World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:4, No:12, 2010

A Simulation Software for DNA Computing Algorithms Implementation

M. S. Muhammad, S. M. W. Masra, K. Kipli, and N. Zamhari

***Abstract*—**The capturing of gel electrophoresis image represents the output of a DNA computing algorithm. Before this image is being captured, DNA computing involves parallel overlap assembly (POA) and polymerase chain reaction (PCR) that is the main of this computing algorithm. However, the design of the DNA oligonucleotides to represent a problem is quite complicated and is prone to errors. In order to reduce these errors during the design stage before the actual in-vitro experiment is carried out; a simulation software capable of simulating the POA and PCR processes is developed. This simulation software capability is unlimited where problem of any size and complexity can be simulated, thus saving cost due to possible errors during the design process. Information regarding the DNA sequence during the computing process as well as the computing output can be extracted at the same time using the simulation software.

***Keywords*—** DNA computing, PCR, POA, simulation software

I. Introduction

EVER since Leonard M. Adleman [1] demonstrated the ability of using molecules of Deoxyribonucleic Acid or DNA as a medium for computation to solve a directed Hamiltonian Path Problem (HPP), the interest in applying DNA to solve similar computational problems have increased tremendously [2, 3, 4]. An in vitro experimental work based on DNA computing approach to solve an engineering scheduling problem in the case of an elevator travel path optimization for a typical building of N floors with M elevators has been presented [5].

One of the main problems during the design process of DNA computing is the synthesizing of DNA oligonucleotides to represent both the input and output of the problem. A mechanism for implementing the DNA computing approach for a much larger and complex problem is needed. The main aim of this research is therefore to develop a simulation software capable of simulating the DNA computing process that can verify the expected result before the actual in vitro experiment is carried out. Since the complexity and costs of the DNA oligonucleotides increases for larger and complex problem, this simulation software will provide a helpful guide for the DNA computing implementation as to eliminate errors during the design process. Information regarding the DNA oligonucleotides sequences for both the input and output could also be extracted from the simulation programme. Before the simulation software is presented, an overview of the elevator scheduling problem and its DNA computing solution design is first explained.

M. S. Muhammad is with the Department of Electronics Engineering, Universiti Malaysia Sarawak, MALAYSIA (phone: 082-583356; fax: 082-583410; e-mail: msaufee@ feng.unimas.my).

S. M. W. Masra, K. Kipli and N. Zamhari are with the Department of Electronics Engineering, Universiti Malaysia Sarawak, MALAYSIA, (e-mail: wmmasnia@feng.unimas.my, kkuryati@feng.unimas.my and znurdiani@feng.unimas.my).

II. Overview Of An Elevator Scheduling Problem

The elevator positions at an instance of a time for a 6 floors building with 2 elevators can be illustrated as in Table I. If the position of each elevator at floor 1, 2, 3, 4, 5, and 6 are represented as nodes V_1, V_2, V_3, V_4, V_5, and V_6 respectively, each of the elevator travel path combinations can thus be represented as a weighted graph. At the same time, the graph edges thus represent the elevator's travel path between floors.

TABLE I
Elevator Positions at an Instance of a Time

Floor No	Elevator A	Elevator B	Hall Call
6		(3 , 2)	
5			
4			↑
3			↓
2			
1	(3 , 5)		

Each of the graph edge weights can thus be mathematically formulated as directly proportional to the elevators travelling time between any floor using

$$\omega_{|j-i|} = (|j - i|)\, T_C + T_S \qquad (1)$$

where

i = elevator's present floor position
j = elevator's destination floor position
$|j - i|$ = total number of floors of elevator's movement
T_C = elevator's travelling time between two consecutive floors
T_S = elevator's stopping time at a floor

The output of the graph, given by sum of the graph weights thus represents the total travelling time of the elevator, i.e.

$$G(E_X) = \sum_{|j-i|=1}^{N-1} \omega_{|j-i|} \quad (2)$$

where $G(E_X)$ is the graph output for elevator X.

The total travelling time of both the elevators can now be calculated by summing up each of the elevator's travelling time, i.e.

$$G(E_T) = G(E_A) + G(E_B) \quad (3)$$

The minimum total travelling time of both elevators thus gives the optimal elevator travel path, i.e.

$$\text{Optimal Travel Path} = G(E_T)_{min} \quad (4)$$

Since the building is 6 floors high, the maximum number of floors that the elevator can travel is (6 – 1) = 5 floors. Now, if we assume that T_C = 5 s, T_S = 15 s, and representing 5 s of time with 10 units, we have from (1)

$$\omega_1 = 40 \quad , \omega_2 = 50, \omega_3 = 60, \omega_4 = 70, \omega_5 = 80$$

A weighted graph representing all possible travel path combinations of elevators A and B with either elevator answering one or both of the hall calls can now be constructed as shown in Fig. 1. Note that all possible end paths of elevator A are joined with the start paths of elevator B. This is done in order that the total output of the graph G (A, B) representing the travel path combinations of the elevators can be calculated.

Since there are two hall calls with two available elevators, it is clearly seen that there are 2^2 = 4 possible travel path combinations for both elevators as tabulated in Table II. The required solution for the elevator scheduling problem is thus the optimal path weight $G(A, B)_3$ = 230 = 115 s.

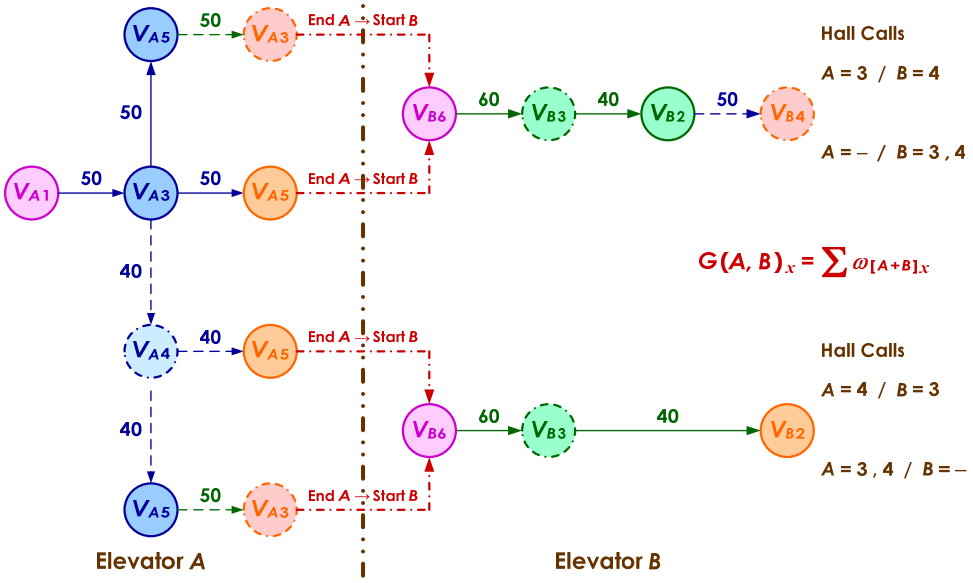

Fig. 1 Weighted graph showing all travel path combinations

TABLE II
TOTAL GRAPH OUTPUT OF ALL TRAVEL PATH COMBINATIONS

Hall Calls	Elevator travel path combinations	Total graph output
A = 3 B = 4	$V_{A1} \rightarrow V_{A3} \rightarrow V_{A5} \rightarrow V_{A3} \rightarrow$ $V_{B6} \rightarrow V_{B3} \rightarrow V_{B2} \rightarrow V_{B4}$	$G(A, B)_1$ = 150 + 150 = **300**
A = – B = 3 , 4	$V_{A1} \rightarrow V_{A3} \rightarrow V_{A5} \rightarrow$ $V_{B6} \rightarrow V_{B3} \rightarrow V_{B2} \rightarrow V_{B4}$	$G(A, B)_2$ = 100 + 150 = **250**
A = 4 B = 3	$V_{A1} \rightarrow V_{A3} \rightarrow V_{A4} \rightarrow V_{A5} \rightarrow$ $V_{B6} \rightarrow V_{B3} \rightarrow V_{B2}$	$G(A, B)_3$ = 130 + 100 = **230**
A = 3 , 4 B = –	$V_{A1} \rightarrow V_{A3} \rightarrow V_{A4} \rightarrow V_{A5} \rightarrow V_{A3} \rightarrow$ $V_{B6} \rightarrow V_{B3} \rightarrow V_{B2}$	$G(A, B)_3$ = 180 + 100 = **280**

III. DNA Computing Algorithm Design

In order to solve the elevator scheduling problem using DNA computing approach, the weighted graph of Fig. 1 is first transformed to represent the start, intermediate and end nodes, and also to differentiate the nodes of different travel path combinations as depicted in Fig. 2. The nodes are then assigned with a specific DNA sequence [6]. All the possible travel path combinations of the elevator are then synthesized so that the DNA oligonucleotides sequence length will directly represent the weight between the nodes as tabulated in Table III. Parallel overlap assembly (POA) [7] is then employed for initial pool generation to generate all the possible travel path combinations, and polymerase chain reaction (PCR) [8, 9] for the amplification of the required optimal path. Finally, gel electrophoresis [10] is performed to separate all the possible travel path combinations according to its length, and the image is captured where the DNA duplex representing the shortest path could be visualized representing the required optimal path solution of the problem as illustrated in Fig. 3. The PCR gel image shows 4 bands indicating all the 4 possible travel paths, i.e. $G(A, B)_3$ = 230bp, $G(A, B)_2$ = 250bp, $G(A, B)_4$ = 280bp and $G(A, B)_1$ = 300bp. This confirms the expected result that the optimal elevator's travel path is given by $G(A, B)_3$ = 230bp = 115 s.

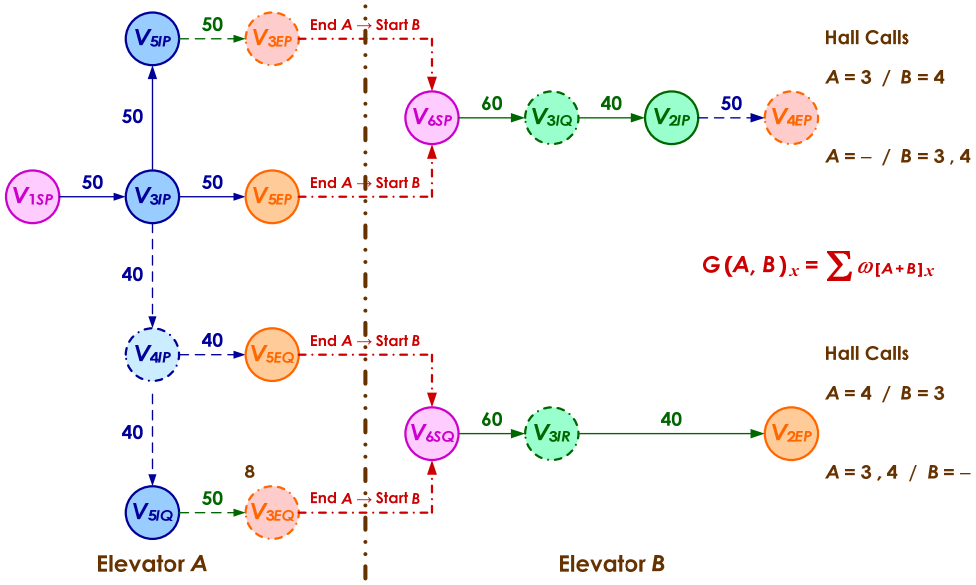

Fig. 2 Weighted graph for DNA computing approach showing different node locations and travel paths

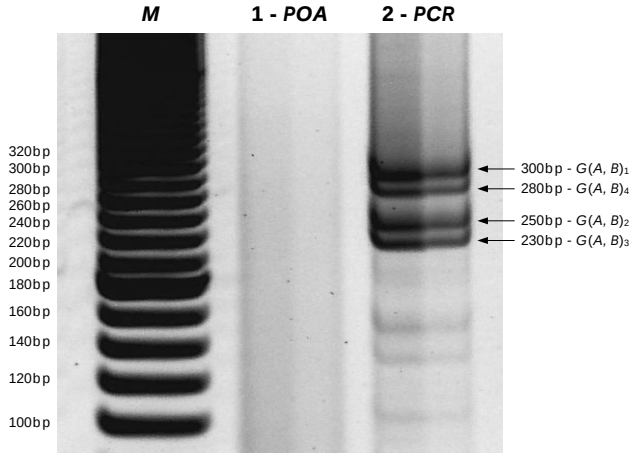

Fig. 3 Gel electrophoresis image showing computing output

TABLE III
SYNTHESIZED DNA OLIGONUCLEOTIDES FOR ALL ELEVATOR TRAVEL PATH COMBINATIONS

Edges	Sequences (5′ – 3′)
$V_{1SP} \rightarrow V_{3IP}$	CGGCGGTCCACTAAATACTAaggtcgtttaaggaagtacgCACTCTTTGTGAACGCCTTC
$V_{3IP} \rightarrow V_{5IP}$	CACTCTTTGTGAACGCCTTCccgtcggttaagcaagtaatgtactatgctTGAACCGGCCCTTTATATCT
$V_{3IP} \rightarrow V_{5EP}$	CACTCTTTGTGAACGCCTTCgcgtcgcttaccgaagcacgCTATAAGGCCAAAGCAGTCG
$V_{3IP} \rightarrow V_{4IP}$	CACTCTTTGTGAACGCCTTCacgtcgtgtaacgaagtcctGTGGGTTAGAGGTAGTCCGG
$V_{5IP} \rightarrow V_{3EP}$	TGAACCGGCCCTTTATATCTacgtgttttacccaagtcagTCATTCGAGTTATTCCTGGG
$V_{3EP} \rightarrow V_{6SP}$	TCATTCGAGTTATTCCTGGGGGACCTGCATCATACCAGTT
$V_{5EP} \rightarrow V_{6SP}$	CTATAAGGCCAAAGCAGTCGGGACCTGCATCATACCAGTT
$V_{4IP} \rightarrow V_{5IQ}$	GTGGGTTAGAGGTAGTCCGGcgctcgttgaagccagtaccCCGCTGATCCTTGCTAAGTA
$V_{4IP} \rightarrow V_{5EQ}$	GTGGGTTAGAGGTAGTCCGGgcgtcttttaATGCCTGGCTAAAGTGAGAC
$V_{5IQ} \rightarrow V_{3EQ}$	CCGCTGATCCTTGCTAAGTAgcggcgtgtcacgaactacgAAATGACCTTTTTAACGGCA
$V_{3EQ} \rightarrow V_{6SQ}$	AAATGACCTTTTTAACGGCATGCACGCAAAACTATTTCAT
$V_{5EQ} \rightarrow V_{6SQ}$	ATGCCTGGCTAAAGTGAGACTGCACGCAAAACTATTTCAT
$V_{6SP} \rightarrow V_{3IQ}$	GGACCTGCATCATACCAGTTacgtggtttaaggaagtacggtactatgctAAGCAATGTGGTTGTAGGGA
$V_{3IQ} \rightarrow V_{2IP}$	AAGCAATGTGGTTGTAGGGAacgtcgctgcaagaactacgAAAGCCCGTCGGTTAAGTTA
$V_{2IP} \rightarrow V_{4EP}$	AAAGCCCGTCGGTTAAGTTAggtcttttaatcaactaatgGGAATCCATTGATCGCTTTA
$V_{6SQ} \rightarrow V_{3IR}$	TGCACGCAAAACTATTTCATccgtgggttaaagaagtcctgtactctcctTCTGCACTGTTAATGAGCCA
$V_{3IR} \rightarrow V_{2EP}$	TCTGCACTGTTAATGAGCCAacgtcttgtcCTAATTTTAGAAATGGCGCG

IV. DEVELOPMENT OF DNA SIMULATION SOFTWARE

The DNA computing implementation involves oligonucleotides sequences design to represent both the problem inputs and outputs. As the problem grows larger, the complexity of the oligonucleotides design gets complicated. In order to assist in the design, we developed a simple simulation programme that is able to simulate the expected DNA process during the computational stage.

The Microsoft Visual Basic platform software programme developed is able to simulate the POA and PCR physical processes of the computation. The flowchart of the designed software programme is shown in Fig. 4. It is a simple user friendly programme that allows the user to choose between the two processes as shown in Fig. 5.

Once the process is chosen, a new menu appears that guides the user to enter the input data, start the simulation process and save the output results. Note that for convenience, the weighted graph of Fig. 2 is relabelled as shown in Fig. 6.

The simulation result is stored in the MS Access database and can be manipulated using the MS Excel. The simulation results for the POA and PCR processes of the problem discussed is shown in Fig. 7 and Fig. 8 respectively. Here, it can be seen that after 4 cycles of POA, all the possible travel path combinations of the elevator are generated. The sequence and length of DNA oligonucleotides representing the travel time is also shown in the simulation output. This verifies the theoretical as well as the in vitro experimental results of the problem. Finally, for the PCR simulation process, as expected, after 2 cycles of PCR process, $2^2 = 4$ sequences are replicated that will represent the DNA computational output of the problem.

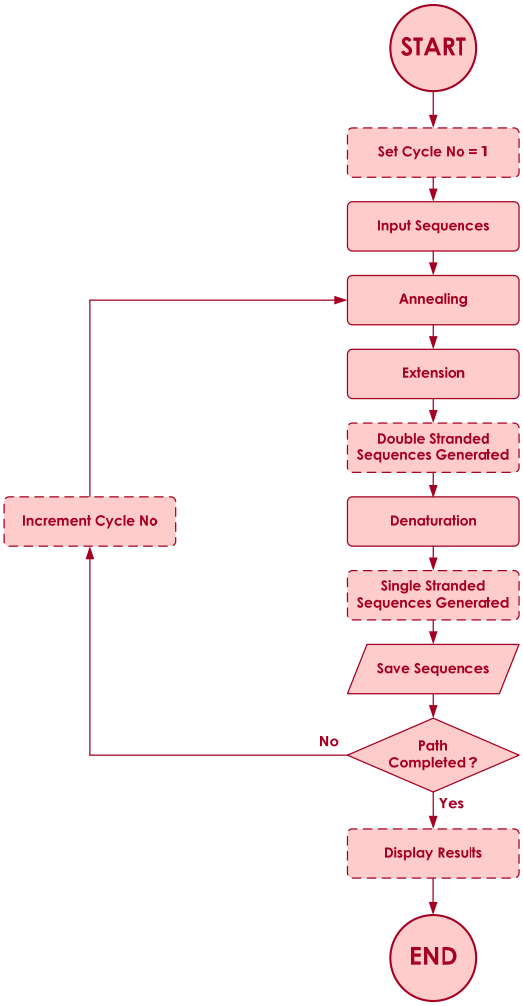

Fig. 4 Simulation software flowchart

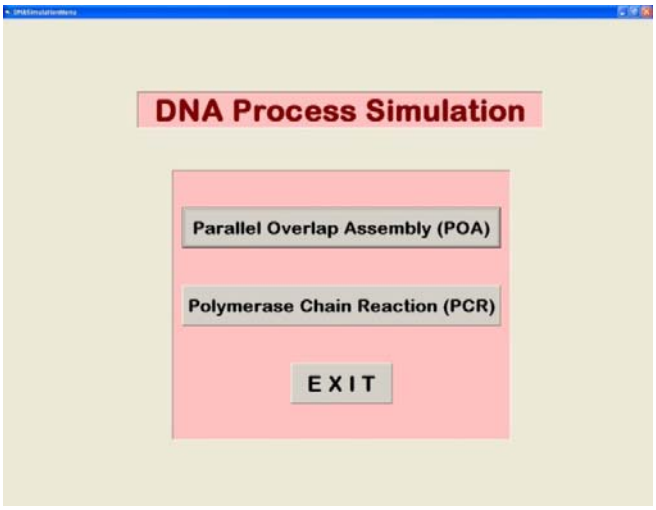

Fig. 5 Simulation software process menu

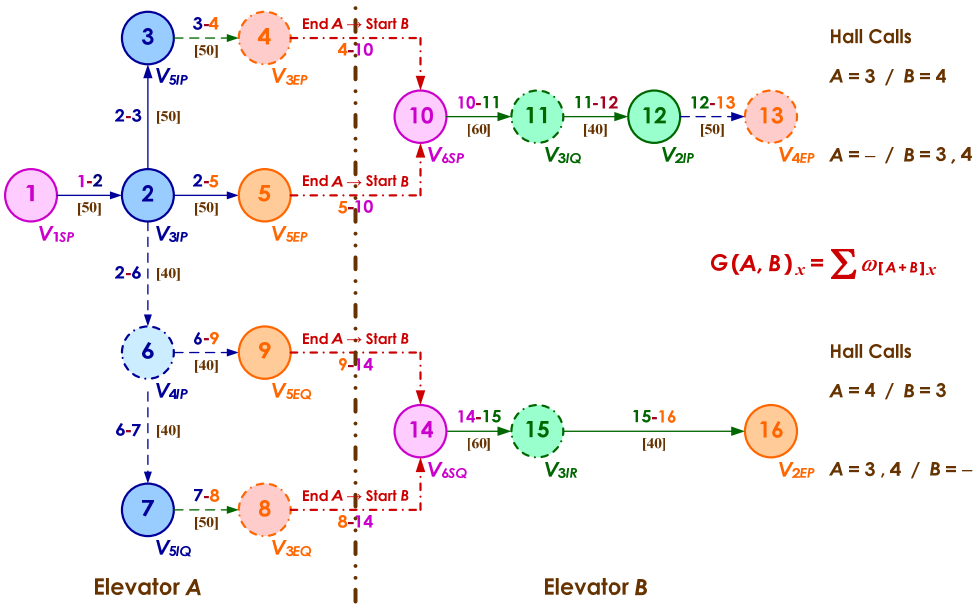

Fig. 6 Relabeled weighted graph for simulation software

Path	DNA Sequence	Cycle #	Length	
1-2	CGGCGGTCCACTAAATACTAaggtcgtttaaggaagtacgCACTCTTTGTGAACGCCTTC	1	60	
2-3	CACTCTTTGTGAACGCCTTCccgtcggttaagcaagtaatgtactatgctTGAACCGGCCCTTTATATCT	1	70	
2-5	CACTCTTTGTGAACGCCTTCgcgtcgcttaccgaagcacgCTATAAGGCCAAAGCAGTCG	1	60	
2-6	CACTCTTTGTGAACGCCTTCacgtcgtgtaacgaagtcctGTGGGTTAGAGGTAGTCCGG	1	60	
3-4	TGAACCGGCCCTTTATATCTacgtgttttacccaagtcagTCATTCGAGTTATTCCTGGG	1	60	
4-10	TCATTCGAGTTATTCCTGGGGGACCTGCATCATACCAGTT	1	40	
5-10	CTATAAGGCCAAAGCAGTCGGGACCTGCATCATACCAGTT	1	40	
6-7	GTGGGTTAGAGGTAGTCCGGcgctcgttgaagccagtaccCCGCTGATCCTTGCTAAGTA	1	60	
6-9	GTGGGTTAGAGGTAGTCCGGgcgtcttttaATGCCTGGCTAAAGTGAGAC	1	50	
7-8	CCGCTGATCCTTGCTAAGTAgcggcgtgtcacgaactacgAAATGACCTTTTTAACGGCA	1	60	
8-14	AAATGACCTTTTTAACGGCATGCACGCAAAACTATTTCAT	1	40	
9-14	ATGCCTGGCTAAAGTGAGACTGCACGCAAAACTATTTCAT	1	40	
10-11	GGACCTGCATCATACCAGTTacgtggtttaaggaagtacggtactatgctAAGCAATGTGGTTGTAGGGA	1	70	
11-12	AAGCAATGTGGTTGTAGGGAacgtcgctgcaagaactacgAAAGCCCGTCGGTTAAGTTA	1	60	
12-13	AAAGCCCGTCGGTTAAGTTAggtcttttaatcaactaatgGGAATCCATTGATCGCTTTA	1	60	
14-15	TGCACGCAAAACTATTTCATccgtgggttaaagaagtcctgtactctcctTCTGCACTGTTAATGAGCCA	1	70	
15-16	TCTGCACTGTTAATGAGCCAacgtcttgtcCTAATTTTAGAAATGGCGCG	1	50	
:				
:				
Path	**DNA Sequence**	**Cycle #**	**Length**	
1-2-3	CGGCGGTCCACTAAATACTAaggtcgttta . . . gtactatgctTGAACCGGCCCTTTATATCT	2	110	
1-2-5	CGGCGGTCCACTAAATACTAaggtcgttta . . . ccgaagcacgCTATAAGGCCAAAGCAGTCG	2	100	
1-2-6	CGGCGGTCCACTAAATACTAaggtcgttta . . . acgaagtcctGTGGGTTAGAGGTAGTCCGG	2	100	
:				
10-11-12	GGACCTGCATCATACCAGTTacgtggttta . . . aagaactacgAAAGCCCGTCGGTTAAGTTA	2	110	
11-12-13	AAGCAATGTGGTTGTAGGGAacgtcgctgc . . . tcaactaatgGGAATCCATTGATCGCTTTA	2	100	
14-15-16	TGCACGCAAAACTATTTCATccgtgggtta . . . acgtcttgtcCTAATTTTAGAAATGGCGCG	2	100	
:				
:				
Path	**DNA Sequence**	**Cycle #**	**Length**	
1-2-3-4	CGGCGGTCCACTAAATACTAaggtcgttta . . . cccaagtcagTCATTCGAGTTATTCCTGGG	3	150	
1-2-5-10	CGGCGGTCCACTAAATACTAaggtcgttta . . . AAAGCAGTCGGGACCTGCATCATACCAGTT	3	120	
1-2-6-7	CGGCGGTCCACTAAATACTAaggtcgttta . . . agccagtaccCCGCTGATCCTTGCTAAGTA	3	140	
1-2-6-9	CGGCGGTCCACTAAATACTAaggtcgttta . . . gcgtcttttaATGCCTGGCTAAAGTGAGAC	3	130	
:				
9-14-15-16	ATGCCTGGCTAAAGTGAGACTGCACGCAAA . . . acgtcttgtcCTAATTTTAGAAATGGCGCG	3	120	
10-11-12-13	GGACCTGCATCATACCAGTTacgtggttta . . . tcaactaatgGGAATCCATTGATCGCTTTA	3	150	
4-10-11-12-13	TCATTCGAGTTATTCCTGGGGGACCTGCAT . . . tcaactaatgGGAATCCATTGATCGCTTTA	3	170	
7-8-14-15-16	CCGCTGATCCTTGCTAAGTAgcggcgtgtc . . . acgtcttgtcCTAATTTTAGAAATGGCGCG	3	160	
:				
:				
Path	**DNA Sequence**	**Cycle #**	**Length**	
1-2-3-4-10-11	CGGCGGTCCACTAAATACTAaggtcgttta . . . gtactatgctAAGCAATGTGGTTGTAGGGA	4	220	
1-2-5-10-11-12	CGGCGGTCCACTAAATACTAaggtcgttta . . . aagaactacgAAAGCCCGTCGGTTAAGTTA	4	210	
1-2-6-7-8-14	CGGCGGTCCACTAAATACTAaggtcgttta . . . TTTAACGGCATGCACGCAAAACTATTTCAT	4	200	
1-2-6-9-14-15	CGGCGGTCCACTAAATACTAaggtcgttta . . . gtactctcctTCTGCACTGTTAATGAGCCA	4	200	
2-3-4-10-11-12	CACTCTTTGTGAACGCCTTCccgtcggtta . . . aagaactacgAAAGCCCGTCGGTTAAGTTA	4	220	
2-5-10-11-12-13	CACTCTTTGTGAACGCCTTCgcgtcgctta . . . tcaactaatgGGAATCCATTGATCGCTTTA	4	210	
2-6-7-8-14-15	CACTCTTTGTGAACGCCTTCacgtcgtgta . . . gtactctcctTCTGCACTGTTAATGAGCCA	4	210	
2-6-9-14-15-16	CACTCTTTGTGAACGCCTTCacgtcgtgta . . . acgtcttgtcCTAATTTTAGAAATGGCGCG	4	190	
3-4-10-11-12-13	TGAACCGGCCCTTTATATCTacgtgttttta . . . tcaactaatgGGAATCCATTGATCGCTTTA	4	210	
6-7-8-14-15-16	GTGGGTTAGAGGTAGTCCGGcgctcgttga . . . acgtcttgtcCTAATTTTAGAAATGGCGCG	4	200	
1-2-3-4-10-11-12	CGGCGGTCCACTAAATACTAaggtcgttta . . . aagaactacgAAAGCCCGTCGGTTAAGTTA	4	260	
1-2-5-10-11-12-13	CGGCGGTCCACTAAATACTAaggtcgttta . . . tcaactaatgGGAATCCATTGATCGCTTTA	4	250	$G(A,B)_2$
1-2-6-7-8-14-15	CGGCGGTCCACTAAATACTAaggtcgttta . . . gtactctcctTCTGCACTGTTAATGAGCCA	4	250	
1-2-6-9-14-15-16	CGGCGGTCCACTAAATACTAaggtcgttta . . . acgtcttgtcCTAATTTTAGAAATGGCGCG	4	230	$G(A,B)_3$
2-3-4-10-11-12-13	CACTCTTTGTGAACGCCTTCccgtcggttaa . . . tcaactaatgGGAATCCATTGATCGCTTTA	4	260	
2-6-7-8-14-15-16	CACTCTTTGTGAACGCCTTCacgtcgtgta . . . acgtcttgtcCTAATTTTAGAAATGGCGCG	4	240	
1-2-3-4-10-11-12-13	CGGCGGTCCACTAAATACTAaggtcgttta . . . tcaactaatgGGAATCCATTGATCGCTTTA	4	300	$G(A,B)_1$
1-2-6-7-8-14-15-16	CGGCGGTCCACTAAATACTAaggtcgttta . . . acgtcttgtcCTAATTTTAGAAATGGCGCG	4	280	$G(A,B)_4$

Fig. 7 POA process simulation showing all the possible travel path combinations

Path	DNA Sequence	Cycle #	Length
1-2-5-10-11-12-13	CGGCGGTCCACTAAATACTAaggtcgtttaaggaagtacg ... ggtcttttaatcaactaatgGGAATCCATTGATCGCTTTA	1	250
[1-2-5-10-11-12-13]'	*GCCGCCAGGTGATTTATGATtccagcaaattccttcatgc ... ccagaaaattagttgattacCCTTAGGTAACTAGCGAAAT*	*1*	*250*
1-2-5-10-11-12-13	CGGCGGTCCACTAAATACTAaggtcgtttaaggaagtacg ... ggtcttttaatcaactaatgGGAATCCATTGATCGCTTTA	1	250
[1-2-5-10-11-12-13]'	*GCCGCCAGGTGATTTATGATtccagcaaattccttcatgc ... ccagaaaattagttgattacCCTTAGGTAACTAGCGAAAT*	*1*	*250*
1-2-6-9-14-15-16	CGGCGGTCCACTAAATACTAaggtcgtttaaggaagtacg ... TAATGAGCCAacgtcttgtcCTAATTTTAGAAATGGCGCG	1	230
[1-2-6-9-14-15-16]'	*GCCGCCAGGTGATTTATGATtccagcaaattccttcatgc ... ATTACTCGGTtgcagaacagGATTAAAATCTTTACCGCGC*	*1*	*230*
1-2-6-9-14-15-16	CGGCGGTCCACTAAATACTAaggtcgtttaaggaagtacg ... TAATGAGCCAacgtcttgtcCTAATTTTAGAAATGGCGCG	1	230
[1-2-6-9-14-15-16]'	*GCCGCCAGGTGATTTATGATtccagcaaattccttcatgc ... ATTACTCGGTtgcagaacagGATTAAAATCTTTACCGCGC*	*1*	*230*
1-2-3-4-10-11-12-13	CGGCGGTCCACTAAATACTAaggtcgtttaaggaagtacg ... ggtcttttaatcaactaatgGGAATCCATTGATCGCTTTA	1	300
[1-2-3-4-10-11-12-13]'	*GCCGCCAGGTGATTTATGATtccagcaaattccttcatgc ... ccagaaaattagttgattacCCTTAGGTAACTAGCGAAAT*	*1*	*300*
1-2-3-4-10-11-12-13	CGGCGGTCCACTAAATACTAaggtcgtttaaggaagtacg ... ggtcttttaatcaactaatgGGAATCCATTGATCGCTTTA	1	300
[1-2-3-4-10-11-12-13]'	*GCCGCCAGGTGATTTATGATtccagcaaattccttcatgc ... ccagaaaattagttgattacCCTTAGGTAACTAGCGAAAT*	*1*	*300*
1-2-6-7-8-14-15-16	CGGCGGTCCACTAAATACTAaggtcgtttaaggaagtacg ... TAATGAGCCAacgtcttgtcCTAATTTTAGAAATGGCGCG	1	280
[1-2-6-7-8-14-15-16]'	*GCCGCCAGGTGATTTATGATtccagcaaattccttcatgc ... ATTACTCGGTtgcagaacagGATTAAAATCTTTACCGCGC*	*1*	*280*
1-2-6-7-8-14-15-16	CGGCGGTCCACTAAATACTAaggtcgtttaaggaagtacg ... TAATGAGCCAacgtcttgtcCTAATTTTAGAAATGGCGCG	1	280
[1-2-6-7-8-14-15-16]'	*GCCGCCAGGTGATTTATGATtccagcaaattccttcatgc ... ATTACTCGGTtgcagaacagGATTAAAATCTTTACCGCGC*	*1*	*280*
	:		
1-2-5-10-11-12-13	CGGCGGTCCACTAAATACTAaggtcgtttaaggaagtacg ... ggtcttttaatcaactaatgGGAATCCATTGATCGCTTTA	2	250
[1-2-5-10-11-12-13]'	*GCCGCCAGGTGATTTATGATtccagcaaattccttcatgc ... ccagaaaattagttgattacCCTTAGGTAACTAGCGAAAT*	*2*	*250*
1-2-5-10-11-12-13	CGGCGGTCCACTAAATACTAaggtcgtttaaggaagtacg ... ggtcttttaatcaactaatgGGAATCCATTGATCGCTTTA	2	250
[1-2-5-10-11-12-13]'	*GCCGCCAGGTGATTTATGATtccagcaaattccttcatgc ... ccagaaaattagttgattacCCTTAGGTAACTAGCGAAAT*	*2*	*250*
1-2-5-10-11-12-13	CGGCGGTCCACTAAATACTAaggtcgtttaaggaagtacg ... ggtcttttaatcaactaatgGGAATCCATTGATCGCTTTA	2	250
[1-2-5-10-11-12-13]'	*GCCGCCAGGTGATTTATGATtccagcaaattccttcatgc ... ccagaaaattagttgattacCCTTAGGTAACTAGCGAAAT*	*2*	*250*
1-2-5-10-11-12-13	CGGCGGTCCACTAAATACTAaggtcgtttaaggaagtacg ... ggtcttttaatcaactaatgGGAATCCATTGATCGCTTTA	2	250
[1-2-5-10-11-12-13]'	*GCCGCCAGGTGATTTATGATtccagcaaattccttcatgc ... ccagaaaattagttgattacCCTTAGGTAACTAGCGAAAT*	*2*	*250*
1-2-6-9-14-15-16	CGGCGGTCCACTAAATACTAaggtcgtttaaggaagtacg ... TAATGAGCCAacgtcttgtcCTAATTTTAGAAATGGCGCG	2	230
[1-2-6-9-14-15-16]'	*GCCGCCAGGTGATTTATGATtccagcaaattccttcatgc ... ATTACTCGGTtgcagaacagGATTAAAATCTTTACCGCGC*	*2*	*230*
1-2-6-9-14-15-16	CGGCGGTCCACTAAATACTAaggtcgtttaaggaagtacg ... TAATGAGCCAacgtcttgtcCTAATTTTAGAAATGGCGCG	2	230
[1-2-6-9-14-15-16]'	*GCCGCCAGGTGATTTATGATtccagcaaattccttcatgc ... ATTACTCGGTtgcagaacagGATTAAAATCTTTACCGCGC*	*2*	*230*
1-2-6-9-14-15-16	CGGCGGTCCACTAAATACTAaggtcgtttaaggaagtacg ... TAATGAGCCAacgtcttgtcCTAATTTTAGAAATGGCGCG	2	230
[1-2-6-9-14-15-16]'	*GCCGCCAGGTGATTTATGATtccagcaaattccttcatgc ... ATTACTCGGTtgcagaacagGATTAAAATCTTTACCGCGC*	*2*	*230*
1-2-6-9-14-15-16	CGGCGGTCCACTAAATACTAaggtcgtttaaggaagtacg ... TAATGAGCCAacgtcttgtcCTAATTTTAGAAATGGCGCG	2	230
[1-2-6-9-14-15-16]'	*GCCGCCAGGTGATTTATGATtccagcaaattccttcatgc ... ATTACTCGGTtgcagaacagGATTAAAATCTTTACCGCGC*	*2*	*230*
1-2-3-4-10-11-12-13	CGGCGGTCCACTAAATACTAaggtcgtttaaggaagtacg ... ggtcttttaatcaactaatgGGAATCCATTGATCGCTTTA	2	300
[1-2-3-4-10-11-12-13]'	*GCCGCCAGGTGATTTATGATtccagcaaattccttcatgc ... ccagaaaattagttgattacCCTTAGGTAACTAGCGAAAT*	*2*	*300*
1-2-3-4-10-11-12-13	CGGCGGTCCACTAAATACTAaggtcgtttaaggaagtacg ... ggtcttttaatcaactaatgGGAATCCATTGATCGCTTTA	2	300
[1-2-3-4-10-11-12-13]'	*GCCGCCAGGTGATTTATGATtccagcaaattccttcatgc ... ccagaaaattagttgattacCCTTAGGTAACTAGCGAAAT*	*2*	*300*
1-2-3-4-10-11-12-13	CGGCGGTCCACTAAATACTAaggtcgtttaaggaagtacg ... ggtcttttaatcaactaatgGGAATCCATTGATCGCTTTA	2	300
[1-2-3-4-10-11-12-13]'	*GCCGCCAGGTGATTTATGATtccagcaaattccttcatgc ... ccagaaaattagttgattacCCTTAGGTAACTAGCGAAAT*	*2*	*300*
1-2-3-4-10-11-12-13	CGGCGGTCCACTAAATACTAaggtcgtttaaggaagtacg ... ggtcttttaatcaactaatgGGAATCCATTGATCGCTTTA	2	300
[1-2-3-4-10-11-12-13]'	*GCCGCCAGGTGATTTATGATtccagcaaattccttcatgc ... ccagaaaattagttgattacCCTTAGGTAACTAGCGAAAT*	*2*	*300*
1-2-6-7-8-14-15-16	CGGCGGTCCACTAAATACTAaggtcgtttaaggaagtacg ... TAATGAGCCAacgtcttgtcCTAATTTTAGAAATGGCGCG	2	280
[1-2-6-7-8-14-15-16]'	*GCCGCCAGGTGATTTATGATtccagcaaattccttcatgc ... ATTACTCGGTtgcagaacagGATTAAAATCTTTACCGCGC*	*2*	*280*
1-2-6-7-8-14-15-16	CGGCGGTCCACTAAATACTAaggtcgtttaaggaagtacg ... TAATGAGCCAacgtcttgtcCTAATTTTAGAAATGGCGCG	2	280
[1-2-6-7-8-14-15-16]'	*GCCGCCAGGTGATTTATGATtccagcaaattccttcatgc ... ATTACTCGGTtgcagaacagGATTAAAATCTTTACCGCGC*	*2*	*280*
1-2-6-7-8-14-15-16	CGGCGGTCCACTAAATACTAaggtcgtttaaggaagtacg ... TAATGAGCCAacgtcttgtcCTAATTTTAGAAATGGCGCG	2	280
[1-2-6-7-8-14-15-16]'	*GCCGCCAGGTGATTTATGATtccagcaaattccttcatgc ... ATTACTCGGTtgcagaacagGATTAAAATCTTTACCGCGC*	*2*	*280*
1-2-6-7-8-14-15-16	CGGCGGTCCACTAAATACTAaggtcgtttaaggaagtacg ... TAATGAGCCAacgtcttgtcCTAATTTTAGAAATGGCGCG	2	280
[1-2-6-7-8-14-15-16]'	*GCCGCCAGGTGATTTATGATtccagcaaattccttcatgc ... ATTACTCGGTtgcagaacagGATTAAAATCTTTACCGCGC*	*2*	*280*

Fig. 8 PCR process simulation of travel path combinations for 2 cycles

V. Conclusion

This paper presents and discusses an elevator scheduling optimization problem solution using DNA computing algorithm. The expected computation output result is verified by the in vitro experimental that has been carried out. In order to assist in designing and synthesizing the DNA oligonucleotides for a larger and complex problem, a simulation programme capable of simulating the POA and PCR physical processes has been developed. The applicability and feasibility of the DNA computing approach could therefore be extended into many more complex problems of this type of nature with this successful DNA computing design, in vitro experimental implementation, and simulation software,.

Acknowledgment

The authors would like to thank Universiti Malaysia Sarawak (UNIMAS) for providing financial support to attend the conference and present the paper.

REFERENCES

[1] L.M. Adleman, "Molecular computation of solutions to combinatorial problems," Science, 266, pp 1021-1024 (1994).
[2] Z. Ibrahim, Y. Tsuboi, O. Ono and M. Khalid, "Direct-proportional length-based DNA computing for shortest path problem," International Journal of Computer Science and Applications, 1(1), pp 46-60 (2004).
[3] J.Y. Lee, S.Y. Shin, S.J. Augh, T.H. Park and B.T. Zhang, "Temperature gradient-based DNA computing for graph problems with weighted edges," Lecture Notes in Computer Science, 2568, pp 73-84 (2003).
[4] Y. Yamamoto, A. Kameda, N. Matsuura, T. Shiba, Y. Kawazoe and A. Ahochi, "Local search by concentration-controlled DNA computing," International Journal of Computational Intelligence and Applications, 2, pp 447-455 (2002).
[5] M.S. Muhammad, Z. Ibrahim, O. Ono and M. Khalid, "Direct-proportional length-based DNA computing implementation for elevator scheduling problem", Proceedings of the IEEE International Region 10 Conference (TENCON2005), Melbourne, pp 711-715 (2005).
[6] F. Udo, S. Sam, B. Wolfgang and R. Hilmar, "DNA sequence generator: A program for the construction of DNA sequences," Proceedings of the Seventh International Workshop on DNA Based Computers, Florida, pp 23-32 (2001).
[7] P.D. Kaplan, Q. Ouyang, D.S. Thaler and A. Libchaber, "Parallel overlap assembly for the construction of computational DNA libraries," Journal of Theoretical Biology, 188(3), pp 333-341 (1997).
[8] J.Y. Lee, H.W. Lim, S.I. Yoo, B.T. Zhang and T.H. Park, "Efficient initial pool generation for weighted graph problems using parallel overlap assembly," Proceedings of the 10th International Meeting on DNA Computing, Milan, pp 357-364 (2004).
[9] J.P. Fitch, Engineering Introduction to Biotechnology, SPIE Press, Washington, (2001).
[10] G. Paun, G. Rozenberg and A. Salomaa, "DNA computing: new computing paradigms," Lecture Notes in Computer Science, 1644, pp 106-118 (1998).