DNA Computing Algorithm for a school Timetable Problem

Kuntala Boruah[1], Manash Kapil Pathak[2], Ranjan Sarmah[1]
[1]Assistant Professor, School of Innovation and Technology, Assam Rajiv Gandhi University of Cooperative Management (ARGUCOM), Sivasagar, Assam, India
[2]Assistant Controller of Examination, Mahapurusha Srimanta Sankaradeva Viswavidyalaya, Nagaon, Assam, India

Deoxyribonucleic acid (DNA) computing is believed to have the potential to offer an effective approach to reduce any NP problem from exponential to polynomial time. Recently, use of biomolecules for solving scheduling problems has gained tremendous attention. In this paper a theoretical proof-of concept algorithm is proposed to address timetable scheduling problem which is a classical NP complete problem. The efficiency of this algorithm owes to the parallel processing property of DNA. Information relating to resources like the set of classes, teachers, time slots and subjects are encoded in the form of unique DNA sequences. Initially, all the possible (valid as well as invalid) allocations are generated and, in each step, the illegal sequences are discarded until finally left out with one or more potential solutions that satisfy the given set of constraints. The time complexity of the proposed algorithm is independent of the size of the problem. Moreover, the proposed algorithm can be applied to solve several other scheduling problems with necessary modifications.

Keywords: DNA; DNA Computing; NP complete problem; Parallelism; Timetable problem.

1. INTRODUCTION

DNA computing is a rapidly evolving area that employs the strands of DNA along with wet lab biochemical reactions for computational purpose. Adleman [1994] recognized the computational power of DNA and demonstrated the first ever experimental solution to an instance of Hamiltonian path problem. After his innovative attempt, DNA computing is witnessing tremendous growth. Lipton [1995] proposed a model to solve the satisfiability problem (SAT). Since then, DNA computing has been receiving enormous attention from researchers working in a wide range of areas. Tides of new models have been proposed since 1994 which are categorized into Adleman-Lipton model, sticker model, restriction enzyme model, self-assembly model and surface-based model, logic gate and Boolean circuit simulation model etc. (Adleman [1998], Lipton [1995], Rowies et al. [1998], Quyang et al. [1997], Winfree et al. [1998], Sakamoto et al. [2000], Smith et al. [1998], Boruah and Dutta [2018]). The parallelism property of DNA has the potential to reduce NP problems from exponential time to polynomial time. Based on this property, several papers have been published suggesting algorithms to solve various NP-complete problems (Li et al. [2006], Xiao et al. [2006], Chang et al. [2012], Wang et al. [2012], Wang et al. [2013], Wang et al. [2014], Chang et al. [2014]).

In this work a theoretical model is proposed to solve an instance of school time table problem using specially encoded strands of DNA. Biochemical operations such as affinity purification, PCR, gel electrophoresis are used as computational tools for the model.

1.1 Related Work in The Field of Timetable Problem

Almost three decades ago Gotlieb [1963] and Even et al. [2002] showed that all timetable problems are NP-Complete. It deals with the task of scheduling faculty (resource person) to deliver lectures to the allotted classes at suitable time slots in accordance with the predefined set of constraints. Over the time, several approaches such as genetic algorithm (Chu and Fang [1999], Cowling et al.

[2002], Sigl et al. [2003], Pezzella et al. [2008], Bhaduri [2009], Ghaemi et al. [2007]), Swarm Optimization(Chu et al. [2006]), agent-based approach (Opera [2007]), combination of simple search and swapping i.e. simple search swapping (Aycan and Ayav [2009]), column generation approach (Huisman [2006]), combination of swarm optimization and local search (Chen and Shih [2013], Zhang et al. [2019]), differential evolution (Zhong et al. [2012]), graph coloring approach (Ganguli and Roy [2017], Redl [2007], Burke et al. [1994], Wood [1968]) etc. have been employed to address such problems.

Beside the above strategies, non-convention techniques such as DNA computing emerged as one of the most interesting candidates due to its inherent parallelism property. Cheng et al. [2010] utilized the DNA tile self-assembly property to solve timetable problems with complexity O(mn). Zhixiang Yin and his co-researchers (Yin and Chen [2010]) demonstrated an easy and feasible model to address timetable problems by employing an innovative technology of Acrydite^{TM} gel separation. Wang et al. [2015] came up with a model to solve an unbalanced assignment problem which ensures applicability to real life situations. Popov et al. [2014] suggested another algorithm to solve timetable problems.

Brute force strategy ensures solution to NP class problems by generating all possible candidate solutions and then each of the candidate is examined for best optimal solution but this strategy suffers from major drawback leading to vast solution space which eventually results in polynomial time (2^n, n! or n^n) search space. DNA computing offers a most convincing solution to overcome these disadvantages owing to its inherent parallelism property.

The organization of this paper is as follows. Section 2 formally describes the proposed algorithm to solve an instance of school timetable by implementing brute force approach in parallel mode. Sub-section 2.1 discusses the theoretical implementation of the proposed algorithm at biochemical level. In sub-section 2.2 the efficiency of the algorithm is investigated in terms of time complexity.

2. DNA ALGORITHM FOR TIMETABLE PROBLEM

A timetable problem is a classical NP-complete problem in which the task is to schedule or allocate resources without any conflict to the given constraints. Every timetable problem has a different set of constraints depending on the requirement, for e.g., an exam scheduling timetable problem has different set of constraints as compared to a class scheduling timetable problem.

In this paper, a new non-deterministic algorithm has been proposed to solve the problem of school timetable scheduling using the parallel processing capacity of DNA. Initially, a set of information needed to be gathered relating to each teachers preference so that a known set is formed. Later on, a set of constraints are formulated according to the requirement. For simplicity of explaining, the authors have considered a special case of a school timetable where:

Information relating to each teachers preference is gathered initially:

(1) Subject interests of the teachers along with the classes or semesters they are interested in teaching those subjects, i.e., if a teacher has a subject interest as science then he has to mention the class he is interested in teaching (for e.g., Science of class XII)
(2) Preferred time slots of each teacher must be acquired in such a way that the total number of preferred time slots is always greater or equal to the number of subjects needed to be taught by each teacher (preferred time slots of each teacher $\geqslant$ number of subjects needed to be taught by each teacher).

Set of constraints for the proposed algorithm:

(1) Not more than one teacher should be allocated to the same class at the same time slot.
(2) All the subjects should be taught to each class exactly once every day.
(3) No class should remain unallocated at any time slot.

The problem is to allot time slots to each teacher according to their preferences without conflicting with the time slots of other teacher assigned to the same class. For e.g., if a teacher T_1

has been assigned subjects S_1 and S_2 to be taught to class C_1 and C_3 respectively then, while formulating the timetable care must be taken to ensure allotments in his preferred time slots and to avoid any allotment in his unfavourable subjects or time slots.

The finite set of classes, teachers, subjects and time slots are represented as:

(1) 'n' numbers of classes: C = {C_1, C_2, C_3, ..., C_n}
(2) 'm' number of teachers: T = {T_1, T_2, T_3, ..., T_m}
(3) 'x' number of subjects: S = {S_1, S_2, S_3, ..., S_x}
(4) 'y' numbers of time slots: t = {t_1, t_2, t_3, ..., t_y}

The proposed algorithm:
Step 1: Generate all possible random allotments for each class separately.
Step 2: Select those allotments that include 1st and last time slots.
Step 3: Extract allotments that have correct length.
Step 4: Keep all allocations that have all subjects and time slots at least once.
Step 5: Generate all possible random allotments for the entire time table.
Step 6: Select those allotments that start with 1st class and end with last class.
Step 7: Extract allotments that have correct length.
Step 8: Keep all allotments that have all subjects, time slots and class at least once.

The above-mentioned algorithm is represented in the form of a pseudocode with four procedures as shown below. DNATimeTable_Main() (Algorithm 1) is the main procedure from where EncodeInformation_Strand() (Algorithm 2), EncodeSplint_Strand1() (Algorithm 3), EncodeSplint_Strand2() (Algorithm 4) are called.

Biological operations: Merge(), Append(), PCR_amplification(), Affinity_purification() and Extract() are used to construct the timetable solving problem which are derived from Adleman - Lipton model (Adleman [1994], Lipton [1995]).

The Adleman-Lipton model is constructed upon the following biological operations:
Append (T, S): String S is ligated to the end of all DNA strands in test-tube T.
Copy (T, tt_1, tt_2, , tt_n): Several replicas of DNA strands of test-tube T is created and placed in test-tubes tt_1 to tt_n .
Merge (T, tt_1, tt_2, , tt_n): The contents of test tubes tt_1 to tt_n are poured into tube T and allow them to undergo biochemical reaction.
Extract (T, S, T+, T-): The operation produces two different test-tube T+ and T- depending on the presence or absence of string S respectively.
Detect (T): Given a tube T, this operation returns true if there is at least one DNA strand in T, otherwise it returns false.
Discard (T): Given a tube T, this operation ignores T.
Affinity_purification(T, S, tt): In this operation magnetic beads attached with covalently bonded DNA sequence are used to filter out all those DNAs in tube T that contain S sequence at least once. The extracted sequences are stored in test-tube tt.

The algorithm returns output as either a non-empty set which signifies one or more feasible scheduling schemes or as an empty set which means the corresponding timetable problem doesn't have any solution or valid schedule for the given constraints.

2.1 Biochemical implementation of the proposed algorithm

For simplicity in demonstrating the biochemical implementation of the proposed algorithm, an instance of school timetable with three classes, four teachers, three time slots, and three subjects are considered, which are represented by the notation as (C_1, C_2 and C_3), (T_1, T_2, T_3, T_4), (t_1, t_2, t_3) and (S_1, S_2, S_3) respectively. Table I show the outline of the desired timetable.

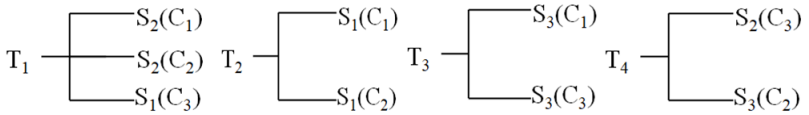
Information relating to each teachers preference:

The following dummy information relating to the subjects and class preferences (Figure 1) is assumed to be collected from each teacher before the execution of the algorithm that is represented

Algorithm 1: Pseudocode for solving the time table problem

DNATimeTable_Main()
Begin
increment_value = 1
Empty (tt_{info}, $tt_{splint1}$, $tt_{splint2}$, tt_{info}, tt_{1k}, tt_{2k}, tt_{3k}, tt_{4k}, tt_{result}, tt_{final_result})
EncodeInformation_Strand1()
EncodeInformation_Strand2()
{
for (k ← 1 to n, increment_value) **do**
{
Merge(tt_{1k}, $tt_{splint1}$, tt_{info})
PCR_amplification(tt_{1k}, $\bar{C}_n$, $\bar{C}_n$)
Extract(tt_{1k}, 170bp, tt_{2k})
for (i ← 1 to x, increment_value) **do**
{
Affinity_purification(tt_{2k}, $\bar{S}_x$, tt_3)
}
end for
for (j ← 1 to t, increment_value) **do**
{
Affinity_purification(tt_3, $\bar{t}_j$, tt_4)
}
end for
Merge(tt_{all}, $tt_{splint2}$, tt_4)
PCR_amplification(tt_{all}, $\bar{C}_1$, $\bar{C}_n$)
Extract(tt_{all}, 510bp, tt_{result})
for (k ← 1 to m, increment_value) **do**
{
Affinity_purification(tt_{result}, $\bar{T}_m$-$\bar{t}_y$, tt_{final_result})
}
end for
}
end for
}
End

as $T_m \rightarrow S_x(C_n)$ i.e. teacher T_m interested in teaching subject S_x to class C_n. Similarly, preference of time slots is shown in Figure 2 which is represented as $T_m \rightarrow t_y$, i.e. Teacher T_m prefers to teach in time slot t_y.



. Figure 1: Subject preferences of each teacher ($T_m \rightarrow S_x(C_n)$).

As the entire algorithm is conceptualized to realize in the wet lab, all inputs (collected information and constraints) are translated in the form of strands of DNA. To attain this, each of the notations such as C_1, C_2, C_3, T_1, T_2, T_3, T_4, S_1, S_2, S_3, t_1, t_2, and t_3 are pre-assigned with

Algorithm 2: Procedure for encoding teacher information strand

EncodeInformation_Strand()
Begin
for (j ← 1 to m, increment_value) **do**
{
ligate(time_slot)
Begin
if (time_slot == first) **then**
tt_{info}← 3'-C_n-t_y-C_n-S_x-T_j-t_y-5'
end if
if (time_slot == last) **then**
tt_{info}← 3'-t_y-C_n-S_x-T_j-t_y-C_n-5'
else
tt_{info}← 3'-t_y-C_n-S_x-T_j-t_y-5'
end if
End
}
end for
End

Algorithm 3: Procedure for encoding splint strand for every class separately

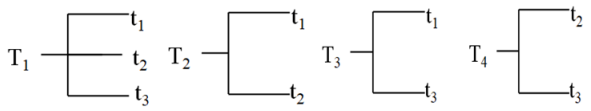
EncodeSplint_Strand1()
Begin
for (i ← 1 to y, increment_value) **do**
{
ligate(tt_splint1, $\bar{t}_1$, $\bar{t}_{i+1}$)
}
end for
End

Algorithm 4: Procedure for encoding splint strand for all the classes together (entire time-table)

EncodeSplint_Strand2()
Begin
for (i ← 1 to n, increment_value) **do**
{
ligate($tt_{splint2}$, $\bar{C}_1$, $\bar{C}_{i+1}$)
}
end for
End

Table I. Instance of a School Time Table

Class	Time slot		
	(9-10 AM) t_1	**(10-11 AM) t_2**	**01-02 PM) t_3**
C_1			
C_2			
C_3			



. Figure 2: Time slots preferences of each teacher ($T_m \rightarrow t_y$).

Table II. Encoding of information strand for each teacher

T_m	Information Strand	
T_1	t_1(where y = 1)	3'-C_1-t_1-C_1-S_2-T_1-t_1-5' 3'-C_2-t_1-C_2-S_2-T_1-t_1-5' 3'-C_3-t_1-C_3-S_2-T_1-t_1-5'
	t_2(where y = 2)	3'-t_2-C_1-S_2-T_1-t_2-5' 3'-t_2-C_2-S_2-T_1-t_2-5' 3'-t_2-C_3-S_1-T_1-t_2-5'
	t_3(where y = 3)	3'-t_3-C_1-S_2-T_1-t_3-C_1-5' 3'-t_3-C_2-S_2-T_1-t_3-C_1-5' 3'-t_3-C_3-S_1-T_1-t_3-C_1-5'
T_2	t_1(where y = 1)	3'-C_1-t_1-C_1-S_1-T_2-t_1-5' 3'-C_2-t_1-C_2-S_1-T_2-t_1-5'
	t_2(where y = 2)	3'-C_2-C_1-S_1-T_2-t_2-5' 3'-t_2-C_1-S_1-T_2-t_2-5'
T_3	t_1(where y = 1)	3'-C_1-t_1-C_1-S_3-T_3-t_1-5' 3'-C_3-t_1-C_3-S_3-T_3-t_1-5'
	t_3(where y = 3)	3'-t_3-C_1-S_3-T_3-t_3-C_1-5' 3'-t_3-C_3-S_3-T_3-t_3-C_3-5'
T_4	t_2(where y = 2)	3'-t_2-C_2-S_3-T_4-t_2-5' 3'-t_2-C_3-S_2-T_4-t_2-5'
	t_3(where y = 3)	3'-t_3-C_2-S_3-T_4-t_3-C_2-5' 3'-t_3-C_3-S_2-T_4-t_3-C_3-5'

fixed length 10-mer random but unique sequence of DNA. On execution of procedure Encode-Information_Strand( ), the derived information of each teacher are translated to single strands of DNA. Table II shows the encoded information of each teacher depending on the allocation of time slots. The encoding at the 1st time slot has C_n at 3'end whereas the encoding at the last time slot has C_n at 5'end. All the intermediate time slots do not have C_n at any of its extreme ends.

The biochemical reaction corresponding to each class is executed in a separate test tube.

To generate all possible combinations, two sets of DNA sequences are used as splints, i.e., Splint1 and splint2; Splint1 is used to generate combinations of allotments corresponding to each class (see Table III) whereas splint2 is used to generate all possible combinations of allotments for the entire timetable (see Table IV).

Splint1 is obtained by ligating complements of t_y and complement of t_{y+1} sequence i.e., 5'-$\bar{t}_y$-$\bar{t}_{y+1}$-3'(see Table III). This 20-mer oligo sequence works as a splint to bring together the encoded information strands based on simple hybridization reactions (shown in Figure 3).

Step 1 to step 4 dedicated to generate scheduling sequence for individual class (each test-tube

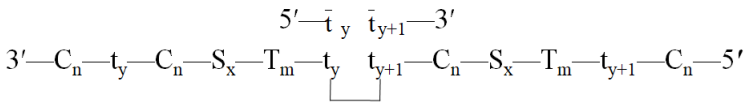
Table III. Encoding of splint$_1$ strands

	5'-$\bar{t}_y$–$\bar{t}_{y+1}$-3'
y = 1	5'-$\bar{t}_1$–$\bar{t}_2$-3'
y = 2	5'-$\bar{t}_2$–$\bar{t}_3$-3'

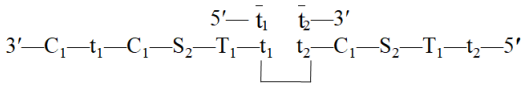
corresponds to single row of the timetable) where as step 5 to step 8 dedicated to generate allocation for entire timetable (all the rows together).

Step 1: *Generate all possible random allotments for each class separately:*

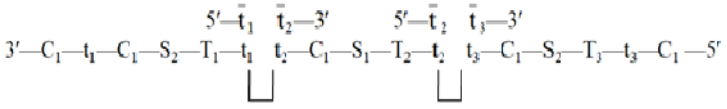
Once the splint1 (shown in Table III) and the information strands (shown in Table II) are encoded, the class-wise random combinations of allocations are generated when 50 pmol of information strand related to a particular class is mixed with 50 pmol of splint strands 5'-$\bar{t}_y$–$\bar{t}_{y+1}$-3'in a test-tube and let them hybridize and ligate to form a longer strand involving all or few allocations (shown in Figure 3). For example to generate probable combinations for class C_1, all encoded information relating to class C1 along with splint sequences 5'-$\bar{t}_1$–$\bar{t}_2$-3'and 5'-$\bar{t}_2$–$\bar{t}_3$-3'are allowed to hybridize. Figure 4 and Figure 5 depicts two of the several probable combinations consisting of both valid and invalid combinations. A sequence is valid if it consists of all the subjects along with all the time slots without repetition.



. Figure 3: Generation of possible combinations for class C_n with the help of splint_1.



. Figure 4: An instance of an invalid combination for class C_1.



. Figure 5: An instance of a valid combination for class C_1.

Similarly, there are three other test-tubes where trillions of random combinations related to their corresponding classes such as for class2 and class3 are generated. The product of step 1 consists of both valid as well as invalid sequences.

Step 2: *Select those allotments that include 1st and last time slots:*

During step 2, only those sequences which includes allotment at time slot t_1 at the beginning and t_3 at the ending are extracted and separated from the remaining sequences. To realize this in a wet lab, polymer Chain Reaction (PCR) is carried out in each of the test tubes with $\bar{C}_n$ as primers. PCR is used to amplify a specific section of DNA with the help of primers that mark the starting and the ending of the desired section. The reason behind choosing $\bar{C}_n$ as primer is because during information encoding (shown in Table I) the first time slot has C_n at 3'end

whereas the last time slot has C_n at 5'end therefore any sequence with C_n at the beginning as well as at the ending signifies that it is associated with allotments to time slots t_1 and t_3 respectively. Thus, in test tube 1, only those combinations which have C_1 at the beginning as well as at the ending will only be amplified. Similarly, in test tube 2 and test tube 3, PCR is carried out with $\bar{C}_2$ and $\bar{C}_3$ as primers respectively.

Step 3: *Extract allotments that have correct length:*

The DNA strands obtained after Step 2 in each of the test tubes are run through gel which results in different bands corresponding to different lengths (gel electrophoresis). The estimated length of sequences which have the allocation of all the time slots exactly once is 170 base pair (bp) so after running through gel only the band associated to the desired length is separated for further use.

Valid allotment sequence = $|3'\text{-}C_n\text{-}t_y\text{-}C_n\text{-}S_x\text{-}T_m\text{-}t_yt_y\text{-}C_n\text{-}S_x\text{-}T_m\text{-}t_yt_y\text{-}C_n\text{-}S_x\text{-}T_m\text{-}t_y\text{-}C_n\text{-}5'|$

Total length = 170 bp

The strand obtained during this step ensures the allocation to all the three time slots but the possibility of any repetition is not checked during this step.

Step 4: *Keep all allocations that have all subjects and time slots at least once:*

Affinity purification is carried out to check the presence of all of the three subjects. During this step the $\bar{S}_x$ strand incubated magnetic bead is used to fish out the desired strand. Only those sequences which contain S_x will anneal to the $\bar{S}_x$ attached to magnetic bead and hence could be easily extracted. To check for the presence of three subjects this process has to be repeated for three time with $\bar{S}_1$, $\bar{S}_2$ and $\bar{S}_3$. Again the same process has to be executed with $\bar{t}_y$ ($y = 1 \ldots 3$) incubated beads to check for all time slots in the sequences. The final fished out strands from each test tubes are the class wise valid allocation schedule.

Step 5 to step 8 dedicated to generate scheduling sequence for all the classes together (entire timetable).

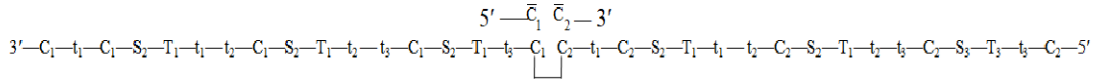
Step 5: *Generate all possible random allotments for the entire time table:*

In step 5 the valid strands from each of the test tubes are mixed in a single test tube so that random DNA sequences are generate for all the classes together (entire timetable). To implement this, second set of splint sequences (shown in Table IV) i.e., splint2 is used to bring the sequences of each class closer so that they can ligate to produce a longer strand. Splint2 strand are designed with complements of C_n strand and C_{n+1} strand i.e., Splint2 $\leftarrow$ append ($\bar{C}_n$, $\bar{C}_{n+1}$).

Table IV. Encoding of splint$_2$ strands

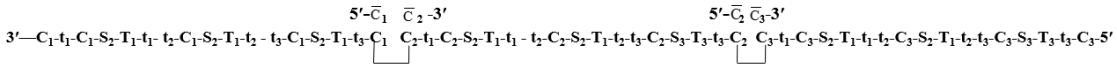
	$5'\text{-}C_n\text{–}\bar{C}_{n+1}\text{-}3'$
n = 1	$5'\text{-}\bar{C}_1\text{–}\bar{C}_2\text{-}3'$
n = 2	$5'\text{-}\bar{C}_2\text{–}\bar{C}_3\text{-}3'$

The splint strands bring the sequences of one class with sequence of another class and as a result allocation sequences for all the classes are derived (shown in Figure 6). Figure 7 shows one instance of such combinations although in reality several valid and invalid possible combinations are generated.



. Figure 6: Generated allocation pattern for two classes C_1 and C_2.

Step 6: *Select those allotments that start with 1st class and end with last class:*



. Figure 7: An example of generated allocation pattern for classes C_1, C_2 and C_3.

By the end of step 5 several sets of random combinations are generated consisting of both valid as well as invalid strands. PCR reaction is carried out in the test tube with $\bar{C}_1$ and $\bar{C}_n$ as primers which result in strands starting with C_1 and ending with C_3.

Step 7: *Extract allotments that have correct length:*

Gel electrophoresis is carried out during this step and the band corresponding to length 510 bp is extracted. Length of valid assignment to all the three classes together = 3* ($|C_1|$ + $|C_2|$ + $|C_3|$)= 3*170 = 510 bp. Sequences with length more or less than 510 bp are invalid as it represents either repetition of certain allocation or incomplete allocation.

Step 8: *Keep all allotments that have all subjects, time slots and class at least once:*

Affinity purification is carried out on the extracted sequences to discard any conflict of allotment. A conflict of allocation arises if a teacher is allocated to two different classes at the same time slot. To implement this, affinity purification is carried out with 5'-$\bar{T}_m$-$\bar{t}_y$-3'sequences conjugated to magnetic beads. If in any sequence there is more than one T_m-t_y sequence then that strand is discarded. The final end products of this step are those strands which represent a valid feasible solution to the timetable problem.

2.2 Evaluating the algorithm in terms of time complexity

Unlike conventional computing, DNA computing involves manipulation of strands of DNA by using relevant biochemical reactions in test tubes. A single test tube solution contains trillions of nucleotide strands that undergo the same reaction at the same time. For e.g., when a hybridization reaction is executed in a test tube, all of the trillion strands in the test tube undergo the reaction at the same time. However, such computing is ineffective in solving tasks which require excessive sequential operation. This property qualifies DNA computing an excellent candidate for exhaustive search and brute force approach to solve NP complete problems in polynomial time. There are several measures to evaluate an algorithm such as time complexity and space complexity. Time complexity which is evaluated by counting the number of biological operations of the algorithm by considering the complexity of every biological operation as O(1) as it is independent of input size i.e., the number of inputs in the form of DNA strands doesn't matter. The time complexity of an algorithm would be the sum of the time complexity of all steps. The time complexity for each step of the proposed algorithm is given in Table V.

Table V. The time complexity for each step of the proposed algorithm

Steps of the algorithm	**Time complexity**
Step 1	O(1)
Step 2	O(1)
Step 3	O(1)
Step 4	$O(n^2)$
Step 5	O(1)
Step 6	O(1)
Step 7	O(1)
Step 8	$O(n^2)$

Total time complexity of the proposed algorithm = O(1)+O(1)+O(1)+$O(n^2)$+O(1)+O(1)+O(1)+$O(n^2)$ = $O(n^2)$

3. CONCLUSION

In this paper an algorithm is proposed to solve an instance of time table problem. Due to the parallelism property of DNA, the algorithm is quite effective in large scale input scenario. The total number of steps is fixed and is not affected by the number of inputs therefore the overall efficiency is independent of problem size. Time complexity of this algorithm is in polynomial time i.e., $O(n^2)$, but space complexity eventually becomes a restrictive factor which marked an upper bound to the instance of the experimentally solvable problem. The authors would like to acknowledge that, though from a theoretical point of view this algorithm is implementable as all the operations are doable; however experimental difficulty can't be avoided due to inherent reliability issues in wet lab experiments. In the near future it is expected that DNA timetable scheduling algorithm finds its application in several other scheduling problems with necessary modifications.

4. ACKNOWLEDGMENTS

Authors would like to thank School of Innovation and Technology, Assam Rajiv Gandhi University of Cooperative Management (ARGUCOM), Sivasagar, Assam, India for providing necessary facilities.

References

Adleman, L. M. 1994. Molecular computation of solutions to combinatorial problems. *Science Vol.266,* No.5187, pp.1021–1024.

Adleman, L. M. 1998. A sticker based model for dna computation. *Journal of Computational Biology Vol.5.*

Aycan, E. and Ayav, T. 2009. Solving the course scheduling problem using simulated annealing. In *2009 IEEE International Advance Computing Conference.* IEEE, pp.462–466.

Bhaduri, A. 2009. University time table scheduling using genetic artificial immune network. In *2009 International Conference on Advances in Recent Technologies in Communication and Computing.* IEEE, pp.289–292.

Boruah, K. and Dutta, J. C. 2018. An improved generalized dna computing model to simulate logic functions and combinational circuits. *International Journal of Information Technology Vol.10,* No.3, pp.379–390.

Burke, E. K., Elliman, D. G., and Weare, R. 1994. A university timetabling system based on graph colouring and constraint manipulation. *Journal of research on computing in education Vol.27,* No.1, pp.1–18.

Chang, W. L., Lin, K. W., Chen, J. C., Wang, C. C., Lu, L. C., Guo, M., and Ho, M. 2012. Molecular solutions of the rsa public-key cryptosystem on a dna-based computer. *The Journal of Supercomputing Vol.61,* No.3, pp.642–672.

Chang, W. L., Ren, T. T., and Feng, M. 2014. Quantum algorithms and mathematical formulations of biomolecular solutions of the vertex cover problem in the finite-dimensional hilbert space. *IEEE transactions on nanobioscience Vol.14,* No.1, pp.121–128.

Chen, R. M. and Shih, H. F. 2013. Solving university course timetabling problems using constriction particle swarm optimization with local search. *Algorithms Vol.6,* No.2, pp.227–244.

Cheng, Z., Chen, Z., Huang, Y., Zhang, X., and Xu, J. 2010. Implementation of the timetable problem using self-assembly of dna tiles. *International Journal of Computers Communications & Control Vol.5,* No.4, pp.490–505.

Chu, S. C., Chen, Y. T., and Ho, J. H. 2006. Timetable scheduling using particle swarm optimization. In *In First International Conference on Innovative Computing, Information and Control-Volume I (ICICIC'06).* Vol. 3. IEEE, pp.324–327.

Chu, S. C. and Fang, H. L. 1999. Genetic algorithms vs. tabu search in timetable scheduling. In *1999 Third International Conference on Knowledge-Based Intelligent Information Engineering Systems Proceedings*. IEEE, pp.492–495.

Cowling, P., Kendall, G., and Han, L. 2002. An investigation of a hyperheuristic genetic algorithm applied to a trainer scheduling problem. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*. Vol. 2. IEEE, pp.1185–1190.

Even, S., Itai, A., and Shamir, A. 2002. On the complexity of time table and multi-commodity flow problems. In *16th Annual Symposium on Foundations of Computer Science (sfcs 1975)*. IEEE, pp.184–193.

Ganguli, R. and Roy, S. 2017. A study on course timetable scheduling using graph coloring approach. *International journal of computational and applied mathematics Vol.12,* No.2, pp.469–485.

Ghaemi, S., Vakili, M. T., and Aghagolzadeh, A. 2007. Using a genetic algorithm optimizer tool to solve university timetable scheduling problem. In *2007 9th International Symposium on Signal Processing and Its Applications*. IEEE, pp.1–4.

Gotlieb, C. C. 1963. The construction of class-teacher timetables. In *IFIP congress*. Vol. 62. pp.73–77.

Huisman, D. 2006. A column generation approach for the rail crew re-scheduling problem. *European Journal of Operational Research Vol.180,* No.1, pp.163–173.

Li, W. X., Xioa, D. M., and He, L. 2006. Dna ternary addition. *Applied mathematics and computation Vol.182,* No.2, pp.977–986.

Lipton, R. J. 1995. DNA solution of hard computational problems. *Science Vol.268,* No.5210, pp.542–545.

Opera, M. 2007. MAS_UP-UCT: A multi-agent system for university course timetable scheduling. *International Journal of Computers Communications & Control Vol.2,* No.1, pp.94–102.

Pezzella, F., Gianluca, M., and Giampiero, C. 2008. A genetic algorithm for the flexible job-shop scheduling problem. *Computers & Operations Research Vol.35,* No.10, pp.3202–3212.

Popov, I. Y., Vorobtova, A. V., and Blinova, I. V. 2014. Dna-algorithm for timetable problem. *International journal of bioinformatics research and applications Vol.10,* No.2, pp.145–156.

Quyang, Q., Kaplan, P. D., Liu, S., and Libchaber, A. 1997. A dna solution of the maximal clique problem. *International Journal of Computers Communications & Control Vol.278,* No.5337, pp.446–449.

Redl, T. A. 2007. University timetabling via graph coloring: An alternative approach. *Congressus Numerantium Vol.187*, pp.174.

Rowies, S., winfree, E., Burgoyne, R., Chelyapov, N. V., Goodman, M. F., Rothermund, P. W., and Adleman, L. M. 1998. A sticker-based model for dna computation. *Journal of Computational Biology Vol.5,* No.4, pp.615–629.

Sakamoto, K., Gouzu, H., Komiya, K., Kiga, D., Yokoyama, S., Yokomori, T., and Hagiya, M. 2000. Molecular computation by dna hairpin formation. *Science Vol.288,* No.5469, pp.1223–1226.

Sigl, B., Golub, M., and Mornar, V. 2003. Solving timetable scheduling problem using genetic algorithms. In *Proceedings of the 25th International Conference on Information Technology Interfaces*. IEEE, pp.519–524.

Smith, L. M., Corn, R. M., Condon, A. E., Lagally, M. G., Frutos, A. G., Liu, Q., and Thiel, A. J. 1998. A surface-based approach to dna computation. *Journal of computational biology Vol.5,* No.2, pp.255–267.

Wang, Z., Tan, J., Huang, D., Ren, Y., and Ji, Z. 2014. A biological algorithm to solve the assignment problem based on dna molecules computation. *Applied Mathematics and*

Computation Vol.244, pp.183–190.

Wang, Z. C., Huang, D. M., Meng, H. J., and Tang, C. P. 2013. A new fast algorithm for solving the minimum spanning tree problem based on dna molecules computation. *Biosystems Vol.114,* No.1, pp.1–7.

Wang, Z. C., Huang, D. M., Tan, J., Liu, T. G., Zhao, K., and Li, L. 2015. A parallel algorithm for solving the n-queens problem based on inspired computational model. *BioSystems Vol.131*, pp.22–29.

Wang, Z. C., Zhang, Y. M., Zhou, W. H., and Liu, H. F. 2012. Solving traveling salesman problem in the adleman-lipton model. *Applied Mathematics and Computation Vol.219,* No.4, pp.2267–2270.

Winfree, E., Liu, F., Wenzler, L. A., and Seeman, N. C. 1998. Design and self-assembly of two dimensional dna crystals. *Nature Vol.394,* No.6639, pp.539–544.

Wood, D. C. 1968. A system for computing university examination timetables. *The Computer Journal Vol.11,* No.1, pp.41–47.

Xiao, D. M., Li, W. X., Yu, J., Zhang, X. D., Zhang, Z. Z., and He, L. 2006. Procedures for a dynamical system on {0,1}n with dna molecules. *BioSystems Vol.84,* No.3, pp.207–216.

Yin, Z. and Chen, M. 2010. Apply acryditetm gel separation to solve timetable problem. *Indonesian Journal of Electrical Engineering and Computer Science Vol.10*, pp.1111–1116.

Zhang, Y., D'ariano, A., He, B., and Penq, Q. 2019. Microscopic optimization model and algorithm for integrating train timetabling and track maintenance task scheduling. *Transportation Research Part B: Methodological Vol.127*, pp.237–278.

Zhong, J. H., Shen, M., Zhang, J., Chung, H. S. H., Shi, Y. H., and Li, Y. 2012. A differential evolution algorithm with dual populations for solving periodic railway timetable scheduling problem. *Transportation Research Part B: Methodological Vol.17,* No.4, pp.512–527.

Dr. Kuntala Boruah is an Assistant Professor in School of Innovation and Technology, Assam Rajiv Gandhi University of Cooperative Management (ARGUCOM), Sivasagar, Assam, India. She has done her Master of Computer Application (MCA) from Dibrugarh University, Dibrugarh, Assam, India and has a Ph D degree in Electronics and Communication Engineering from Tezpur University (A Central University), Tezpur, Assam, India. Her areas of interest includes: DNA Computing, Image Processing, IoT, etc.

Mr Manash Kapil Pathak is an Assistant Controller of Examination in Mahapurusha Srimanta Sankaradeva Viswavidyalaya, Nagaon, Assam, India. He has done his Msater of Computer Application (MCA) from Jorhat Engineering College, Jorhat, Assam, India. His areas of interest includes: DNA Computing, IoT, etc.

Dr. Ranjan Sharma received Ph.D degree in computer science from Assam University, Silchar, Assam, India in the year 2018. Currently he is working as an Assistant Professor in the School of Innovation and Technology at Assam Rajiv Gandhi University of Cooperative Management, Sivasagar, Assam, India. His main research interests are Artificial Intelligence, Bioinformatics and IoT.