Decision Tree Based Routine Generation (DRG) Algorithm: A Data Mining Advancement to Generate Academic Routine and Exam-time Tabling for Open Credit System

Ashiqur Md. Rahman
North South University, Computer Science and Engineering Department, Dhaka, Bangladesh
Email: ashiq_rahman@yahoo.com

Sheik Shafaat Giasuddin and Rashedur M Rahman
North South University, Computer Science and Engineering Department, Dhaka, Bangladesh
Email: rshafaat@gmail.com, rashedurrahman@yahoo.com

Abstract—**In this paper we propose and analyze techniques for academic routine and exam time table generation for open credit system. The contributions of this paper are multi-folds. Firstly, a technique namely Decision tree based Routine Generation (DRG) algorithm is proposed to generate an academic routine. Secondly, based on the DRG concept, Exam-time Tabling algorithm (ETA) is developed to implement conflict free exam-time schedule. In open credit course registration system any student may choose any course in any semester after completion of pre-requisite course(s). This makes the research more challenging and complex to accomplish. Academic routine and exam time-table generation are in general *NP*-Hard problems, i.e., no algorithm has been developed to solve it in reasonable (polynomial) amount of time. Different methods based on heuristics are developed to generate good time-table. In this research we developed heuristic based strategies that generate an efficient academic routine and exam time-table for a university that follow open credit system. OLAP representation helps to classify the courses along with the proposed algorithm to eliminate some constraints. Day-based pattern, minimum manhattan distance between courses of same teacher; minimum conflicted course distribution has been stage-managed to classify the courses. Our ETA algorithm is based on decision tree and sequential search techniques.**

Index Terms— **OLAP, Crosstable, Conflict List, Favorite Slot, Faculty Choice, Course Color, Day-time slot pattern.**

I. Introduction

This paper depict Decision Tree based Routine Generation (DRG) algorithm to generate a university class routine within a tolerable range of some constraints and conflict free Exam-Time Tabling algorithm (ETA). A decision-tree based classification algorithm has been introduced to solve this *NP*-Hard Problem [22]. CPL (constraint logic programming) is a respected technology for solving hard problems which include many (non-linear) constraints [1]. Constraints propagation technique has been applied to overcome the preferential requirements for slots of teachers, courses from pre-advising by students and class room allocation. Versatile choices for courses may lead to a deadlock situation. Golz used priorities heuristic ordering [2] where Abdennadher introduced an optimized cost-based rule mining [3,4] to solve these type of problems. On the other hand, knowledge based in a hyper heuristic course scheduling using case based reasoning is used to maximize the rule covering area [5]. Further expansion is possible to accomplish the exam-time tabling using OLAP technique [16]. Exam-time tabling is another highly constrained combinatorial optimization problem. The major objective is to confirm 100% conflict free exam schedule with a fixed interval of days. Limited room capacity and room availability problems must be overcome to place exams on each time slot. The computational time is reduced by using heuristic based search in comparison with the permutation of courses for exam-time tabling. Identification of a novel heuristic is the most challenging task. Using OLAP, proposed conflict free Exam-Time Tabling algorithm (ETA) produces substantial results to accommodate all students with zero conflict tolerance.

DRG presents key features of generating class routine with minimum computational time. Heterogeneous distribution of courses is classified with maximum satisfaction of all constraints. Section II describes about previous related works and preliminaries in details. Section III illustrates the problem dimensions with the data filtering technique used in the paper to summarize further manipulation; pursued by the proposed algorithm, DRG, and the classification procedure to find the feasible solution in Section IV followed by Exam-Time Tabling algorithm (ETA) in section V. Extensive computational

doi:10.4304/jcp.5.1.12-22

results are conducted to study the performance analysis for both algorithms in section VI. Finally the paper concludes the work in section VII.

II. Preliminaries & Related Work

A university class routine generation problem – as considered in this work – consists of assigning each course in a set of slots (classes) in a limited class rooms within the teachers' favorite time slots. This highly constrained problem is optimized by simulated annealing and genetic algorithm [6,7]. Seven different major and minor objectives are discovered and deadlock situation is overcome by randomly exploring the composite neighborhood [8]. The most closely related attempt with this work appears to be the constraint programming approach used by Boizumault [9] and the simulated annealing approaches explored by Dowsland and Thompson [10,11]. The principal innovation in DRG is the sequential use of these two methods. DRG may select some poor slots, with respect to teachers or students, under a tolerable conflict range. A similar sequential approach has been taken in DRG on other problems: White and Zhang [12] used constraint programming to second a starting point for tabu search in solving course timetabling problems. For a high school timetabling Yoshikawa tested several combinations of two stage algorithms, including a greedy algorithm followed by simulated annealing and a constraint programming phase followed by a randomized greedy hill climbing algorithm (which is deemed to be the best combination of those used). In a similar vein, Burke, Newell and Weare [13] used their work on sequential construction heuristics [14] to generate initial solutions for their memetic algorithm [15].

Wide variety of courses increases the emergence to provide adequate exam-time tables for the educational institutions. The development of an examination timetable requires the institution to schedule a number of examinations in a given set of time slots, so as to satisfy a given set of constraints. A common constraint for any educational institution is that none of the students can have more than one exam scheduled at the same time. Many other constraints were presented by Marlot in [17]. Sequential construction heuristics have been applied to the publicly available data in a variety of forms by selecting exams from a randomly chosen subset of all exams by Burke [14] where Carter [19, 20] allow limited backtracking de-allocation of exams. On the other hand Caramia [18] includes an optimization step after each exam allocation. Sequential construction heuristics order the exams in some way and attempt to allocate each exam to an ordered session by satisfying all the constraints. Using a memetic algorithm for exam timetabling Burke, Newall and Weare [15] proposed a hybrid algorithm consist of a simulated annealing phase to improve the quality of solution, and a hill climbing phase for further improvement. To avoid local maxima problem these solutions require random jitter [21] whereas the proposed algorithm has no impact on randomization.

III. Problem Description & Data Filtering

The routine maps a set of courses chosen by students and teachers to a specific room and time-slot. A major objective, in developing an automated system, is to minimize the hassle of separating conflicted courses from choices by students. In this paper the major identified problems are (a) Number of lectures per week for each course are fixed, (b) Room overlapping is prohibited, (c) Fitting the routine with teacher's favorite timeslots, and (d) trying to assert different timeslots to same level of courses. On the other hand, the minor objectives are (e) day-timeslots pattern for the course, (f) room capacity, (g) avoiding gaps between classes of same teacher, if possible, (h) single class for student per day and (i) ensuring compactness of interclass time difference.

The required scattered data contains total courses (course choices from pre-advising by students) $C = \{c_1, c_2, c_3, \ldots, c_n\}$ where the dependencies between courses are also maintained. Here course dependency can be defined as $\exists C_i, C_i \rightarrow C_j$ where $C_i, C_j \subset C$. For this paper the students', $S = \{S_1, S_2, S_3, \ldots, S_z\}$, course choices can be derived as $S_j = \{c_i\}$ where $\exists$ i, $c_i \in C$ and $1 \leq i \leq n$ and $|S_j|$ = *max_course_choice* for the student as shown in Fig. 1. Teachers' favorite timeslots are grouped according to day-timeslots pattern. Here group A and B is formed for the teacher t_k, where $T = \{t_1, t_2, t_3, \ldots, t_m\}$, $A(t_k)$ = {favorite time_slots of t_k | sequential time slots for Saturday, Monday and Wednesday} and $B(t_k)$ = {favorite time slots of t_k | sequential time slots for Sunday, Tuesday and Thursday}, whereas time_slots = {1, 2, …, 30} contains 5 sequential slots per day starting from Saturday. Here Friday is considered as an off-day. Priority of the teacher has been introduced by $P(t_k) = \{1, 2, \ldots, 10\}$ where a higher value represents higher priority. An exceptional priority is also introduced as 11 reflecting part-time faculty, whose projected time cannot be changed. Fig. 2. and Fig. 3. shows the teachers' wish-list and the course distribution among the teachers'. Target of this work is to find the values of the "*class slot routine*" field of the Fig. 3. The resultant routine vector, $V = \{\{c_i\}_q\}$ $\forall$ i, $c_i \in C$ and $1 \leq i \leq n$ and $1 \leq q \leq 30$, consists of the course classification as per day required for each course and class room availability.

In this paper, this huge dimensionality of dataset is reduced by initiating an OLAP (On-Line Analytical Processing) representation [16]. Here a *Crosstable* (Cr) of (n × n) courses are initialized. Cr (n × n) = $\{conflict_{i,j}\}$ where $1 \leq i,j \leq n$ and 'n' is the total number of courses requested by the student (or is 'n' number of courses offered by the department). Here the conflict of $Cr_{i,j}$ is a positive integer that reflects the common students between c_i and c_j. The diagonal values of $Cr_{i,i}$ show the total number of students requesting for the course c_i. $\sum Cr_{i,j}$, $[1 \leq j \leq n]$ is the total conflict for the course c_i [$\forall$ i, i $\neq$ j]. Maximum "chaos" (conflict) courses can be easily sorted out from this two dimensional conflict distribution (*Crosstable*).

To minimize the potential for time conflict, an admissible heuristic (*h*) is imposed to regroup the courses according to their dissimilarity. Graph Coloring algorithm is used to cluster where the conflict in *Crosstable* confirms the weights of the edges in the graph. Here the admissible heuristic is applied as the maximum number of colors needed to find the minimum number of groups of courses. Here faculty redundancy is also considered as weight, that is, same faculty of different courses cannot be in the same group.

Student ID	Course ID	Semester ID
S_1	C_1	1
S_1	C_3	1
S_2	C_1	1
S_2	C_3	1
S_3	C_2	1
S_3	C_4	1
S_3	C_5	1
S_4	C_4	1

Figure 1. Courses Pre-Advised by the Students.

Teacher ID	Favorite Slots	Priority
t_1	7,8,9,13,17,18,19,22,23,24	7
t_2	2,12,22,29	11
t_3	5,10,15,20,25,30	9
t_4	2,5,8,9,12,15,18,22,23,24,25	10

Figure 2. Teachers' Slots Preferences along with Priority.

Teacher ID	Course ID	Class Slot Routine	Sem. ID	Class per Week
t_1	c_1	7,17	1	2
t_1	c_4	8,18,24	1	3
t_2	c_2	2,12	1	2
t_3	c_3	5,15	1	2
t_4	c_5	5,15,25	1	3

Figure 3. Courses and Classes Distribution for the Teachers.

The output of Graph Coloring Algorithm assigns a color to all courses individually in Fig. 5(a); same colored courses are treated as a group. Fig. 4. shows the resultant *Crosstable*. Each group may consist more than the threshold limit members with tolerable conflict range. Here the number of the rooms is considered as the threshold value. In this work the tolerable conflict range is set to 0.

Course ID	c_1	c_2	c_3	c_4	c_5	Total Conflict
c_1	**40**	5	7	0	0	**12**
c_2	5	**50**	0	2	1	**8**
c_3	7	0	**35**	5	0	**12**
c_4	0	2	5	**40**	1	**8**
c_5	0	1	0	1	**45**	**2**
	t_1	t_2	t_3	t_1	t_4	

Figure 4. *Crosstable* for n × n Courses.

This easy formation of coloring may lead to a measure of the undesirability of having classes overlapped in the routine. It will be effective to try to fit the most "chaos" courses of high priority teachers in the routine first. Random selection may be used to select teacher having same priority. The data used in this work to test the algorithm is real.

Constraint programming model and data filtering techniques for routine generation motivate the increasing interest to develop an exam-time tabling. In routine generation algorithm the colored courses refer the non-conflicting sets of courses. Faculty redundancy is not considered as constraint any more but the room capacity. For ETA the graph consisting edge weight between two courses $D_{y,z} = C_y + C_z - Cr_{i,j}$ where C_y and C_z represents the number of students of C_i and C_j courses respectively.

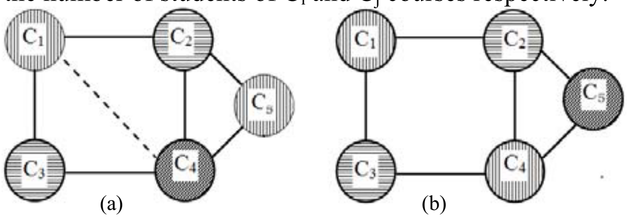

Figure 5(a). Courses Graph Color for DRG. (b). Courses Graph Color for ETA.

$\exists i,j \; \sum D_{y,z} \leq$ total_room_capacity and $|D_{y,z}| \leq$ room_capacity shown in Fig. 5(b). Important factor of grouping courses is that the number of members in each group must not exceeds the total number of room availability. The minimum requirement of days for 100% concurrent conflict free exam is greater than or equal to the number of groups that is the numbers of color requires coloring the graph. If each day consists of more than one slot of examination and the provided day is less than the numbers of color then the *Crosstable* is able to ensure the numbers of consecutive examinations for an individual or groups of students on the day. Each exam slot holds a group of courses only, with zero conflict. But the consecutive slot may embrace some conflicts among the groups due to differ in color. By using dynamic data structure the number of consecutive examinations between different groups of courses can be easily sorted out described in section V.

IV. THE DECISION TREE BASED ROUTINE GENERATION (DRG) ALGORITHM

The aim of the proposed DRG algorithm is to classify all the courses with a degree of satisfaction. Enormous permutation of courses may lead to a time consuming process. So a standardized branch & bound condition may be applied to reduce the problem surface area. The DRG sequentially follows 4 sets of cascading decision trees. Depending upon the emergence and success rate, the result of one tree is propagated to another tree as shown in Fig. 6. These transitions may lead to a solution but also may degrade the satisfaction threshold. A control portion helps to justify the problem solution needed to be more explored or not.

Each transition from one decision tree to another shrinks the overall problem surface area by eliminating the classified courses. Classical decision tree takes certain decision depending upon some gain factor. Beside this, the proposed trees concentrate on the reduced problem dimension which helps to classify the unclassified courses with tolerable time complexity.

The key factor for each of the four decision trees is (a) PDRG: Teacher Priority, (b) CDRG: Highest conflicted course, (c) TDRG: Tolerable conflict and (d) NTRG: Neighbor slots by ignoring teacher's wish list. A university routine is created and remains unchanged for a particular semester. If the placed courses do not match the favorite slots of the teacher, the evolved dissatisfaction is much higher for a higher prioritized teacher. So, we used PDRG as our first decision tree. From an observation it is clear that the placing conflicted course in a dense routine vector is difficult as it can introduce student conflict in the routine. So, it will wise to consider the higher conflicted courses first as they are the principle component which reflects the major problem surface area. By doing this the problem surface area is reduced easily. For this reason CDRG is the second selection. The decision tress TDRG and PDRG are quite similar. But TDRG introduces considerable student conflict in to the routine vector.

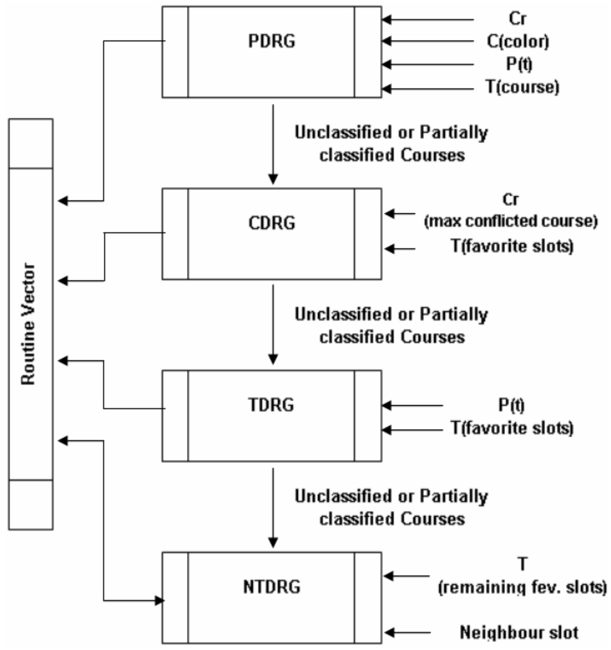

Figure 6. Program flow of DRG

Therefore, TDRG plays the third role in the whole algorithm. After all the three decision trees, the unclassified and partially classified courses shows, there is no place (slot) according to the teacher's favorite slots. Exploring over the contour of the teachers favorite slots is necessary in order to achieve the course's class per week constraints. Only student conflict is considered in TDRG where the day-time pattern is maintained strictly. On the other hand, NTDRG will dissatisfy the teachers & may not follow the day-time pattern. Hence, NTDRG is the final decision tree.

A. Priority regulated DRG (PDRG)

The first decision tree accumulates the high prioritized teacher t_k and most conflicted course c_i of t_k to the routine vector with maximum fitness value. The *max_fit* function tries to discover the day-time pattern for the course. If the consequential day-time slots are already occupied by other courses, it checks the corresponding course color. If the color of the courses is same it classifies the course c_i with a degree. Here the fitness value of a course is referred as degree. The highest returned degree of a day-time slot, as maximum fitness value, is selected as the course class for c_i. This operation provides a pattern-based course distribution, $A(t_k)$ or $B(t_k)$, in the routine although some courses may be placed partially as per their *class_per_week* and *max_class_per_slot* constraints. In this manner, the low priority teachers may suffer by not getting the classes in a sequential manner. But in practice 26% of the total courses can be placed with zero conflict and with a high level of satisfaction. The level of satisfaction is a quantitative measure of placement for a course with respect to the teachers' favorite slots. The definitive PDRG tree is shown in Fig. 7. The partially placed and not yet placed courses then elected as cascaded input to second level of exploration. The pseudo code presented in algorithm 1 describes actions to be taken by Priority regulated DRG algorithm (PDRG). The computational time of this operation requires O ($n \times m$) where the maximum number of courses per teacher is 'n' and the number of faculties is 'm'.

```
PDRG( )
{
// select the high prioritized teacher;
// select most conflicted course of the teacher;
// select the corresponding faculty's low frequent favorite
// time slot
   1. Find max-patterned day time slots;
      IF the time slot is empty PLACE the course;
   2. ELSE find the color of the course that already placed
      on the slot;
      2.1. IF the color is same AND on the range of the
           room AND the course is not already placed on
           that day before, PLACE the course;
      2.2. ELSE select the next max-pattern;
   3. every time after PLACEing the course, remove the
      slot from the faculty favorite slot list;
   4. repeat the step 1,2 until the course slot remain
      unchanged;
}
```

Algorithm 1. Priority regulated DRG (PDRG)

B. Chaos eradication DRG (CDRG)

In this decision tree, the less demanded time slots (with respect to number of rooms) are labeled as "cold" whereas high demanded ones are labeled as "hot". Among the remaining most conflicted courses (may not or partially placed) with low frequent time slots of the corresponding teacher are chosen for the second decision tree. If the considerable slot is empty then the course is placed in that slot, otherwise the colors have to be matched. If the color of the courses placed in the slot matches with the concerning course, the latter course is classified, and if not other options are taken into consideration for the teacher. The considerable courses in CDRG are the overlooked courses from PDRG, where PDRG confirms the day-time pattern is not possible for the courses due to color and the scattered choice of slots by the teachers. So day-time slots pattern are ignored in this operation. After using this second decision tree few courses may remain unchanged. Nevertheless this time the number of remaining courses is much less than the previous one. Around 32% of the remaining courses are placed successfully by CDRG. The combined effort of decision trees still provides high confidence. Fig. 8. demonstrates the Chaos eradication DRG. The pseudo

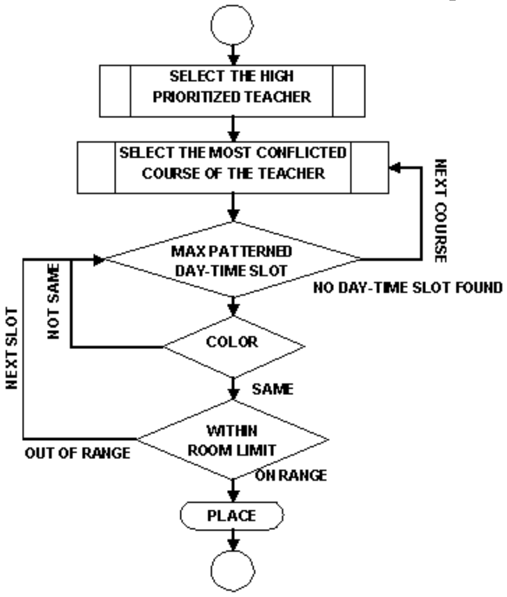

Figure 7. Decision Tree of PDRG

code presented in algorithm 2 describes actions to be taken by Chaos eradication DRG (CDRG). Now this tree holds time complexity of O((n – d) × (m – l)) where the maximum number of courses per teacher is 'n', the number of faculties is 'm' and 'd' is the already classified courses of each teacher by the PDRG and 'l' is the number teacher whose all courses were placed in the routine by PDRG.

CDRG( )

{

// select the most conflicted courses not yet placed or
// partially placed;
// select the remaining low frequent favorite slot of the
// faculty;

1. IF the slot is empty PLACE the course;
2. ELSE find the color of the course that already placed on the slot;
   2.1. IF the color is same AND on the range of the room AND the course is not already placed on that day before, PLACE the course;
   2.2. ELSE select the next low frequent favorite slot of the faculty;
3. Repeat the step 1,2 until the course remain unchanged state;

}

Algorithm 2. Chaos eradication DRG (CDRG)

C. *Tolerable DRG (TDRG)*

The first two decision trees are aimed at automated generations of a better assignment. The second approach seeks to find an assignment of vector which may be more difficult to locate in the search space using the already assigned vector. The third decision tree allows the remaining courses according to the priority of the teachers to find a place into the routine within a tolerable conflict range of the subsequent teachers' favorite slots. Here the course color is overlooked. This classification now introduces errors into the system by considering the tolerable students conflicts only. Important issue is that, this manipulation may iterate several times to include as many courses possible to place in to the routine. 22% of the unclassified and partially classified courses are labeled with a tolerable error.

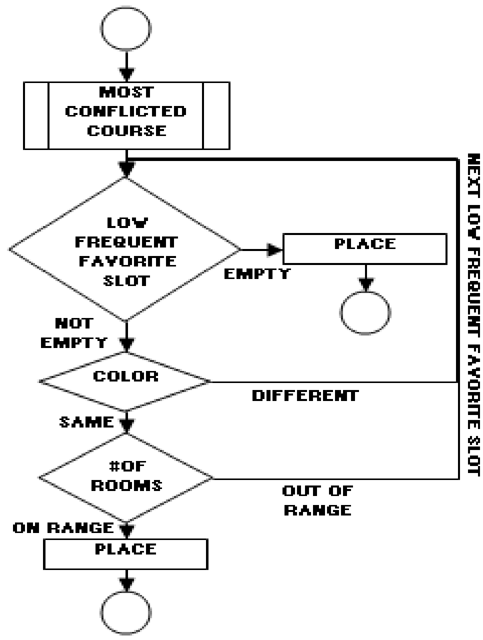

Figure 8. Decision Tree of CDRG

The flowchart presentation of this decision tree is shown in Fig. 9. The pseudo code presented in algorithm 3 describes actions to be taken by Tolerable DRG (TDRG). The average time complexity of TDRG is approximately O (t × (n – (d + o)) × (m – (l + p)) where the maximum number of courses per teacher is 'n', the number of faculties is 'm' and 'd' is the already classified courses of each teacher by the PDRG and 'l' is the number teacher whose all courses were placed in the routine by PDRG. 'o' is the number of courses placed by CDRG and 'p' is the number of teachers whose courses are completely placed by CDRG. Here 't' is the number of TDRG iterates.

TDRG( )

{

// select the most conflicted courses not yet placed or

// partially placed of a high prioritized teacher;
// select the remaining low frequent favorite slot of the
// faculty;

1. IF the slot is empty PLACE the course;
2. ELSE IF on the range of the room AND conflict among the courses are in tolerable range AND the course is not already placed on that day before, PLACE the course;
3. ELSE select the next low frequent favorite slot of the faculty;
4. Repeat the step 2,3 until the course remain unchanged state;

}

Algorithm 3. Tolerable DRG (TDRG)

D. Neighboring Tour DRG (NTDRG)

The fourth possibility search allows the courses to look over the contour of the teachers' favorite slots to find the least conflicted slots. Although the students' scattered choices is overlooked in TDRG but the placed courses do not displease teachers' preference. The final decision tree is modeled in such a manner so that the rest of unplaced courses are going to be graded into the routine vector within a minimum distance of the teachers' choice. Manhattan Distance *(MD)* is calculated for this placement of courses. *MD* = total slot gap of t_k on each day i.e. the total unused slots of a teacher on a particular day. Manhattan Distance is a vital performance measuring tool to find the slot gaps per day for an individual teacher. The optimization can be done by keeping Cumulative Manhattan Distance *(cMD)* for the teachers as low as possible where $cMD = \sum_{\text{all used day by } t} MD(t)$ It is assured that the courses were not yet placed on that day earlier. From the remaining courses, a course is selected according to the priority of the teacher.

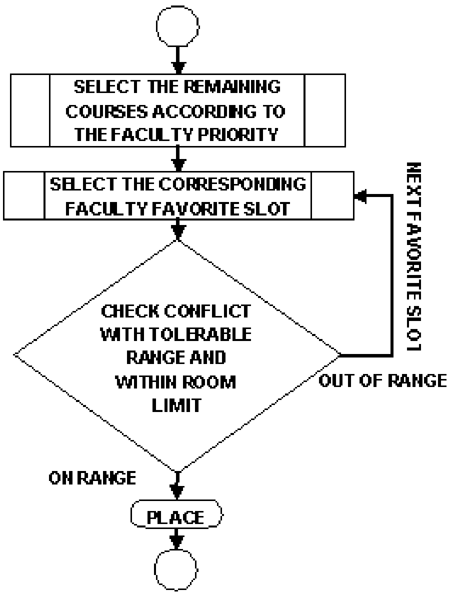

Figure 9. Decision Tree of TDRG

Complement of the intersection between the classification value of that course and the corresponding teachers' used slots are considered as new host slots. The neighboring slots of the new hosts are the most likely candidates. Among the candidates the most "cold" (less desired slots) slots are considered as candidates.

Considerable issues in this placement are tolerable conflict range, allocable number of rooms and day-time misjudgment. Fig. 10. demonstrates the overall scenario of Neighboring Tour DRG (NTDRG). The approximate time complexity is O(2 × (n – (d + o + u)) × (m – (l + p + v)) ≈ O((n – (d + o + u)) × (m – (l + p + v)) where the maximum number of courses per teacher is 'n', the number of faculties is 'm' and 'd' is the already classified courses of each teacher by the PDRG and 'l' is the number of teacher whose all courses were placed in the routine by PDRG. 'o' is the number of courses placed by CDRG and 'p' is the number of teachers whose courses are completely placed by CDRG. 'u' is the number of courses placed by TDRG and 'v' is the number of teachers whose courses are completely placed by TDRG. The pseudo code presented in algorithm 4 describes actions to be taken by Neighboring Tour DRG (NTDRG).

NTDRG( )
{
// select the courses not yet placed or partially placed;
// select the corresponding faculty routine;

1. Find the candidate slot (where candidate slot is the neighboring slots of the used slots of the faculty);
2. IF on the range of the room AND conflict among the courses are in tolerable range AND the course is not already placed on that day before, PLACE the course;
3. ELSE select the next candidate slot of the faculty;
4. Repeat the step 2,3 until the course remain unchanged state;

}

Algorithm 4. Neighboring Tour DRG (NTDRG).

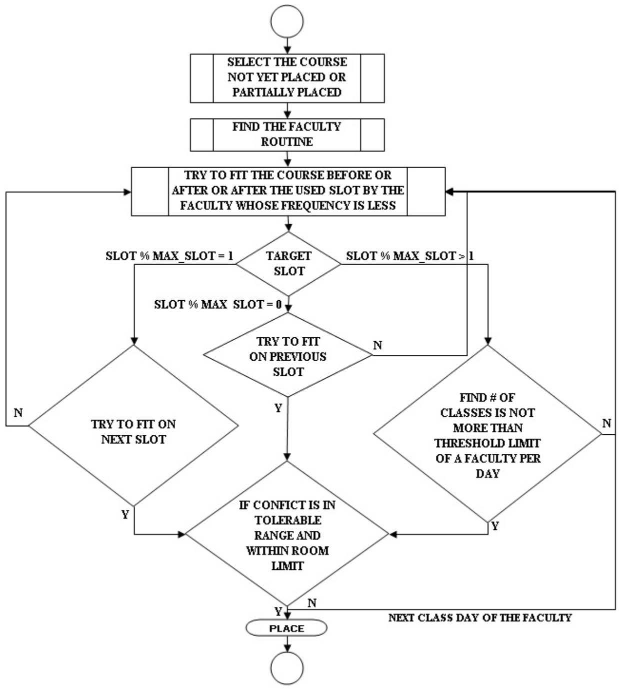

Figure 10. Decision Tree of NTDRG

These four sequential decision trees feed forward to an acceptable solution. It is ensured that the course is not yet place on that very day whenever the "PLACE" decision is taken. The Overall Complexity is shown in equation (1).

= PDRG + CDRG + TDRG + NTDRG
= O(mn) + O((n-d)(m-l)) + O(t(n-(d+o))(m-(l+p))) + O((n-(d+o+u))(m-(l+p+v)))
= mn + mn – nl – md + dl + t(mn – nl – np – md + dl + dp – mo + ol + op) + mn – nl – np – nv – md + dl + dp + dv – mo + ol + op + ov – mu + ul + up + uv
= (t + 3)mn - (d + td + to + d + o + u)m – (l + tl + l + tp + v)n + (dl + tdl + tdp + tol + top + dl + dp + dv + ol + op + ov + ul + up + uv)
= A.mn – B.m – C.n + K -------------------- (1)

where,

A = t + 3;
B = (t + 2)d + (t + l)o + u;
C = (t + 2)l + tp + v;
and K = dl + tdl + tdp + tol + top + dl + dp + dv + ol + op + ov + ul + up + uv

The cumulative time complexity of DRG is O(A.mn - B.m - C.n), The cumulative time complexity of DRG mainly depend upon the number of iterations in TDRG algorithm and number of courses placed by each and every decision tree. The algorithm degrades due to the fact that the students have a freedom to choose any course (assuming that the prerequisite course is completed).

Firstly PDRG faces problem if same prioritized teachers focus into same favorite time slots. By using CDRG this situation may prevail over considering a level of discontinuity into the day-time pattern for the courses. In second run, if the low "chaos" course holds high prioritized teacher then the classification may dissatisfy the teacher. For the third rotation the conflict may arise for the students for not considering the color. For NTDRG, if the host slot is elected as the first ($V\{\{c_i\}_q\}$ $\forall$ i, $c_i \in C$ and $1 \le i \le n$ and $1 \le q \le 30$ where q = 6, 11, 16, 21, 26 is the first slots of the day) and last ($V\{\{c_i\}_q\}$ $\forall$ i, $c_i \in C$ and $1 \le i \le n$ and $1 \le q \le 30$ where q = 5, 10, 15, 20, 25 is the last slots of the day) slot of the day, the previous and the next consecutive slots are from different days. So, this day jump increases huge distance for the teacher which may lead to an unfeasible classification. After NTDRG a few courses may remain partially classified or unclassified due to three major factors (1) Number of rooms not adequate, (2) Teacher's preferred time slot is not applicable and (3) Student conflict may cross tolerable conflict range.

The resultant unclassified or partially classified courses, after all decision trees exploration, represent the problem can not be solved without the extensive relaxation of the constraints mentioned earlier. Un-classification or partially classification may tag to a course not only for the conflict but also for room availability. So, such event may occur where there is no conflict among the courses but the low prioritize course (due to teacher's priority in PDRG or low conflicted score in CDRG or little bit higher considerable conflict score in TDRG) is not able to be placed in to the routine vector for room limitation. Same situation may arise for other two constraints. So, to eliminate the partially classified or unclassified courses the above mentioned factors have to be compromised that is by increasing the room capacity or expanding teacher's favorite time or ignoring students' conflicts.

V. Exam-time Tabling Algorithm (ETA)

Typical constraint programming method is applied to the above exam-time tabling problem. By considering the equal or less number of members only from the power set of the groups of courses, the ETA calculates the total number of conflicts and the total number of students on each group. The proposed algorithm also tries to place the course groups on a specific exam slot. The total numbers of valid power set are the combination of the groups along with the given slots for the exam. Valid range in calculating the number of members is 1 to equal to given slot value 's' i.e. ${}^{Gn}C_s + {}^{Gn}C_{s-1} + \ldots + {}^{Gn}C_1$ where 'Gn' is the total number of course groups. By sorting the groups in descending order according to their total student to conflict ratio, the ETA tries to judge the most appropriate groups to place in to the exam-time table first. Here if the conflict signifies zero then the total student to conflict ratio remains equal to the total number of the students. Among the sorted power set list the most supported single member is selected to be placed on each day of the exam-time table by using greedy algorithm i.e. by choosing the most profitable groups first. Support vector is calculated from the ratio of the total student and conflict. If the provided day for exam scheduling is less and the groups of courses and number of slots is more than one on each day, then the placed group will try to calculate the most appropriate unplaced groups and select the group to form the pair. This situation may produce concurrent conflicts but conflicts among the courses on each slot remain zero. These newly paired groups are eliminated from the list so that other placed groups can select their feasible member groups. Considering the scenario described in Fig. 4, the resultant colorized course groups are G1 = $\{c_1, c_4\}^{80}$ and G2 = $\{c_2, c_3\}^{85}$ and the power set of the groups are {G1}, {G2}, {G1, G2}. Here the highest considerable set are those, whose number of members are less or equal to the provided slots for exam per day. Total student to conflict ratio of the selective power sets are 80, 85 and 165/19, where the ETA mainly concentrates on. If the provided days for the exam are equal to the number of groups then the most appropriate exam schedule is Day 1 = G1 and Day 2 = G2. There is an 11.5% possibility of having concurrent exam, if the provided days for exam are less then the number of colorized course groups. The pseudo code presented in algorithm 5 describes actions to be taken by Exam-Time Tabling Algorithm (ETA)**.**

ETA( )
{
1. Color the courses and from the group;

2. Find the power set of the groups according to the provided exam slots per day;
3. Sort the power set in descending order according to their total student to conflict ratio treated as "Gain";
4. PLACE the highest "Gain"ed course on each day of exam;
5. Find the most appropriate pair for the group;
6. Optimize the exam routine by imposing the MST (Minimum Spanning Tree – by using Prims algorithm; Here the graph of the groups contains edges with conflicts) to spread away the concurrent exams for the student.

}

Algorithm 5. Exam-Time Tabling Algorithm (ETA).

For better exam-time table ETA also calculates the conflicts among the each day-placed group and considers these groups as a vertex. Further optimization can be done by using PRIMS algorithm. In this case PRIMS algorithm will increase the Manhattan distance *(MD)* of the closest conflicting groups where the non conflicting groups also hold an edge with zero weight. This optimization stage is used to maximize the day gap of exams for the students. So by using PRIMS algorithm, ETA founds the minimum spanning tree of group of courses where the nearest group exams hold less common students. Manhattan Distance *(MD)* represents the value of total student to conflict ratio between the groups of the days. The cumulative time complexity of ETA is $\approx O(s^{Gn} + Gn^2)$ depends upon the provided slots per day (s) strictly.

VI. Experimental Results

A. Algorithm Analysis

The Decision tree based Routine Generation algorithm (DRG) is a set of 4 sequential feed-forwarded trees. Important observation has been made by considering the fitness value where it represents the best matching score among these selected day-time pattern slots for a selected teacher. An example is shown in Fig. 1, 2, 3, 4 and 5(a) where the first decision tree (PDRG) selects t_2 because of high priority and t_2 (11) holds the courses $\{c_2\}$. The day-time pattern shows {{2,12,22},{29}} = {2,12}, {12,22} or {2,22} as the best match for the course c_2 from t_2 favorite time slots. Here after imposing the day-time pattern with respect to number of class per week for the course, the calculated ordered sets of fitness value are {2,12}, {12,22}, {2,22} and {29} where the first three sets hold highest and equal fitness value corresponding with class per week. As no courses have been placed yet vector V is empty i.e.; $V(\{c_i\}_2) = \{\ \}$ & $V(\{c_i\}_{12}) = \{\ \}$ & $V(\{c_i\}_{22}) = \{\ \}$ & within room limit hence, the fitness value({2,12}) = fitness value({12,22}) = fitness value({2,22}), where the fitness value is an integer number representing the maximum possibility to take the classes on the provided time slots. So, PDRG places c_2 in slot {2,12}, i.e. $V(\{c_2\}_2)$ and $V(\{c_2\}_{12})$ where {2,12} is a random selection.

Again PDRG selects t_4 with the priority 10 as its next candidate which holds the courses $\{c_5\}$. Here the day-time pattern is, {{2,12,22},{5,15,25}..} = {2,12,22},{5,15,25} extracted from the t_4's favorite slots. As $V(\{c_i\}_2) = \{c_2\}$ & $V(\{c_i\}_{12}) = \{c_2\}$ $V(\{c_i\}_{22}) = \{\ \}$ & $V(\{c_i\}_5) = \{\ \}$ & $V(\{c_i\}_{15}) = \{\ \}$ $V(\{c_i\}_{25}) = \{\ \}$ and the fitness value({2,12,22}) < fitness value({5,15,25}). Therefore c_5 is placed in slot {5,15,25}, i.e. $V(\{c_5\}_5)$ and $V(\{c_5\}_{15})$ and $V(\{c_5\}_{25})$ where $V(\{c_i\}_2)$ and $V(\{c_i\}_{12})$ is not empty. In similar approach the for the teacher t_3 the course c_3 is placed $V(\{c_3\}_{10})$ and $V(\{c_3\}_{20})$ where $V(\{c_3\}_5)$, $V(\{c_3\}_{15})$ and $V(\{c_3\}_{25})$ is ignored due to different course color. And for t_1 the courses $\{c_1, c_4\}$ hold 12 and 8 as total conflicts respectively. Selected course c_1 can be placed {7,17}, {8,18}, {9,19}, {13,23}, {22}, {24} accordingly to the t_1 favorite slots. Course c_1 is placed in $V(\{c_1\}_7)$ and $V(\{c_1\}_{17})$. And from the remaining time slots none the generated pattern provide sufficient classes for the course c_4 where the required number of classes is 3. So the course c_4 is placed on $V(\{c_4\}_8)$ and $V(\{c_4\}_{18})$ and c_4 is tagged as partially placed course needed to be explored more. Random selection among courses for exploration is acceptable if more than one course holds same conflict score.

This class based reasoning left 1 partially placed course, need to be walked around more. Next decision tree CDRG is now in operation with fewer courses as compared with the beginning and slot 24 will be allocated for the course c_4. The important issue is, although the classes are placed in zigzag fashion but all the selected slots for the course c_4 are from the favorite slots of the faculty respectively. So, the denoted term *teacher satisfaction* is 100% for the case and overlapping classes for the student is zero (i.e., *student satisfaction*). The results of above example are shown in Fig. 11. In practice the situation may be more complex with many courses.

By using the same data filtering technique for Exam-Time tabling the proposed algorithm ETA generates the power set of courses according to their course color extracted from the *cross_table*(Cr) where the teacher redundancy is ignored Fig. 5(b). So, the resultant set is, $\{c_1\}$, $\{c_2\}$, $\{c_3\}$, $\{c_4\}$, $\{c_5\}$, $\{c_1, c_4\}$ and $\{c_2, c_3\}$. If the provided exam days are 3 and exam slots per day are 2 the for 100% conflict free exam per slots are day 1 : $\{c_1, c_4\}$, day 2 : $\{c_2, c_3\}$ and day 3 : $\{c_5\}$.

	PDRG	CDRG	TDRG	NTDRG	Final Finding
Unclassified	0	0	-	-	0
Partially classified	1	0	-	-	0
Day-time patterned classified	4	1	-	-	5
Student conflict	0	0	-	-	0
Unsatisfied time	0	1	-	-	1

Figure 11. Simulation result of DRG

Further more if the provided days for exam are 2 and slots are 2 then the exam-time tabling is like, day 1 : slot 1 $\{c_1, c_4\}$, slot 2 $\{c_5\}$ and day 2 : slot 1 $\{c_2\}$, slot 2 $\{c_3\}$, where day 1 consists zero conflict of exam on each slots but 1 consecutive exam of an student. Fig. 12 shows the overall outcome of the ETA algorithm for this special scenario.

B. Experimental Results

Test results for DRG algorithm are carried out on a PC with Pentium IV/1.6 GHz processor and 256 MB of memory. Table I. shows the computational results for semester 1 and semester 2 with 66 and 61 courses correspondingly. Where for semester 1, around 24 teachers, with minimum 10 classes per week and at least 3 courses and 120 students with 430 combinations of choices of courses are considered as input. For semester 2, around 23 teachers, with minimum 10 classes per week and at least 3 courses and 106 students with 414 combinations of choices of courses are considered as input.

Day		Slot 1 Cr. (total std)	Slot 2 Cr. (total std)	Slot 3 Cr. (total std)	Con-current Exam Conflict	Overall Gain
Day = 3 Slot = 2	1	c_1 (40)	c_4 (40)	-	0	80
	2	c_2 (50)	c_3 (35)	-	0	85
	3	c_5 (45)	-	-	0	45
Day = 2 Slot = 2	1	c_2 (50) c_3 (35)	c_5 (45)	-	1	130
	2	c_1 (40)	c_4 (40)	-	0	80
Day = 1 Slot = 2	1	-	-	-	-	Not Possible
Day = 1 Slot = 3	1	c_1 (40) c_4 (40)	c_5 (45)	c_2 (50) c_3 (35)	21	10

Figure 12. Simulation result of ETA

In Table I. and Fig. 11 *unclassified courses* refers to the number of courses that were not classified by any decision trees, *partially classified courses* gives the number of courses partially classified (the number classes already placed into the routine is less then the required classes). *Day-time pattern* shows the numbers of courses that followed the day-time pattern. s*tudent conflict* estimates the percentage of unsatisfied requirements for courses by students. *Unsatisfied time* refers the number of time slots automatically generated beyond teachers' favorite choice. *Final Finding* refers the final output for every subsection after using the all cascade trees. The main objective of this classification is to achieve the state where the value the *unclassified course* is equal to zero. Final value of *unsatisfied time* and s*tudent conflict* shows the teacher and student satisfaction respectively. Randomization or prediction on classification is not used in DRG. So the output of this ≈ O(A.mn - B.m - C.n) based deterministic algorithm strictly depends upon the input.

Test results for ETA are generated depending upon 61 courses with average 25 students per course within 9 exam days and 2 slots per day for semester 2. Table II. as well as Fig. 12 describes the outcome of the Exam-time Tabling Algorithm. Here the "*slots*" represents the total numbers of consecutive exam on a single day. Time duration is 3 hours for exam-time tabling whereas same slot holds 1 hour as class duration in DRG. Total student to conflict ratio on a particular exam day is referred as *overall gain*. Around 3.8% of the total students hold concurrent exam schedule whereas scheduled courses on each slot are 100% conflict free. The *overall gain* also confirms the profit for taking the course groups together as a candidate on a single day. High satisfaction of the students attests a high-quality exam-time tabling.

VII. Conclusion

Timetabling problem usually varies significantly from institution to institution in terms of specific requirements and constraints [22]. Many current successful university timetabling systems are often applied only in the institutions where they were designed. The meta-heuristic, heuristic and hybrid methods are used to solving timetabling problems so as case base reasoning. The main idea is to try and design an algorithm that will choose the right decision tree to carry out a certain task in a certain situation.

This paper outlines the algorithm Decision Tree based Routine Generation (DRG) using OLAP representation, to construct a university class routine and conflict free Exam-Time Tabling algorithm (ETA) to produce conflict free exam schedule with a fixed interval of days. It should be noted that the DRG algorithm brings the complexity to a considerable level and this solution classifies 96% - 97% of the courses as well 93% - 95% satisfaction for teacher. For this data set, students' satisfaction is 100% but in general 90% - 93% satisfaction may be achievable by using DRG. Preferential requirements (*teacher satisfaction*) on time variables are met around 93%. Again the ETA algorithm provides satisfactory exam-time table with 100% satisfaction. The results also illustrate that the proposed algorithms achieve significant performance gains over different data set.

The proposed algorithms are designed in such a manner so that they are easy to code and imply significant importance to construct generalized automated time-tabling software. Author(s) of the paper realize the difference in constraints level in different institutes; however in future generalized automated time-tabling software will be examined. This paper does not consider any classical benchmark problems. It is important to analyze the performance with other established algorithms. Incorporating those heterogeneous constraints with the proposed data structure will also be examined in future.

Acknowledgement

The proposed algorithm is employed to construct a class and exam routine for a reputed university in Bangladesh. Author(s) of the paper express gratitude to the university authority for providing indispensable information.

TABLE I.
COMPUTATIONAL RESULTS OF DRG

		PDRG	CDRG	TDRG	NTDRG	Final Finding
Semester-1	Unclassified courses	5	3	1	0	0
	Partially classified courses	45	32	2	2	2
	Day-time patterned classified	16	31	63	64	64
	Student conflict	0	0	0	0	0
	Unsatisfied time	0	0	3	7	9
Semester-2	Unclassified courses	2	2	1	0	0
	Partially classified courses	41	26	2	2	2
	Day-time patterned classified	18	33	58	59	59
	Student conflict	0	0	0	0	0
	Unsatisfied time	0	0	5	8	11

TABLE II.
COMPUTATIONAL RESULTS OF ETA

		# of Courses on Slot 1	# of Courses on Slot 2	Concurrent Exam Conflict	Total Student on Slot 1	Total Student on Slot 2	Overall Gain
Semester-2	Day 1	3	4	3	72	63	45
	Day 2	3	3	2	77	63	70
	Day 3	2	2	5	32	53	17
	Day 4	4	4	5	76	87	32.6
	Day 5	5	5	11	91	64	14
	Day 6	3	2	3	46	18	21.3
	Day 7	4	4	7	42	74	16.5
	Day 8	2	5	2	28	71	49.5
	Day 9	2	3	2	16	68	42

REFERENCES

[1] Christelle Guért. Narendra Jussien, Patrice Boizumault and Christain Prins, "Building university timetable using constraint logic programming," Springer-Verlag LNCS 1153, pp. 130-145, 1996.

[2] Hans-Joachim Goltz, Georg Küchler and Dirk Matzkae, "Constraint-based timetabling for universities," INAP'98, pp. 75-80, 1998.

[3] Slim Abdennadher and Michael Marte, "University course timetabling using constraint handling rules," Journal of Applied Artificial Intelligence, vol. 14, no. 4, pp. 311-326, 2000.

[4] Thom Früwirth, "Constraints handling rules," Constraint Programming: Basic and Trandes, LNCS 910, Springer, 1995.

[5] E.K. Burke, B.L. MacCarthy, S. Petrovic and R. Qu, "Knowledge Discovery in a Hyper-Heuristic for Course Timetabling Using Case-Based Reasoning," PATAT 2002, 4th International Conference, pp. 90-103, August 2002.

[6] D. Abramson, "Constructing school timetables using simulated annealing: Sequential and parallel algorithms," Management Science, vol. 37, pp. 98-113, 1991.

[7] A. Schaert, "Tabu search techniques for large high-school timetabling problems," 13th National Conference on Artificial Intelligence AAAI'96, pp. 363-368, 1996.

[8] Luca Di Gaspero and Andra Schaerf, "Multi-Neighbour Local Search for Course Timetabling," PATAT 2002, 4th International Conference, pp. 128-132, August 2002.

[9] P. Boizumault, Y. Delon, and L. Peridy, "Constraint logic programming for examination timetabling," Journal of Logic programming, vol. 26, pp. 217-233, 1996.

[10] K. Dowslan, "Using simulated annealing for efficient allocation of students to practical classes," Applied Simulated Annealing – Lecture Notes in Economics and Mathematical System, Springer-Verlag, vol. 396, pp. 125-150, 1993.

[11] K. Dowslan, "Off-the peg or made-to-measure? Timebaling and scheduling with sa and ts," PATAT'97, Springer – Verlag, pp. 37-52, 1998.

[12] G.M. White and J. Zhang, "Generating complete university timetables by combining tabu search with constraint logic," PATAT'97, Springer – Verlag, pp. 187-198, 1998.

[13] E.K. Burke, J. Newall, and R.F. Weare, "Initialization strategies and diversity in evolutionary timetabling," Evolutionary Computation Journal, vol. 6.1, pp. 81-103, 1998.

[14] E.K. Burke, J. Newall, and R.F. Weare, "A simple heuristically guided search for the timetable problem," Proceeding of the International ICSC

Symposium on Engineering of Intelligent System, pp. 574-579, 1998.
[15] E.K. Burke, J. Newall, and R.F. Weare, "A memetic algorithm for university exam timetabling," Lecture Notes in Computer Science, Springer-Verlag, vol. 1153, pp. 241-250, 1996.
[16] Tan, Steinbach and Kumar, Introduction to Data Mining, pp. 130-139, 2004.
[17] Liam T.G. Merlot, Natashia Boland, Barry D. Hughes, and Peter J. Stuckey, "A Hybrid Algorithm for the Examination Timetabling Problem", Proceedings of the 4th International Conference on the Practice and Theory of Automated Timetabling - PATAT'2002, Springer – Verlag, pp. 348-371.
[18] M. Caramia, P. Dell'Olmo, and G.F. Italiano, "New algorithms for examination timetabling", Proceedings: Algorithm Engineering 4th International Workshop, WAE 2000, Germany, September 2000. "Lecture Notes in Computer Science 1982", Springer-Verlag, Berlin Heidelberg New York, pp. 230-241, 2001.
[19] M. Carter, G. Laporte, and J. Chinneck, "A general examination scheduling system", Interfaces, pp. 109-120, 1994.
[20] M. Carter, G. Laporte, and S.T. Lee, "Examination timetabling: algorithmic strategies & applications", Journal of the Operational Research Society, pp. 373 - 383, 1996.
[21] Bart Selman, Henry A. Kautz, and Bram Cohen, "Noise strategies for improving Local search", 12th National Conference on Artificial Intelligence (AAAI-94), pp. 337– 343, 1994.
[22] T. B. Cooper, J. H. Kingston, "The Complexity of Timetable Construction Problems", Practice and Theory of Automated Timetabling, Springer Verlag, pp. 283-295, 1996.

Ashiqur Md. Rahman received his B.Sc. Degree in Computer Science and Engineering from American International University Bangladesh, Dhaka in January, 2004. He is currently perusing his M.Sc. degree from North South University, Dhaka since January 2006. He has authored in 4 national and international journal and conference papers in the area of Data Mining, VHDL, Cryptography and PVc module design. His current research interest is in Grid Computing especially in large Grid Environment.

Shafaat S. Giasuddin received his B.Sc. Degree in Computer Science from Ahsanullah University of Science and technology, Dhaka in November, 2005. He is currently perusing his M.Sc. degree from North South University, Dhaka since January 2006. He has authored in 3 national and international journal and conference papers in the area of Data Mining, VHDL and Cryptography. His current research interest is in Data Mining especially in corporate sector like banking and telecommunication.

Rashedur M. Rahman received his Ph.D. Degree in Computer Science from University of Calgary, Canada in November, 2007. He has received his M.Sc. degree from University of Manitoba, Canada in 2002 and Bachelor degree from Bangladesh University of Engineering and Technology (BUET) in 2000 respectively. He is currently working as an Assistant Professor in North South University, Dhaka, Bangladesh. He has authored more than 25 international journal and conference papers in the area of parallel, distributed, grid computing and knowledge and data engineering. His current research interest is in data mining especially on financial, educational and medical surveillance data, data replication on Grid, and application of fuzzy logic for grid resource and replica selection.