Experimental Implementation of Direct-Proportional Length-Based DNA Computing for Elevator Scheduling Problem

Mohd Saufee Muhammad, Zuwairie Ibrahim, Satomi Ueda, Osamu Ono and Marzuki Khalid
Institute of Applied DNA Computing
Meiji University
1-1-1 Higashi-mita, Tama-ku
Kawasaki-shi, Kanagawa-ken
JAPAN 214-8571
{msaufee, zuwairie, satomixx, ono}@isc.meiji.ac.jp, marzuki@utmkl.utm.my

Abstract

Previously, ideas and implementation methods for solving elevator scheduling problem using DNA computing method had been proposed. In this paper, results of biochemical experiments that have been carried out to realize the computing approach are presented. Every possible elevators travel path combinations are encoded by DNA sequences of length directly proportional to the elevator's traveling time based on certain initial conditions such as elevators present and destination floors, and hall calls from a floor. Parallel overlap assembly is employed for an efficient initial pool generation of all possible travel path combinations and polymerase chain reaction for sequence amplification. Finally, gel electrophoresis is performed to separate the DNA sequence according to its length, and the shortest DNA sequence representing the elevator's optimal path can thus be visualized from the gel electrophoresis image. The experimental result shows that the DNA computing approach can be well-suited for solving such real-world problem of this type of nature.

1. Introduction

The practical possibility of using molecules of Deoxyribonucleic Acid or DNA as a medium for computation was first demonstrated in 1994 by Leonard M. Adleman [1]. Using the tools of biomolecular engineering, Adleman successfully solved a directed Hamiltonian Path Problem (HPP) in his experiment.

Instead of the traditional silicon-based computing technologies, DNA computing is a form of computing that uses DNA and molecular biology. Adleman's pioneering work set the new approach for this new field of bio-computing research. Computing with DNA generated a tremendous amount of excitement by offering a brand new paradigm for performing and viewing computations. Adleman's experiment [2] solved a simple instance of the Traveling Salesman Problem (TSP) by manipulating the DNA molecules. This marked the first solution of a mathematical problem with the tools of biology.

Computing with DNA offers many advantages over traditional silicon-based computing due to several reasons. These include massive parallelism and memory capacity. The primary advantage offered by most proposed models of DNA based computation is the ability to handle millions of operation in parallel. DNA computing can reach approximately 10^{20} operations per second compared to today's teraflop supercomputers. Certain operations in DNA computing (for example, hybridization – the bonding of two DNA strands to form the double helix) are over a billion times more energy efficient as compared to conventional computers. Also, DNA stores information at a density of about one bit per nm^3 – about a trillion times as efficiently as videotape.

DNA computation relies on devising algorithms to solve problems using the encoded information in the sequence of nucleotides that make up DNA's double helix strand, breaking and making new bonds between them to reach the answer. Each strand may be viewed as a chain of nucleotides, or bases. An *n*-letter sequence of consecutive bases is known as an *n*-mer or an oligonucleotide of length *n*. The four DNA nucleotides are adenine (A), guanine (G), cytosine (C) and thymine (T). Each strand has, according to chemical convention, a 5' and a 3' end, thus any single strand has a natural orientation. The classical double helix of DNA is formed when two separate strands bond together. Bonding occurs by the pairwise attraction of bases; A bonds with T and G bonds with C. The pairs (A, T) and (G, C) are known as Watson-Crick complementary base pairs [3].

Research on DNA computing approach to solve engineering related problems however has not been very well established. Since DNA computing is very suitable to solve combinatorial problems, an elevator scheduling problem is chosen as a benchmark to be solved using this computing technique. The elevator scheduling problem involves finding an optimal path, or in other words, finding the shortest elevator travel path of a building with certain number of elevators and floors. However, this is a complex combinatorial problem since certain criteria need to be fulfilled for the problem solution such as initial elevator position, its destinations and hall calls made for an elevator.

There are several research reports on DNA computing techniques for solving shortest path

problems of a weighted graph. Nayaranan and Zorbalas [4] proposed a constant proportional length-based DNA computing technique for TSP. Yamamoto *et al.* [5] proposed a concentration-controlled DNA computing to accomplish local search for solving shortest path problem. Lee *et al.* [6] proposed a DNA computing technique based on temperature gradient to solve the TSP problem. Ibrahim *et al.* [7] on the other hand proposed a direct-proportional length-based DNA computing for shortest path problem.

All the methods proposed for computing weighted graph have not been well applied to solve a real word problem. Previously, ideas and implementation methods for solving elevator scheduling problem using DNA computing method had been proposed [8]. In this paper, a direct-proportional length-based DNA computing for shortest path problem has been utilized to solve the elevator scheduling problem. Results of the biochemical experiments to realize the computing approach are presented. Using this technique, the elevator's traveled paths and traveling time are represented by DNA sequences of specific length. These DNA sequences are designed based on certain initial conditions such as elevator's present and destination floors, and hall calls for an elevator from a floor. Constraints such as node position in the graph, and initial pool generation method are investigated and discussed in detail for the successful implementation of the DNA computing method used.

2. Elevator Scheduling Problem

Consider a typical situation of a building with M elevators and N floors. The present elevator positions, its destinations and hall calls on each floor at a particular instance can be illustrated as in Table 1.

Table 1. Elevator situation at a particular instance

Floor no.	Elevator 1	Elevator 2	…	Elevator M–1	Elevator M	Hall calls
N		N–3, 3, 1				↓
N–1				7, 2		↑
:	:	:	:	:	:	:
3						↑
2	5, 6, N–1					↓
1					8, N–4, N	

These elevator travel paths can be represented using a weighted graph. The elevator positions at floor 1, 2, 3, … , $N-2$, $N-1$, N are represented with nodes V_1, V_2, V_3, … , V_{N-2}, V_{N-1}, V_N respectively, and the weight between every node can be represented as

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

where

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

The graph of all possible travel path combinations of one of the elevator can be constructed as shown in Fig. 1.

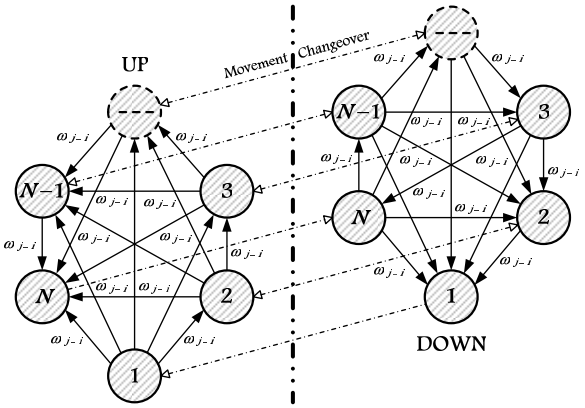

Fig. 1. Graph of all possible travel path combinations of an elevator

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

$$G(E_x) = \sum \omega_{|j-i|} \qquad \textbf{(2)}$$

For a building with M elevators, M similar graphs as shown in Fig. 1 can be duplicated representing all M elevators travel paths. The total traveling time of all the elevators can now be calculated by summing up each of the elevator's traveling time, i.e.

$$G(E_1, E_2, \cdots, E_{M-1}, E_M) = G(E_1) + G(E_2) + \cdots + G(E_{M-1}) + G(E_M) \qquad \textbf{(3)}$$

The minimum total traveling time of all the elevators with all initial conditions and requirements satisfied thus gives the optimal elevator travel path, i.e.

$$\text{Optimal Travel Path} = G(E_1, E_2, \ldots, E_{M-1}, E_M)_{min} \qquad \textbf{(4)}$$

3. Direct-proportional Length-based DNA Computing Solution

Let us now consider a building with 2 elevators and 7 floors. Elevator A is presently at 2nd floor and its destinations are 4th and 6th floors, while elevator B is presently at 7th floor and its destinations are 4th and 3rd floors. There is a hall call at 5th floor going up, and a hall call at 4th floor going down, as illustrated in Table 2.

Table 2. Elevator scheduling problem example

Floor no.	Elevator A	Elevator B	Hall call
7		(4 , 3)	
6			
5			↑
4			↓
3			
2	(4 , 6)		
1			

In order to solve the elevator scheduling problem using direct-proportional length-based DNA computing method for solving shortest path problem, several computing steps are performed that are discussed below.

Step 1. Represent the elevator position at a floor as nodes V_1, V_2, V_3, V_4, V_5, V_6 and V_7 for all the 7 floors in the building respectively.

Step 2. Assign weights between every node that will directly represent the elevator's traveling time between the floors. Since the building is 7 floors high, the maximum number of floors that the elevator can travel is $(7-1) = 6$ floors. Now, assume that $T_C = 5$ s, $T_S = 15$ sec, and representing 5 sec of time with 10 units we have using (1)

$$\omega_1 = 1(5) + 15 = 20 \text{ sec} = 40$$
$$\omega_2 = 2(5) + 15 = 25 \text{ sec} = 50$$
$$\omega_3 = 3(5) + 15 = 30 \text{ sec} = 60$$
$$\omega_4 = 4(5) + 15 = 35 \text{ sec} = 70$$
$$\omega_5 = 5(5) + 15 = 40 \text{ sec} = 80$$
$$\omega_6 = 6(5) + 15 = 45 \text{ sec} = 90$$

Step 3. Construct a weighted graph representing all possible travel path combinations of each elevator with either elevator *A* or *B* answering one, or both, or none of the hall calls as shown in Fig. 2. 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.

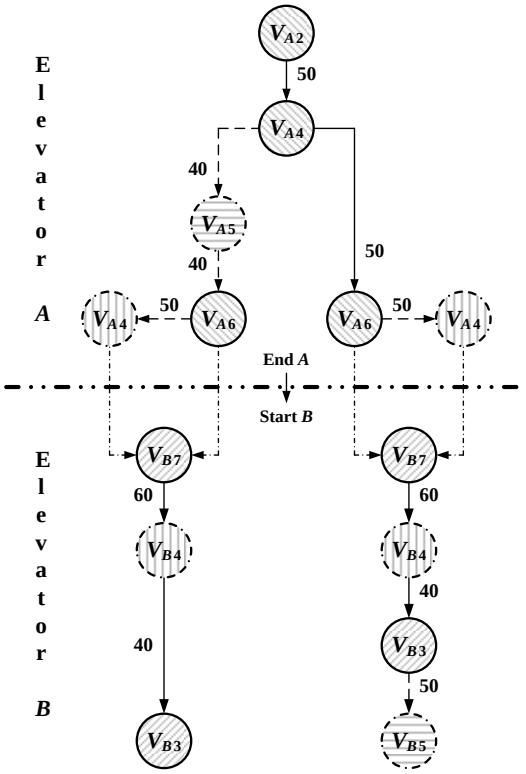

Fig. 2. Weighted graph representing all possible travel path combinations of elevators *A* and *B*

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 of the elevators as tabulated in Table 3. The required solution for the elevator scheduling problem is thus the optimal path weight with the total graph output $G(A, B)_3 = 230 = 115$ sec.

Table 3. Total graph output for all possible travel path combinations of elevators *A* and *B*

Elevator	Hall calls	Elevator movements	Total graph output
A	–	$V_{A2} \to V_{A4} \to V_{A6} \to$	$G(A, B)_1$
B	4′, 5	$V_{B7} \to V_{B4} \to V_{B3} \to V_{B5}$	$= (100) + (150)$ $= 250$
A	4′	$V_{A2} \to V_{A4} \to V_{A6} \to V_{A4}$	$G(A, B)_2$
B	5	$\to V_{B7} \to V_{B4} \to V_{B3} \to V_{B5}$	$= (150) + (150)$ $= 300$
A	5	$V_{A2} \to V_{A4} \to V_{A5} \to V_{A6}$	$G(A, B)_3$
B	4′	$\to V_{B7} \to V_{B4} \to V_{B3}$	$= (130) + (100)$ $= 230$
A	4′, 5	$V_{A2} \to V_{A4} \to V_{A5} \to V_{A6}$	$G(A, B)_4$
B	–	$\to V_{A4} \to V_{B7} \to V_{B4} \to V_{B3}$	$= (180) + (100)$ $= 280$

Step 4. Redraw the graph of Fig. 2 in order to distinguish between start, immediate and end nodes and also to differentiate between the nodes of different travel path combinations as shown in Fig. 3.

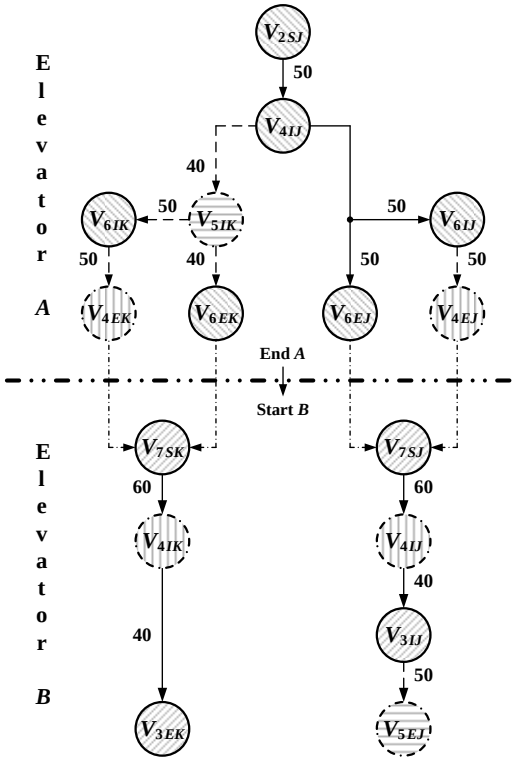

Fig. 3. Modified weighted graph showing node locations and paths of elevators *A* and *B*

Note that *S*, *I* and *E* denote start, intermediate and end nodes respectively, while *J* and *K* denotes the different travel paths of the elevators. This is important since every different node location and path in the graph will

be represented with different oligos in order to obtain all the possible travel path combinations that fulfill all the initial conditions and requirements stated.

Step 5. Generate a unique DNA sequence for each of the nodes where each intermediate node of different travel paths is assigned with a specific DNA sequence and each start or end node of different travel paths is assigned with another specific DNA sequence. Hence, every DNA sequence assigned to each node will identify its location and travel path. Using available software for DNA sequence design named DNASequenceGenerator [9], the sequence is generated as shown in Table 4. The GC contents (GC%), melting temperature (T_m) are also shown in the table, and the sequence complements are shown in Table 5.

Table 4. Generated DNA sequences for nodes

Node *Vi*	20-mer Sequence (5'–3')	GC%	T_m (°C)
V_{2SJ}	cggcggtccactaaatacta	50	60.0
V_{4IJ}	cactctttgtgaacgccttc	50	60.8
V_{5IK}	gtgggttagaggtagtccgg	60	60.8
V_{6IJ}	tgaaccggccctttatatct	45	60.7
V_{6IK}	ccgctgatccttgctaagta	50	60.4
V_{4EJ}	tcattcgagttattcctggg	45	59.9
V_{4EK}	aaatgacctttttaacggca	35	59.4
V_{6EJ}	ctataaggccaaagcagtcg	50	59.9
V_{6EK}	atgcctggctaaagtgagac	50	59.3
V_{7SJ}	ggacctgcatcataccagtt	50	59.8
V_{7SK}	tgcacgcaaaactatttcat	35	59.2
V_{3IJ}	aaagcccgtcggttaagtta	45	60.8
V_{4IK}	tctgcactgttaatgagcca	45	60.4
V_{3EK}	ctacggataggtgtctggga	55	59.9
V_{5EJ}	ggaatccattgatcgcttta	40	59.9

Table 5. Complement of generated DNA sequences

Node *Vi*	20-mer Sequence (5'–3')
$\overline{V_{2SJ}}$	tagtatttagtggaccgccg
$\overline{V_{4IJ}}$	gaaggcgttcacaaagagtg
$\overline{V_{5IK}}$	ccggactacctctaacccac
$\overline{V_{6IJ}}$	agatataaagggccggttca
$\overline{V_{6IK}}$	tacttagcaaggatcagcgg
$\overline{V_{4EJ}}$	cccaggaataactcgaatga
$\overline{V_{4EK}}$	tgccgttaaaaaggtcattt
$\overline{V_{6EJ}}$	cgactgctttggccttatag
$\overline{V_{6EK}}$	gtctcactttagccaggcat
$\overline{V_{7SJ}}$	aactggtatgatgcaggtcc
$\overline{V_{7SK}}$	atgaaatagttttgcgtgca
$\overline{V_{3IJ}}$	taacttaaccgacgggcttt
$\overline{V_{4IK}}$	tggctcattaacagtgcaga
$\overline{V_{3EK}}$	tcccagacacctatccgtag
$\overline{V_{5EJ}}$	taaagcgatcaatggattcc

Step 6. Synthesize the oligos for every node path in the graph according to the following rules [7] so that the oligos length will directly represent the weight between the nodes:

(i) If *i* is a start node and *j* is an intermediate node, synthesize the oligo as

$$V_i(20) + W_{ij}(\omega_{ij} - 30) + V_j(20)$$

(ii) If *i* is an intermediate node and *j* is an end node, synthesize the oligo as

$$V_i(20) + W_{ij}(\omega_{ij} - 30) + V_j(20)$$

(iii) If *i* and *j* are both intermediate nodes, synthesize the oligo as

$$V_i(20) + W_{ij}(\omega_{ij} - 20) + V_j(20)$$

where *V* denotes the DNA sequence for node, *W* denotes the DNA sequence for weight, ω denotes the weight value, and '+' denotes a 'join' between the DNA sequence. All the synthesized oligos based on the stated rules are shown in Table 6 where capital letters denote the nodes and small letters denote the weight between the nodes.

Table 6. Synthesized DNA sequences for node paths

Node Path	DNA Sequence (5' – 3')
$V_{2SJ} \rightarrow V_{4IJ}$	CGGCGGTCCACTAAATACTAaggtcgttta aggaagtacgCACTCTTTGTGAACGCCTTC
$V_{4IJ} \rightarrow V_{5IK}$	CACTCTTTGTGAACGCCTTCacgtcgtgta acgaagtcctGTGGGTTAGAGGTAGTCCGG
$V_{4IJ} \rightarrow V_{6IJ}$	CACTCTTTGTGAACGCCTTCccgtcggttaagcaa gtaatgtactatgctTGAACCGGCCCTTTATATCT
$V_{4IJ} \rightarrow V_{6EJ}$	CACTCTTTGTGAACGCCTTCgcgtcgcttaccgaa gcacgCTATAAGGCCAAAGCAGTCG
$V_{5IK} \rightarrow V_{6IK}$	GTGGGTTAGAGGTAGTCCGGcgctcgttga agccagtaccCCGCTGATCCTTGCTAAGTA
$V_{5IK} \rightarrow V_{6EK}$	GTGGGTTAGAGGTAGTCCGGGcgtcttttaATG CCTGGCTAAAGTGAGAC
$V_{6IJ} \rightarrow V_{4EJ}$	TGAACCGGCCCTTTATATCTacgtgtttta cccaagtcagTCATTCGAGTTATTCCTGGG
$V_{6IK} \rightarrow V_{4EK}$	CCGCTGATCCTTGCTAAGTAgcggcgtgtc acgaactacGAAATGACCTTTTTAACGGCA
$V_{4EJ} \rightarrow V_{7SJ}$	TCATTCGAGTTATTCCTGGGGGACCTGCAT CATACCAGTT
$V_{6EJ} \rightarrow V_{7SJ}$	CTATAAGGCCAAAGCAGTCGGGACCTGCA TCATACCAGTT
$V_{4EK} \rightarrow V_{7SK}$	AAATGACCTTTTTAACGGCATGCACGCAA AACTATTTCAT
$V_{6EK} \rightarrow V_{7SK}$	ATGCCTGGCTAAAGTGAGACTGCACGCAA AACTATTTCAT
$V_{7SJ} \rightarrow V_{4IJ}$	GGACCTGCATCATACCAGTTacgtggtttaaggaag tacggtactatgctCACTCTTTGTGAACGCCTTC
$V_{7SK} \rightarrow V_{4IK}$	TGCACGCAAAACTATTTCATccgtgggttaaagaa gtcctgtactctcctTCTGCACTGTTAATGAGCCA
$V_{4IJ} \rightarrow V_{3IJ}$	CACTCTTTGTGAACGCCTTCacgtcgctgc aagaactacGAAAGCCCGTCGGTTAAGTTA
$V_{4IK} \rightarrow V_{3EK}$	TCTGCACTGTTAATGAGCCAacgtcttgtcCTAC GGATAGGTGTCTGGGA
$V_{3IJ} \rightarrow V_{5EJ}$	AAAGCCCGTCGGTTAAGTTAggtcttttaa tcaactaatgGGAATCCATTGATCGCTTTA

Step 7. All the synthesized oligos are then poured into a test tube for initial pool generation using parallel overlap assembly (POA) method [10]. POA is used as suggested by Lee *et al.* [11] who demonstrated that POA is a more efficient and economical initial pool

generation method for weighted graph problems. POA operation consists of three steps: hybridization, extension, and denaturation. During the annealing step, the temperature is decreased slowly so that partial hybridization is allowed to occur at respective locations. The extension on the other hand is applied with the presence of polymerase enzyme and the polymerization can be done from 5' to 3' direction. The generated double stranded DNA molecules are then separated during denaturation step in which the temperature is increased until the double stranded DNA molecules are separated to become single stranded DNA molecules. An example of the POA showing the optimal path for this elevator scheduling problem is depicted in Fig. 4.

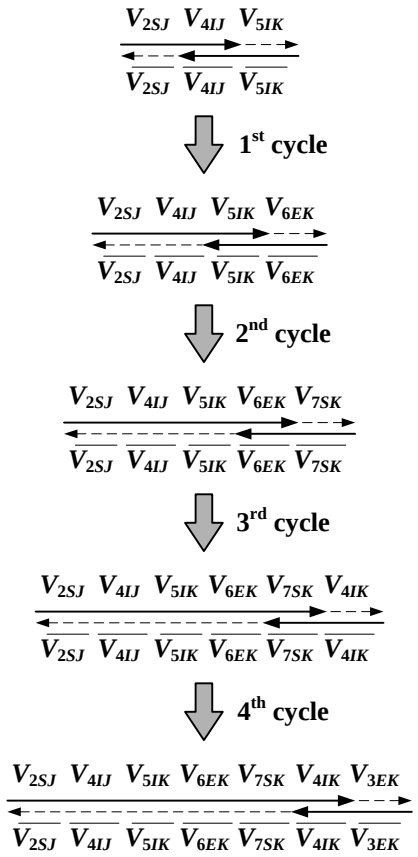

Fig. 4. POA for elevator optimal path $V_{A2} \rightarrow V_{A4} \rightarrow V_{A5} \rightarrow V_{A6} \rightarrow V_{B7} \rightarrow V_{B4} \rightarrow V_{B3}$. The continuous arrows represent the synthesized oligos and dotted arrows represent the elongated part during polymerization. The arrowhead indicates the 3' end

Step 8. An initial pool of solution is produced at this stage. The optimal path combinations among many other alternative path combinations of the problem have to be filtered. This filtering process copies the target DNA duplex exponentially using polymerase chain reaction (PCR) process [12]. PCR proceeds in cycles of 3 steps at different temperatures: denaturation (95°C), involves separation of the double strand DNA molecules, annealing (55°C) where primers are 'annealed' to both the single strands ends and extension (75°C) process where polymerase enzymes are used to extend the primers into replicas of the DNA molecules. This sequence is repeated causing an exponential growth in the number of target DNA molecules. For this problem, all the DNA molecules containing start node V_{2SJ} and end node V_{3EK} are amplified exponentially. Numerous amounts of DNA strands representing the start node V_{2SJ} and end node V_{3EK} passing through all possible travel path combinations will be presented once the PCR operation is accomplished.

Step 9. Finally, in order to separate all the possible travel path combinations according to its length, gel electrophoresis [13, 14] is performed onto the output solution of the PCR. The gel electrophoresis image are then captured, where the DNA duplex representing the shortest path starting from V_{2SJ} and end node V_{3EK} could be visualized representing the required optimal path solution of the problem.

4. Experiment Setup and Results

The POA method for initial pool generation is performed in a 100 μl solution consisting of 64.0 μl distilled water (Maxim Biotech), 15.5μl oligos (Proligo Primers & Probes, USA), 10 μl dNTP (TOYOBO, Japan), 10 μl 10× KOD dash buffer (TOYOBO, Japan), and 0.5 μl KOD dash polymerase (TOYOBO, Japan). The solution is then subjected to POA reaction of 25 cycles where the different temperatures for each cycle are 94ºC for 30sec, 55ºC for 30sec and 74ºC for 10sec respectively.

PCR is then performed for DNA amplification in order to select the paths that begin with node V_{2SJ} and ending at node V_{3EK} and V_{5EJ}. PCR is performed in a 25 μl solution consisting of 17.375 μl distilled water (Maxim Biotech), primers V_{2SJ} , $\overline{V_{3EK}}$, and $\overline{V_{5EJ}}$ of 0.5 μl each, 1 μl POA template, 2.5 μl dNTP (TOYOBO, Japan), 2.5 μl 10× KOD dash buffer (TOYOBO, Japan), and 0.125 μl KOD dash polymerase (TOYOBO, Japan). The solution is then subjected to PCR reaction of 25 cycles where the different temperatures for each cycle are 94ºC for 30sec, 55ºC for 30sec and 74ºC for 10sec respectively, i.e. the same as POA process.

Finally, the PCR solution is subjected to gel electrophoresis for 30 minutes in order to visualize the computation result. SYBR Gold (Molecular Probes) is used to stain the gel after gel electrophoresis process before the gel image is captured.

The captured image for the POA and PCR process is shown in Fig. 5. Lane *M* denotes 20bp ladder while lanes 1 and 2 denote POA and PCR products respectively. It is clearly seen from the POA gel image that the band is blurs denoting that all possible travel paths are successfully generated. The PCR gel image shows 4 bands indicating all the four possible travel paths, i.e. $G(A, B)_3$ = 230bp, $G(A, B)_1$ = 250bp, $G(A, B)_4$ = 280bp and $G(A, B)_2$ = 300bp. This confirms the expected result that the optimal elevator's travel path is given by $G(A, B)_3$ = 230bp = 115 sec.

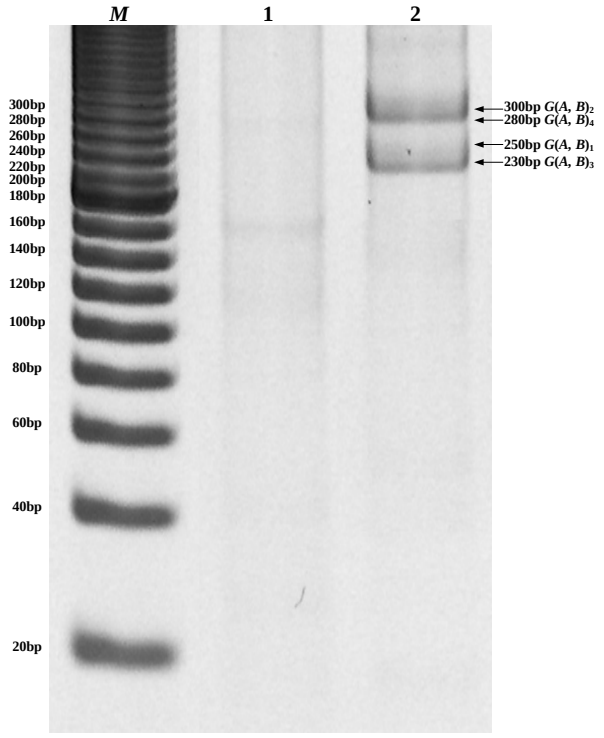

Fig. 5: Gel electrophoresis image result. Lane *M* denotes 20bp ladder and lanes 1 and 2 is the POA and PCR products respectively

5. Conclusions

Implementation ideas and experimental procedures for an application of direct-proportional length-based DNA computing method to solve an elevator scheduling problem have been presented and discussed in details in this paper. Experimental result that has been carried out verifies that the shortest DNA sequence length represents the required optimal path for the elevator scheduling problem. This can be visualized from the gel electrophoresis image of Fig. 5 where all the four possible travel paths combination of the elevator movement are represented by the four bands of PCR product image sorted in sequence of DNA lengths representing the travel time of each travel path combination. For a larger problem with *M* elevators, *N* floors and *Y* hall calls, all the M^Y travel path combinations can be represented by specific DNA sequences synthesized using the rule as in [7]. POA and PCR can thus be performed to extract the required computing solution from the gel electrophoresis image. This research shows that this type of engineering problem is applicable and achievable to be solved using DNA computing approach. Hence, the applicability and feasibility of DNA computing could therefore be extended into many more complex problems of this type of nature.

References

[1] L.M. Adleman, "Molecular Computation of Solutions to Combinatorial Problems," Science, Vol. 266, 1994, pp. 2021-1024.

[2] L.M. Adleman, "Computing with DNA," Scientific American, 1998, pp. 34-41.

[3] J.D. Watson, and F.H.C. Crick, "A Structure for Deoxyribose Nucleic Acid", Nature, Vol. 171, 1953, pp. 737-738.

[4] Narayanan, and S. Zorbalas, "DNA Algorithms for Computing Shortest Paths," Proceedings of Genetic Programming, 1998, pp. 718-723.

[5] 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, Vol. 2, 2002, pp. 447-455.

[6] 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, Springer-Verlag, Vol. 2568, 2003, pp. 73-84.

[7] 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, Vol. 1, Issue 1, 2004, pp. 46-60.

[8] M.S. Muhammad, S. Ueda, O. Ono, and M. Khalid, "DNA-based Computing for Solving Elevator Scheduling Problem," 3rd International Conference on Computer Applications (ICCA2005), 2005, pp. 507-514.

[9] 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, 2001, pp. 23-32.

[10]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, Vol. 188, Issue 3, 1997, pp. 333-341.

[11]J.Y. Lee, H.W. Lim, S.I. Yoo, B.T. Zhang, and T.H. Park, "Efficient Initial Poo Generation for Weighted Graph Problems using Parallel Overlap Assembly," Preliminary Proceeding of the 10th International Meeting on DNA Computing, 2004, pp. 357-364.

[12]J. P. Fitch, *Engineering Introduction to Biotechnology*, SPIE Press, 2001.

[13]G. Paun, G. Rozenberg, and A. Salomaa, "DNA computing: New Computing Paradigms," Lecture Notes in Computer Science, Springer-Verlag, Vol. 1644, 1998, pp. 106-118.

[14]Y. Yamamoto, A. Kameda, N. Matsuura, T. Shiba, Y. Kawazoe, and A. Ahochi, "A Separation Method for DNA Computing based on Concentration Control," New Generation Computing, Vol. 20, No. 3, 2002, pp. 251-262.