International Journal of Engineering and Technical Research (IJETR)
ISSN: 2321-0869 (O) 2454-4698 (P), Volume-7, Issue-5, May 2017

An Improved Approach for Examination Time Tabling Problem Using Graph Coloring

Md. Atahar Hussain, Mr. Nagesh Sharma, Mr. Ram Kumar Sharma

***Abstract*— This paper referred a graph-coloring approached algorithm for the exam time tabling problem specificities application, with the concrete objective of achieving fairness, accuracy, and optimal exam time period allocation. Through the work, we have considered some assumptions and constraints, closely related to the handling exam time tabling problem, and this mainly driven from experience at various colleges. The performance concern of the algorithm has mention in this paper.**

***Index Terms*— Exam Time Table, Graph Algorithms, Graph Coloring, Exam Scheduling, Performance Improvement, Performance Analysis.**

I. INTRODUCTION

An undirected graph G represented by an ordered pair (V, E) where V represents a set of nodes and E represents a set of edges between nodes. Two nodes (i, j) are adjacent if and only if there is an edge between i & j. The graph coloring is a well-known approach to solve exam time table problem [1, 3, 4, 5, 7]. In graph coloring, assigned colors to the nodes of the graph such that no two adjacent nodes have the same color. Since last decade, Exam Time tabling Problem has been a challenging task that universities faces General graph coloring algorithms are well known and have been extensively studied by researchers [2, 3, 4, 6, 7, 10, 11, 13, 14, 15, 16]. and have been extensively studied by researchers [2, 3, 4, 6, 7, 10, 11, 13, 14, 15, 16].

The challenge is to schedule exams of running courses in university in a limited period of time repeatedly. Schedule the exam time table in a way that avoid the confliction (collapse) of time-slot, i.e. no two or more exams for the same student are allotted at the same time. A fair schedule does not allow to schedule more than two time-slots for a student on one day. In the fairness in schedule, it does not leave a big gap between exams for the students. The exam

scheduling problem can be defined as follows: "We denoted the courses by nodes of a graph,

Where two nodes are adjacent if the two corresponding courses are taken by at least one student. Then, it is required to assign each course represented by a node as time-slot, such that no two adjacent nodes have the same time-slot, in this manner condition that a set of constraints referred in theproblem are also met." We can also solve this problem by using graph coloring mythology.

Md. Atahar Hussain, Research Scholar, Dept. of Computer Science & Engineering, NIET, Greater Noida India Mobile No. 7503965586
Nagesh Sharma, Assistant Professor in Information Technology Department at, Noida Institute of Engineering & Technology GreaterNoida India MobileNo.-9999100436,
Ram Kumar Sharma, Assistant Professor in Information Technology Department at, Noida Institute of Engineering & Technology Greater Noida India MobileNo. 9654624322,

In this way, a mechanism for automatic exam time able generation that can be describe the fairness and accurately in allocating timeslots in examination schedule. As a result, this paper shows a graph-coloring-based algorithm for the exam time tabling application which acquires the objectives of fairness, accuracy, and optimal exam time-slot. The main difference between various studies is the set of assumptions and constraints taken into consideration. Burke, Elliman and Weare [11], followed a similar approach using graph coloring. However, in their algorithm, they pointed only the conflicts without any constraints. Moreover, the algorithm pointed in [11] only minimized the conflicts rather than eliminating them. In this paper, we pointed important assumptions and constraints, closely related to the exam Time tabling problem that driven from the real life requirements collected from various universities. Such assumptions and constraints are distinct from other graph coloring problems. We have summarized these assumptions and constraints as follows:

1. The number of time-slot (TS) per day (exam period) can be set by the administrator. TS depend on department specific constraints. For example, a university that uses a 3-hours exam period and begins the exam day at 9:30 am and finish at 5:00 pm, may set TS to 2.

2. The number of concurrent exams or concurrency level (N_P), depends on the number of available rooms, and the availability of faculty member to conduct the examination. N_p is determined by the registrar's office. Here N_p is a system parameter and the exam scheduling algorithm has been examined with several N_P values.

3. A student shall not allow for more than (x) exams per day and this is referred as a system tunable parameter.
4. A student shall not have a gap of more than (y) days between two consecutive exams, and it also determined by department (another fairness requirement).
5. The schedule shall be done in the minimum possible time-slot.. The exam schedule is an outcome of the scheduling algorithm.
6. Now, we give some relevant definitions to the underlined problem. Let N be a list of all courses to be scheduled. The length of this list is n.

A course at position i in the list N is referred to using an index N_i. Let G be the graph that represents the list N of courses. We impose a weight W_{ij} to each edge of graph G, where W_{ij} is defined as the number of students present in both courses N_i and N_j. An edge e_{ij} exists between nodes N_i and N_j iff W_{ij} is not 0. We define a weight matrix W to be an NXN matrix, where n is the number of courses to be scheduled for the exams, and W_{ij} equals the weight of the edge e_{ij} that joins the courses N_i and N_j. Such a weight on the edges of graph G represents the exam conflict complexity in courses N_i and N_j.

The degree D_i of a node N_i is defined as the number of edges connected to that node. A large degree of a node N_i indicates that there is a large number of students registered in this course. The degree D_i is also a measure of conflict complexity. An example of a weighted graph G and the corresponding weight matrix W is given in Figure 1 and Table 1, respectively. In Figure 1, N_2 and N_5 both have degree 3. In Table 1, the weight of the edge e_{15} is 6.

II. The Coloring Scheme Description

The coloring scheme for the exam time tabling problem uses a double indexed color (C_{AB}), where the index (A) represents the day of the exam and (B) represents the exam time-slot on a given day. The range of (B), show the number of exam time slots is allocated by the registrar.

The range of the index (A) is a parameter generated as an outcome by the algorithm. Objective of algorithm is to minimize the index (A). The parameter A is also be set by the registrar and administrator. It also is bound by optimal number of colors for the given graph. However, finding optimal is known to be NP complete. The algorithm imposed in this paper is to achieve optimal performance (minimal number of colors) in polynomial time.

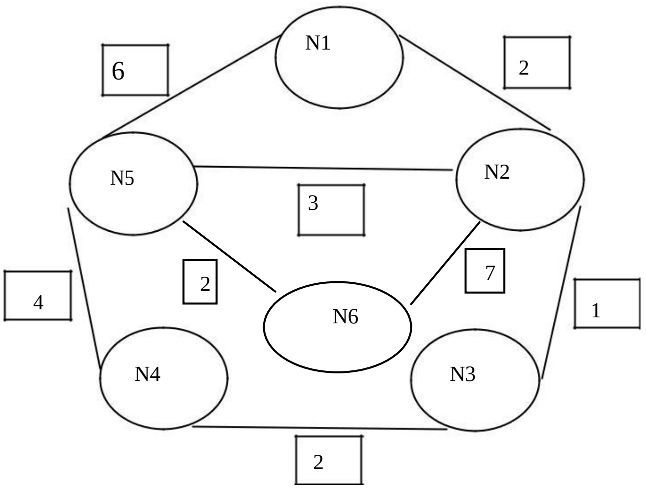

Figure 1. A weighted graph G.

Table 1. A weight matrix W of the graph.

	N_1	N_2	N_3	N_4	N_5	N_6
N_1	0	2	0	0	6	0
N_2	2	0	1	0	3	7
N_3	0	1	0	2	0	0
N_4	0	0	2	0	4	0
N_5	6	3	0	4	0	2
N_6	0	7	0	0	2	0

We define the weight of a color to be W (C_{AB}) = (A-1)*l + B; l is the range of B. a color C_{AB} is said to be smaller than color C_{YZ} if the weight W (C_{AB}) is smaller than W (C_{YZ}). The coloring scheme allows two or more non-adjacent nodes to have the same color (C_{AB}). The number of nodes having the same color referred the number of concurrent exam time-slot, which is bounded by the number of available rooms and the maximum allowable concurrent sessions by the administrator. In graph coloring problems, there is no restriction on the assignment of the same color to non-adjacent nodes in the graph. The exam-time tabling problem as explained above imposes a constraint on the maximum number of nodes assigned the same color. The Time scheduling algorithm allows the user to impose a maximum limit on the number of available color C_{AB}. The number of instances of a color C_{AB} is referred to as the concurrency limit of the color C_{AB} denoted CL (C_{AB}). Note that a course with multiple sections is assigned one color. However, the multiple sections will consume multiple instances of the same color, assuming that each section will make the exam in a separate room.

2.1 Fairness and Accuracy of Algorithm

To achieve fairness, the algorithm defines the following parameters:

1. Internal distance (D_I): This is the distance between two colors (C_{AB}) and (C_{AC}) with the same index (A) and indexes B and C, and defined by

$$D_I = |C-B| \qquad (1)$$

D_I represents the exam scattering on the same day A for the same set of students.

2. External distance (D_E): This is the distance between two colors (C_{AB}) and (C_{CD}), and defined by

$$D_E = |C-A| \qquad (2)$$

D_E represents the exam scattering across different days.

3. Total distance between colors (C_{AB}) and (C_{CG}) is given by

$$D = \gamma * D_E + D_I, \qquad (3)$$

OR

$$D = \gamma *(|C-A|) + |G-B| \qquad (4)$$

The factor (γ) can be varied for different coloring scheme. Here distance D is a design parameter of the algorithm.

2.2. Important Specialized Considerations

Time tabling problem has consideration at the implementation level. For example, node with large degree represents a course in which many students are registered to many other courses. Also, nodes with large degrees have large number of students. In order to have an efficient time schedule; the nodes with larger degrees should be colored first. By Giving priority to the nodes with the larger degrees which tend to schedule the university required courses early in the exam period. The weight of an edge depicted the number of common students registered at both courses (nodes) connected to that edge. Giving priority in the coloring algorithm to nodes connected to a large weight-edge will enable a solution optimization towards the larger groups of students. Another point to consider before we evaluating the algorithm is the multi-section courses. Multi-sections of a multi-sections course should be allocate at the same time, and thus, the corresponding nodes should have one color. And also, they occupy several rooms. The number of rooms used by a course has an impact on the concurrency level per time slot. When such multi-sections are scheduled for a time slot a color, the concurrency level is to be reduced by the number of sections for that course. For implementation purposes, we depict the nodes of the graph with a value equal to the number of sections in the course; we shall call this value the course concurrency level CL (N_i). Thus, we assign a concurrency limit for each color N_p (C_{AB}).

After assigning a color to a node N_i, we reduce the concurrency limit of the color by CL (C_i). The concurrency limit is set by Administrator and depends on the number of available rooms, and staff to monitor the exams.

III. Algorithm for Graph Color Schedule

The algorithm consists of two major steps. The first step is to builds the weight matrix and graph. The second step is assign colors to the nodes of the graph.

3.1 Main Algorithm

1. Draw Weight Matrix and Graph

1. Show files for students and listing all courses studies by student, which is use for scheduling for the examination. Each course corresponds to a node in the matrix. Set the concurrency level of each node to the number of sections for the given course.

2. In this case, each node implies the course, and find the set of adjacent nodes, including the weight of the edges connecting the node to its adjacent nodes. Fill the weight matrix W_M with weight values W.

3. Create an undirected graph using the weight matrix.

4. Find the degree for each node in the graph.

2. Color the Graph

Arrange the nodes in the weight matrix in a descending order on the basis of degree of nodes. Nodes with similar degrees are ordered based on the largest weight W in its adjacency list. Nodes with similar degrees D and weights W are ordered based on their node id (smallest ID first).

```
Set sortedCourse = The sorted list of nodes
mentioned in above  Step 1.

Set numberOfColoredCourses = 0
For  i = 1 to sortedCourseLength do
Begin
   If  numberOfColoredCourses =
   numberOfCourses then exit loop and
   finish
  If $N_i$ is not colored then
   Begin

     If i = 1 then
     Begin
        $C_{ab}$ = getFirstNodeColor ($N_i$)
        If $C_{ab}$ = null then Exit and finish, {No
       schedule is possible.}
     End
     Else
     Begin

      $C_{ab}$ = getSmallestAvailableColor ($N_i$) End

   If $C_{ab}$ != null then
    Begin

     Set Color ($N_i$) = $C_{ab}$
       numberOfColoredCourses =
       numberOfColoredCourses + 1
     CL ($C_{ab}$) = CL ($C_{ab}$) - CL ($N_i$) End
 End

 Set Array M = getOrderedAdjacencyCoursesOf$N_i$
  ()

 For j = 1 to numberOfCoursesInArrayM do Begin

    If $M_j$ is not colored then
     Begin
      $C_{cd}$ = getSmallestAvailableColor ($M_j$) If
      Color$_{cd}$ != null then
        Begin

         Set Color ($M_j$) = $C_{cd}$
         numberOfColoredCourses =
         numberOfColoredCourses + 1
         CL ($C_{cd}$) = CL ($C_{cd}$) - CL ($M_j$) End

End End End
```

3. Color the neighbour node

1. Description of Sub_routine “getFirstNodeColor”:
   Input : The course N_i that needs to be Colored.

   Output: The color assigned to N_i or null.
   Algorithm:

```
For a = 1 to maxScheduleDays do: For b =
  1 to numberOfTimeSlots do:
    If CL (Color$_{ab)}$) $\geq$ CL ($N_i$) then return Color$_{ab}$ return
     null
```

2. Description of Sub_routine “getSmallestAvailableColor”:
   Input: The course N_i that needs to be colored.
   Output: The color assigned to N_i or null.
   Algorithm:

```
 get AL($N_i$), the AdjacencyList of $N_i$ For j
= 1 to maxScheduleDays do Begin
   For k = 1 to numberOfTimeSlots do:
     Begin
       Set valid = true
       For r = 1 to Length (AL ($N_i$)) do
          Begin
            $C_{EF}$ = Color ($AL_r$)

            If $C_{EF\,!}$ = null then
              Begin

                 If E! =j or F! =k then
                  Begin

                    If DE {( $C_{EF}$), ($C_{jk}$)} = 0 then
                      Begin

                        If DI {( $C_{EF}$), ($C_{jk}$)} <= 1 then
                          Begin

                             Valid = false
                          Exit loop End

                      End
                 If CL ($C_{jk}$) <= CL ($N_i$) then
                    Begin

                      Valid = false
                    Exit loop End

                If check3ExamsConstraint ($N_i$, $C_{jk}$ , j) = False
                 then
                   Begin
 Valid = false Exit loop
                 End
               End

               Else
               Begin

 Valid = false Exit loop
               End
               End

         Else Exit the current iteration of loop End

 If valid = true then Return $C_{jk}$
 End

 End return
null
```

3. Description of Sub_routine "check3ExamsConstraint":

Input : The course N_i that needs to be colored.
The color C_{AC} that needs to be tested.
The day j for $Color_{kd}$

Output: returns true if color is valid, Otherwise it returns false Algorithm:

```
get a list of students Si registered in course $N_i$

For r = 1 to Length (Si) do:
 Begin
   Set Counter = 0

   For q=1 to NumberOfTimeSlots do:
    Begin
      Get a list of courses CRS assigned to $C_{jq}$ For
      u = 1 to Length (CRS) do:
        Begin
          Get a list of students $S_u$ registered in
          course $N_u$

          If $Si_r$ exists in list $S_u$ then
           Begin
             Counter = Counter + 1

             If Counter = 2 then return false End

        End End
   End
  return true
```

3.2. Complexity Analysis

A. We assume that largest degree D = D_N; and that node v_1 has degree K_1

A1. Step 1 assigns the smallest color, say N_1 to node v_1. The total number of steps required to color all the nodes in the neighbor list of v_1 is
$1+2+3+ \dots + D_N = (D_N{}^2+ D_N)/2 = O(D_N{}^2)$

A2. Repeat the coloring procedure for the next node v_2 with degree D_2. The number of steps required to color all the nodes adjacent to node v_2 is
$1+2+3+ \dots + D_2 = (D_2{}^2+ D_2)/2 = O(D_2{}^2)$

B. In general, the number of steps required to color all the nodes in the neighbor list of any node v_i with degree d_i is
$(D_i{}^2+ D_i)/2 = O(D_i{}^2)$

C. Let the average degree of nodes be ρ. Then the average number of steps required to color the neighbors of node v_i with degree ρ is $O(\rho^2)$

C1. Repeat the coloring procedure in step 1 and step 2 until all nodes are colored.
C2. Since each coloring step colors on the average ρ nodes, the coloring procedure will be repeated on the average (n/ρ), where n is the number of nodes.
C3. The total number of coloring steps required to color all nodes, on the average is

$$O((n/\rho).(p\ 2)) = O(n.\rho)$$

The complexity equation (above) can expressed as

$\sum_{i=1}^{n} \rho$, where ρ P= $(\sum_{i=1}^{n} D_i)/n$.

3.3. Algorithm Efficiency Analysis

Our algorithm has a linear complexity, except when (ρ = n-1) and hence a polynomial solution of the second degree. We prove the following:

Lemma: The algorithm described above achieves the minimum number of colors, when the upper bound of colors is given by the clique (largest completely connected sub-graph).

Proof: A completely connected graph with size P requires P+1 color. The algorithm detects the clique in the graph. The algorithm also detects the clique related to each node in the graph starting from the node with the largest degree. Then, the algorithm use to colors the largest completely connected sub-graphs first, thus utilizing the minimal available colors to color in the sub-graphs. For each node, the algorithm will not use more colors than those required by the largest completely connected sub-graph. Thus the largest number of colors used by the algorithm is only that required by the largest sub-graph, which is the absolute minimal possible number of colors.

IV. Performance Analysis

The algorithm for examination scheduling using graph coloring approach was applied to a course list of a educational institution. The number of courses in the test is 150 with an average of two sections per course, for a total of 700 exams per session to schedule. The graph produced for the courses has an average degree of 28 and a maximum degree of 334. The coloring algorithm completed in 60 seconds. This algorithm has implemented in Python. We ran the algorithm with different parameters. Number of exam time-slots per day (2, 3, 4, 5) treats as variable. In this way, concurrency limit is varied between 8 and 96. The most important constraint is that a student cannot have more than two exams per day. The administrator can plots for decide on the number of days and number of exam sessions per day for examination schedule. For example, with 2 exam slots per day, the exam period can be completed in 30 days with 40 sessions per day.

Note that administrator can produce several schedules in a short period of time and select the appropriate schedule. Note that the execution time is a linear function of the number of courses. The average degree does not increase at the same rate as the number of courses. Furthermore, we have tested our algorithm against 5 samples of the 13 Toronto data sets collected from 13 institutions. In this test, we have taken two factors under consideration, namely as the number of slots and the penalty. With respect to former factor, that show that their results slightly outperform. With respect to the recent factor, we get closed results are obtained, in this case, our algorithm have beat in some sets, and has been beat in some other sets. The results of this algorithm are shown in Table-2, and results plotted in Figure-4 and Figure-5, respectively. In Toronto case, some drawback is mentioned here such as there was nothing mentioned about the maximum possible concurrent exams per time-slot, which does not impose a constraint on issue. In our case we included. Also, in Toronto case, another drawback has nothing mentioned about the number of days, only, they use number of time slots. So, when we run to compare with respect to this factor, we have counted number of time-slots used in all days of our algorithms to get the total number of time-slots used by the examination schedule.

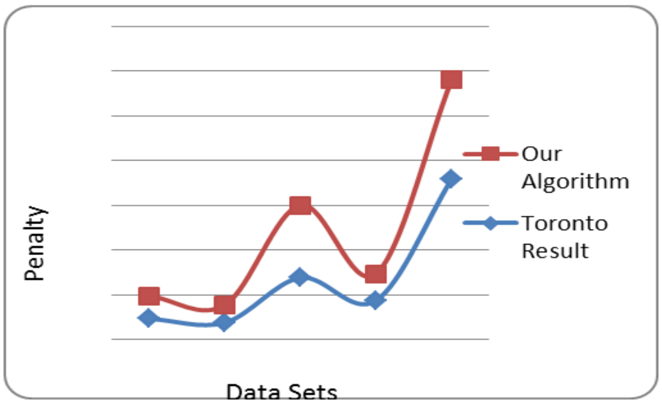

Figure 2. Results with respect to Slots

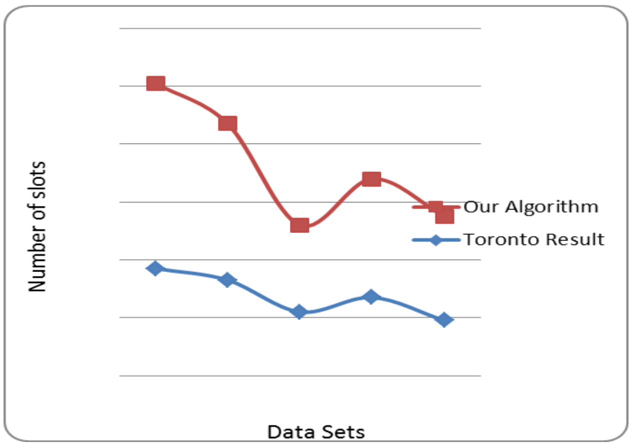

Figure 3. Results with respect to penalty.

Benchmark Data	Toronto Results		Our Algorithm	
	Slots	Penalty	Slots	Penalty
ZOO91	37	4.68	64	5.10
BOT92	33	3.83	54	3.95
STAT93	22	13.85	30	16.21
MAT94	27	8.75	41	6.01
PHY95	19	35.82	36	22.20

Table 2. Results of our algorithm vs Toronto results.

V. Conclusion and Future Work

The number of concurrent exam or concurrency level (N_p) depends on the number of available rooms and availability of faculty member to conduct the exams. The value of N_p is usually assigned by the administrator and this paper assumes that N_p is a system parameter, and we will run the Time table schedule algorithm with several N_p values. In a later work, the actual distribution of exam time-slot to rooms will be included.
This algorithm imposed in this paper is to achieve near optimal performance by choosing minimal number of colors in polynomial time. We are currently finding a modification of the algorithm, which will achieve the absolute minimal for a certain set of graphs.

REFERENCES

[1] Bang-Jensen J. and Gutin G., Digraphs: Theory, Algorithms and Applications, Springer-Verlag, 2000.

[2] Bean D., “Effective Coloration,” The Journal of Symbolic Logic,

[3] Alon N., “A Note on Graph Colorings and Graph Polynomials,” Journal of Combinatorial Theory Series B, vol. 70, no. 1, pp. 197-201, 1997.

[4] Baldi P., “On a Generalized Family of Colorings,” Graphs and Combinatorics, vol. 6, no. 2, 1990.

[5] Bar-Noy A., Motwani R., and Naor J., “The Greedy Algorithm is Optimal for On-line Edge Coloring,” Information Processing Letters, vol. 44, no. 5, pp. 251-253, 1992.

[6] Husseini S., Malkawi M., and Vairavan K., “Graph Coloring Based Distributed Load Balancing Algorithm and Its Performance Evaluation,” in the 4th Annual Symposium on Parallel Processing, 1990.

[7] Batenburg K. and Palenstijn W., “A New Exam Timetabling Algorithm,” Leiden Institute of Advanced Computer Science (LIACS), http://visielab.ua.ac.be/staff/batenburg/papers/bapa_bnaic_2003.pdf.

[8] Husseini S., Malkawi M., and Vairavan K., “Distributed Algorithms for Edge Coloring of Graphs,” in the 5th ISMM International Conference on Parallel and Distributed Computing Systems, 1992.

[9] Jensen T. and Toft B., Graph Coloring Problems, Wiley-Interscience, 1995.

[10] Burke E. and Petrovic S., “Recent Research Directions in Automated Timetabling,” European Journal of Operational Research (EJOR), vol. 140, no. 2, pp 266-280, 2002.

[11] Burke E., Elliman D., and Weare R., “A University Timetabling System Based on Graph Coloring and Constraint Manipulation,” Journal of Research on Computing in Education, vol. 27, no. 1, pp. 1-18, 1994.

[12] Burke E., Elliman D., and Weare R., “Automated Scheduling of University Exams,” Department of Computer Science, University of Nottingham, 1993.

[13] Christofides N., Graph Theory: An Algorithmic Approach, Academic Press, 1975.

[14] Gross J. and Yellen J., Handbook of Graph Theory, Discrete Mathematics and its Applications, CRC Press, vol. 25, 2003.

[15] Husseini S., Malkawi M., and Vairavan K.,“Analysis of a Graph Coloring Based Distributed Load Balancing Algorithm,” Journal of Parallel & Distributed Systems, vol. 10, no. 2, pp. 160-166, 1990.

[16] Bauernoppel F. and Jung H., “Fast Parallel Vertex Coloring,” in L. Budach (eds.), Fundamentals of Computation Theory, FCT '85, Cottbus, GDR, Sept, 985, vol. 199 of Lecture Notes in Computer Science, pp. 28-35, Springer-Verlag, Berlin, 1985.

Md. Atahar Hussain, Research Scholar, Dept. of Computer Science & Engineering, NIET, Greater Noida India Mobile No. 7503965586

Nagesh Sharma, Department Name Information Technology, Noida Institute of Engineering & Technology Greater Noida MobileNo.-9999100436

Ram Kumar Sharma, Department Name - Information Technology, Noida Institute of Engineering & Technology Greater Noida MobileNo.9654624322.