_3^1\ x_4^1\ x_5^0\ x_6^0\ x_7^0\ x_8^0\ x_9^0\ x_{10}^0\ x_{11}^0\ x_{12}^0\ x_{13}^0\ x_{14}^0\ x_{15}^0\ x_{16}^0\ x_{17}^0\ x_{18}^0\ x_{19}^0\ x_{20}^0$
C G C
……………………
$S_{18,2}^1\ S_{18,1}^1\ S_{19,2}^1\ S_{19,1}^0\ S_{20,2}^0\ S_{20,1}^0$
$B_{18}\ x_1^0\ x_2^0\ x_3^0\ x_4^0\ x_5^0\ x_6^0\ x_7^0\ x_8^0\ x_9^0\ x_{10}^0\ x_{11}^0\ x_{12}^0\ x_{13}^0\ x_{14}^0\ x_{15}^0\ x_{16}^0\ x_{17}^0\ x_{18}^1\ x_{19}^1\ x_{20}^1$
T G A
A C G C A T T A G C A T C G G C A T G A
C A T
A G C
T C G
C A T
A C G
A T T
G C A
C G G
A T G
C G C
T T A
C A T
G G C
T G A
G C A
T A G
A T C
G C A

Figure 11: Example of cutting process from DNA sequence into many equal length fragments

We cut the DNA sequence into 3 equal length portions and use a modified Euler path approach to find an optimal solution on a constructed De Bruijn bioinformatics graph. Given a DNA sequence, Figures 12 shows a special cutting process from a DNA sequence into many overlapped nucleon fragments in equal length.

The De Bruijn solution space of k-tuple - 1 fragments in tube T_0 is constructed in Figure 13. Each fragment denotes one vertex in the De Bruijn graph.

```
ParallelDeBruijnSolutionSpace($T_0$, $k$, $q$)
For $m = 1$ to $q - k+2$
      (1)$T_3 = +(T_0, x_m^1)$ and $T_4 = -(T_0, x_m^1)$
      For $n = m$ to $m + k - 2$
            (2a)$T_5 = +(T_3, x_n^1)$ and $T_6 = -(T_3, x_n^1)$
            (2b)$Amplify(T_5, T_3, T_3^{backup})$
      End For
      (3)If ($m + k - 1 <= q$) Then
            For $n = q$ down to $m + k - 1$
                  (3a)$T_7 = +(T_5, x_n^1)$ and $T_8 = -(T_5, x_n^1)$
                  (3b)$Amplify(T_8, T_5, T_5^{backup})$
            End For
         Else If
    $T_8 = \cup(T_8, T_5)$
       End If
      (4)$Append\text{-}head(T_8, B_m)$
      (5)$T_{solution\_space} = \square\ (T_{solution\_space}, T_8)$
      (6)$T_0 = \cup(T_0, T_4)$
End For
EndAlgorithm
```

Figure 12: Cutting process from DNA sequence into many equal length fragments

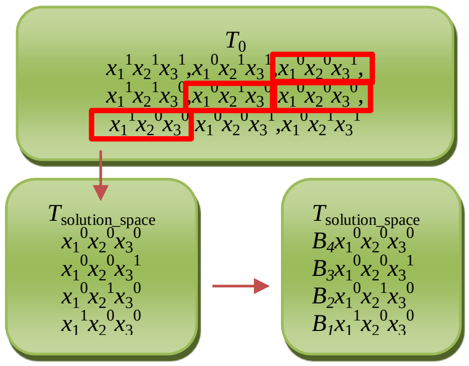


Figure 13: Example of Figure 12

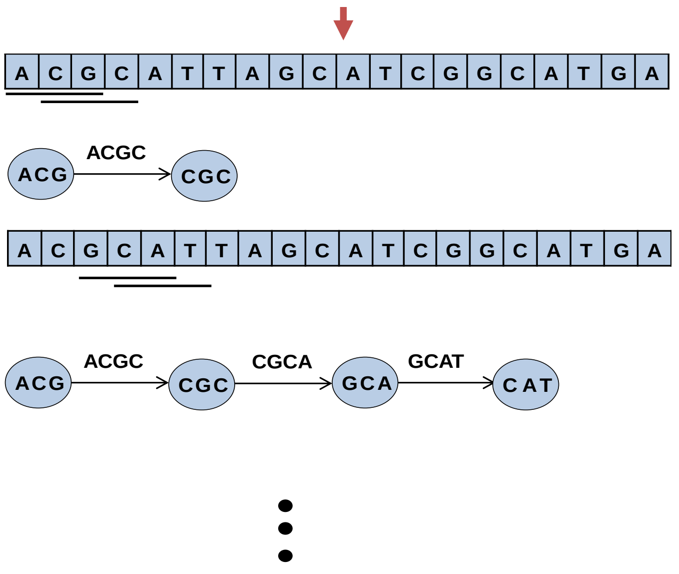

A C G C A T T A G C A T C G G C A T G A
ACG
ACGC
CGC
A C G C A T T A G C A T C G G C A T G A
ACG
ACGC
CGC
CGCA
GCA
GCAT
CAT

Figure 14: Example of finding all overlaps for De Bruijn graph construction

Proc. of the Intl. Conf. on Advances In Bio-Informatics And Environmental Engineering - ICABEE 2014.
Copyright © Institute of Research Engineers and Doctors. All rights reserved.
ISBN: 978-1-63248-019-4 doi: 10.15224/ 978-1-63248-019-4-02

A	C	G	C																
	C	G	C	A															
		G	C	A	T														
			C	A	T	T													
				A	T	T	A												
					T	T	A	G											
						T	A	G	C										
							A	G	C	A									
								G	C	A	T								
									C	A	T	C							
										A	T	C	G						
											T	C	G	G					
												C	G	G	C				
													G	G	C	A			
														G	C	A	T		
															C	A	T	G	
																A	T	G	A
A	C	G	C	A	T	T	A	G	C	A	T	C	G	G	C	A	T	G	A

Figure 15: Reassemble all overlaps for De Bruijn graph construction

Figure 16 appends stickers in the head of each fragment in the effective solution space.

```
ParallelStickerAppended($T_{Sticker}$,$T_{solution\_space}$, $q$)
(1)For $n = q$ downto 1
    (1a)$T_3 = +(T_{Sticker},x_n^1)$ and $T_4 = -(T_{Sticker}, x_n^1)$
      If ($X_n$= A) Then
        (1b)Append-head($T_3$,$s_{n,2}^0$) and Append-head($T_3$,$s_{n,1}^0$)
        (1c)$T_{Sticker} = \cup(T_3, T_4)$
      Else If ($X_n$= G) Then
        (1d)Append-head($T_3$,$s_{n,2}^1$) and Append-head($T_3$,$s_{n,1}^0$)
        (1e)$T_{Sticker} = \cup(T_3, T_4)$
      Else If ($X_n$= C) Then
        (1f)Append-head($T_3$,$s_{n,2}^0$) and Append-head($T_3$,$s_{n,1}^1$)
        (1g)$T_{Sticker} = \cup(T_3, T_4)$
      Else If ($X_n$= T) Then
        (1h)Append-head($T_3$,$s_{n,2}^1$) and Append-head($T_3$,$s_{n,1}^1$)
        (1i)$T_{Sticker} = \cup(T_3, T_4)$End If
EndFor
EndAlgorithm
```

Figure 16: Two-bit sticker model construction

$T_{solution\_space}$
$s_{4,1}^1 s_{4,2}^0 B_4 x_1^0 x_2^0 x_3^0$
$s_{3,1}^0 s_{3,2}^0 B_3 x_1^0 x_2^0 x_3^1$
$s_{2,1}^1 s_{2,2}^1 B_2 x_1^0 x_2^1 x_3^0$
$s_{1.1}^0 s_{1.2}^0 B_1 x_1^1 x_2^0 x_3^0$

Figure 17: Example of two-bit sticker model

The algorithm in Figure 18 creates the De Bruijn bioinformatics graph. This algorithm merges the repeated vertices into De Bruijn bioinformatics graph. Symbol "In_i" is used to indicate the *i*th repeated vertex.

```
ConstructDeBruijngraphs ($T_{DB\_graph}$,$T_{Sticker}$,$q$)
(1)$T_9 = +(T_{Sticker},B_1)$ and $T_{10} = -(T_{Sticker},B_1)$
(2)Append-head($T_9$,$In_1$)
(3)For $i$ = 2 to size of $T_{10}$
     (3a)$T_{11} = +(T_{10},B_i)$ and $T_{12} = -(T_{10},B_i)$
(4)For $j$ = 1 to size of $T_9$
       (4a)$T_{13} = +(T_9,B_j)$ and $T_{14} = -(T_9,B_j)$
               (4b)ParallelComparator($T_0^{table}$,$T_{13}$,$T_{11}$,$i$,$j$,$k$)
            (5)If (Detect($T_0^{table}$) = "yes") then
                  (5a)Append-head($T_{13}$,$In_i$)
                  (5b)$T_9 = \cup(T_{13},T_{14})$
               (5c)Discard($T_{13}$, $T_{14}$)
               Terminate the execution of the loopEnd If
         (6)$T_9 = \cup(T_{13},T_{14})$
         (7)Discard($T_{13}$, $T_{14}$)End For
         (8)If (Detect($T_0^{table}$) = "no") then
            (8a)Append-head($T_{11}$,$In_i$)
            (8b)$T_9 = \cup(T_9,T_{11})$End If
         (9)Discard($T_{10}$)
         (10) $T_{10} = \cup(T_{10},T_{12})$
         (11)Discard($T_{11}$, $T_{12}$)
 End For
   (12)$T_{DBGraph} = \cup(T_{DBGraph},T_9)$
EndAlgorithm
```

Figure 18: De Bruijn graph construction

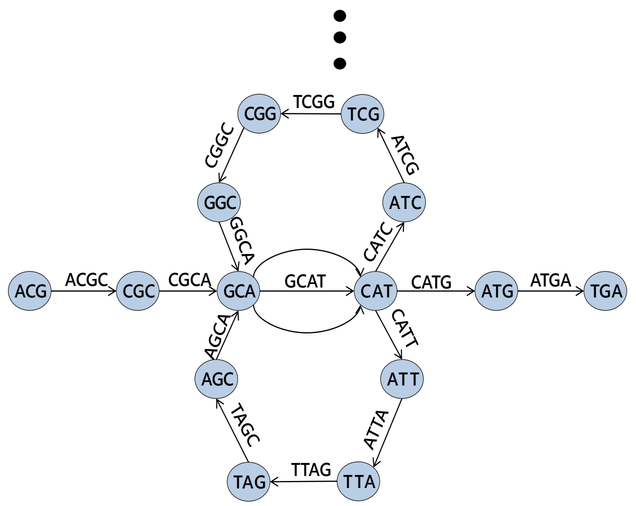


Figure 19: Example of De Bruijn graph construction

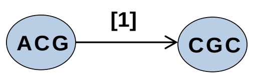
The algorithm in Figure 20 constructs an optimal Euler path by using the modified Euler path approach.

```
ModifiedEulerPathInDeBruijngraphs ($T_{DB\_graph}$,$T_{Routing}$,$q$, $k$)
For $m$ = 1 to $q-k+1$
   $T_{13} = +(T_{DBGraph},In_m)$ and $T_{14} = -(T_{DBGraph},In_m)$
      For $n = q-k+1$
     $T_{15} = +(T_{13},B_n)$ and $T_{16} = -(T_{13},B_n)$
     If (Detect($T_{15}$) = "YES") then
          Append($T_{Routing}$,$B_n$)
          Discard($T_{15}$, $T_{16}$)
          Terminate the execution of the loop
```

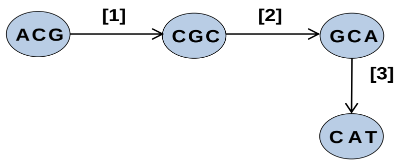
Proc. of the Intl. Conf. on Advances In Bio-Informatics And Environmental Engineering - ICABEE 2014.
Copyright © Institute of Research Engineers and Doctors. All rights reserved.
ISBN: 978-1-63248-019-4 doi: 10.15224/ 978-1-63248-019-4-02

```
            End If
        Discard($T_{15}$, $T_{16}$)
        End For
        $T_{DBGraph} = \cup(T_{13}, T_{14})$
End For
EndAlgorithm
```

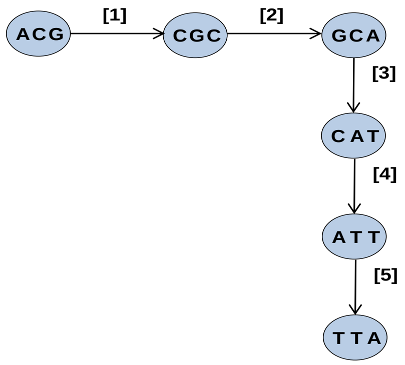
Figure 20: Parallel modified euler path algorithm



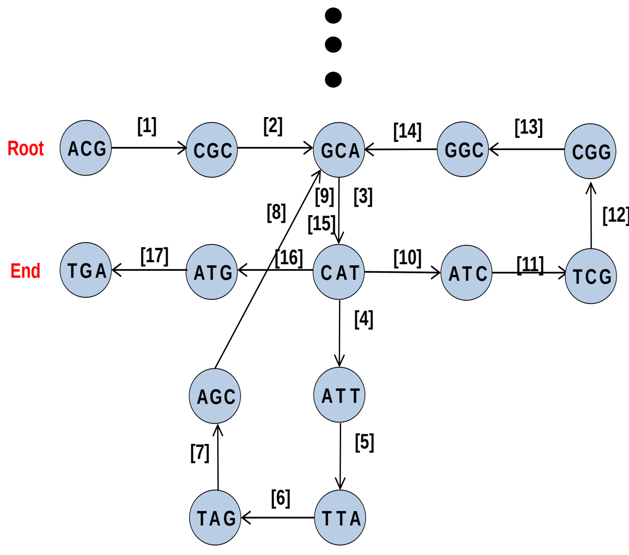
(a)The root is vertex ‘ACG’. Add vertex ‘CGC’ to the root and from path [1] between them.



(b) Add vertex ‘GCA’ and ‘CAT’ to travel paths [2]and[3].



(c) Add vertex ‘ATT’ and ‘TTA’ to travel paths [4]and[5].



(d) construction of the modified Euler path.

Figure 21: Example of the modified Euler path construction

VI. Complexity Analysis

(1) The time complexity of **Algorithm I** to construct the De Bruijn graph is estimated in $O(n^3)$ polynomial bound.
(2) The time complexity of **Algorithm II** to append stickers is estimated in $O(3n\text{-}1)$ polynomial bound.
(3) The time complexity of **Algorithm III** for parallel modified euler path in De Bruijn is estimated in $O(n^3)$ polynomial bound.
(4) The time complexity of total algorithm is in $O(n^3)$ polynomial bound.

VII. Conclusion

Taking into account efficiency and accuracy, this paper uses a fast bioinformatics method to construct a De Burijn graph along with a modified Euler path in order to solve a cancer genome assembly problem. The research utilizes parallelism to make executions with greater efficiency. We hope that this research demonstrates that bioinformatics computing is a technology worth pursuing, and should thus attract more scholars into this domain.

References

[1] Venturi, M; Hambly, RJ; Glinghammar, B; Rafter, JJ; Rowland, IR. *Genotoxic activity in human faecal water and the role of bile acids: a study using the alkaline comet assay*. Carcinogenesis, 1997, 18, 2353-2359.
[2]Pevzner P. A., Tang Haixu. *Fragment Assembly with Double-Barreled Data*. Bioinformatics，2001，17 (1):225—233.
[3]Pevzner P.,*1-Tuple DNA Sequencing：Computer Analysis*. Journal of Bimolecular Structure and Dynamics, 1989, 7(1):63-73.
[4] L. Euler, “"Solutio problematis ad geometriam situs pertinentis,” *Comment. Academiae Sci. I. Petropolitanae*, 128–140, 1736.
[5]Algorithms for de novo short read assembly using De Bruijn graphsDaniel R. Zerbino and Ewan Birney