© 2017 IJSRST | Volume 3 | Issue 2 | Print ISSN: 2395-6011 | Online ISSN: 2395-602X

National Conference on Advances in Engineering and Applied Science (NCAEAS)
16th February 2017
In association with
International Journal of Scientific Research in Science and Technology

Smart Time Table

Aafrin Siddiqui, Adeeba Sameen, Ibtesam Ali, Latika Lakkerwar, Rizwana Parveen

Department of computer science & engineering, Anjuman College of Engineering, RTMNU, Nagpur, Maharashtra, India

ABSTRACT

Timetable creation is a very arduous and time consuming task. To create timetable it takes lots of patience and man hours. Time table is created for various purposes like to organize lectures in school and colleges, to create timing charts for train and bus schedule and many more. To create timetable it requires lots of time and man power .In our paper we have tried to reduce these difficulties of generating timetable by Genetics Algorithm. By using Genetic algorithm we are able to reduce the time require to generate time table and generate a Timetable which is more accurate, precise and free of human errors. The first phase contains all the common compulsory classes of the institute, which are scheduled by a central team. The second phase contains the individual departmental classes. Presently this timetable is prepared manually, by manipulating those of earlier years, with the only aim of producing a feasible timetable.

Keywords : Genetic Algorithm, Timetable, Constraints, Chromosomes

I. INTRODUCTION

The Time generation is the most Fundamental activity in any Educational institution. It is also the most difficult and time-consuming process.

The basic aim of our project is to automate the timetable generation process. Our aim is to design a user interactive program that generates the timetable according to the given constraints. The program is designed with special emphasis on the engineering college requirements. The program can simply be extended to suit to the requirements of other kinds of institutions also.

Most colleges have a number of different courses and each course has a number of subjects. Now there are limited faculties, each faculty teaching more than one subjects. So now the time table needed to schedule the faculty at provided time slots in such a way that their timings do not overlap and the time table schedule makes best use of all faculty subject demands. We use a genetic algorithm for this purpose. In our Timetable Generation algorithm we propose to utilize a timetable object. This object comprises of Classroom objects and the timetable for every them likewise a fitness score for the timetable. Fitness score relates to the quantity of crashes the timetable has regarding alternate calendars for different classes.

Classroom object comprises of week objects. Week objects comprise of Days. Also Days comprises of Timeslots. Timeslot has an address in which a subject, student gathering going to the address and educator showing the subject is related also further on discussing the imperatives, we have utilized composite configuration design, which make it well extendable to include or uproot as numerous obligations. In every obligation class the condition as determined in our inquiry is now checked between two timetable objects. On the off chance that condition is fulfilled i.e. there is a crash is available then the score is augmented by one.

II. METHODS AND MATERIAL

The problem under consideration is to automate the process of timetable scheduling in an educational institution subjected to the given constraints. The user will specify the constraints and these constraints will

drive the scheduling of timetable. The user may specil ✓
some of the following constraints. ✓

- The number of departments in his institution.
- The number of staff personnel available in eac department.
- The number of classes in each department.
- The number of subjects to be dealt for each class.
- The minimum number of hours required completing each subject.
- The total number of available hours for each day.
- The number of laboratories available.
- The number of practical sessions per week that are necessary for each class.
- Along with the above-mentioned constraints the user may specify some of the weak constraints such as
- A teacher should not engage two consecutive slots. i.e. he should be provided with an interval of at least one slot between two classes.
- The workload on all teachers should be uniform.
- The practical should be continuous for thre consecutive slots.

The teacher who is assigned with the subject having practical must be engaged with the lab slots during practical session i.e. he should not be engaged with other slots while the practical session is going on.

Background

When you make a class schedule, you must take in consideration many requirements (number of professor students, classes and classrooms, size of classroom, laboratory equipment in classroom, and many others). These requirements can be divided into several groups by their importance. Hard requirements (if you break one of these, then the schedule is infeasible)

- A class can be placed only in a spare classroom.
- No professor or student group can have more than one class at a time.
- A classroom must have enough seats to accommodate all students.
- To place a class in a classroom, the classroom must have laboratory equipment (computers, in our case) if the class requires it.Some soft requirements (can be broken, but the schedule is still feasible)
- Preferred time of class by professors.
- Preferred classroom by professors.

Distribution (in time or space) of classes for student groups or professors. Hard and soft requirements, of course, depend on the situation. In this example, only hard requirements are implemented. Let's start by explaining the objects which makes a class schedule

Objects of Class Schedule

Professor

The Professor class has an ID and the name of the professor. It also contains a list of classes that a professor teaches.

Section

Group the Section Group class has an ID and the name of the student group, as well as the number of students (size of group). It also attends.

Classroom

The Room class has an ID and the name of the classroom, as well as the number of seats and information about equipment (computers). If the classroom has computers, it is expected that there is a computer for each seat. IDs are generated internally and automatically.

Course

The Course class has an ID and the name of the course**.**

Class

CourseClass holds a reference to the course to which the class belongs, a reference to the professor who teaches, and a list of student groups that attend the class. It also stores how many seats (sum of student groups' sizes) are needed in the classroom, if the class requires in the classroom, and the duration of the class (in hours).

Algorithm

The genetic algorithm is fairly simple. For each generation, it performs two basic operations:

Randomly selects N pairs of parents from the current population and produces N new chromosomes by performing a crossover operation on the pair of parents.

Randomly selects N chromosomes from the current population and replaces them with new ones. The algorithm doesn't select chromosomes for replacement if it is among the best chromosomes in the population.

And, these two operations are repeated until the best chromosome reaches a fitness value equal to 1 (meaning that all classes in the schedule meet the requirement). As mentioned before, the genetic algorithm keeps track of the M best chromosomes in the population, and guarantees that they are not going to be replaced while they are among the best chromosomes.

```
// Genetic algorithm
class Algorithm
{

private:


// Population of chromosomes
vector<Schedule*> _chromosomes;

// Inidicateswheahter chromosome belongs to
// best chromosome group
vector<bool> _bestFlags;

// Indices of best chromosomes
vector<int> _bestChromosomes;

// Number of best chromosomes currently saved in
// best chromosome group
int _currentBestSize;

// Number of chromosomes which are replaced in
// each generation by offspring
int _replaceByGeneration;

// Pointer to algorithm observer
ScheduleObserver* _observer;

// Prototype of chromosomes in population
 Schedule* _prototype;

// Current generation
int _currentGeneration;

// State of execution of algorithm
AlgorithmState _state;

// Synchronization of algorithm's state
CCriticalSection _stateSect;

// Pointer to global instance of algorithm
static Algorithm* _instance;

// Synchronization of creation and destruction
// of global instance
staticCCriticalSection _instanceSect;

public:

// Returns reference to global instance of algorithm
static Algorithm&GetInstance();

// Frees memory used by gloval instance
staticvoidFreeInstance();

// Initializes genetic algorithm
Algorithm(intnumberOfChromosomes,
intreplaceByGeneration,
inttrackBest,
 Schedule* prototype,
ScheduleObserver* observer);

// Frees used resources
 ~Algorithm();

// Starts and executes algorithm
void Start();

// Stops execution of algoruthm
void Stop();

// Returns pointer to best chromosomes in population
 Schedule* GetBestChromosome() const;

// Returns current generation
inlineintGetCurrentGeneration() const { return
_currentGeneration; }

// Returns pointe to algorithm's observer
inlineScheduleObserver* GetObserver() const { return
_observer; }

private:

// Tries to add chromosomes in best chromosome group
voidAddToBest(intchromosomeIndex);

// Returns TRUE if chromosome belongs to best
chromosome group
boolIsInBest(intchromosomeIndex);

// Clears best chromosome group
voidClearBest();

};
```

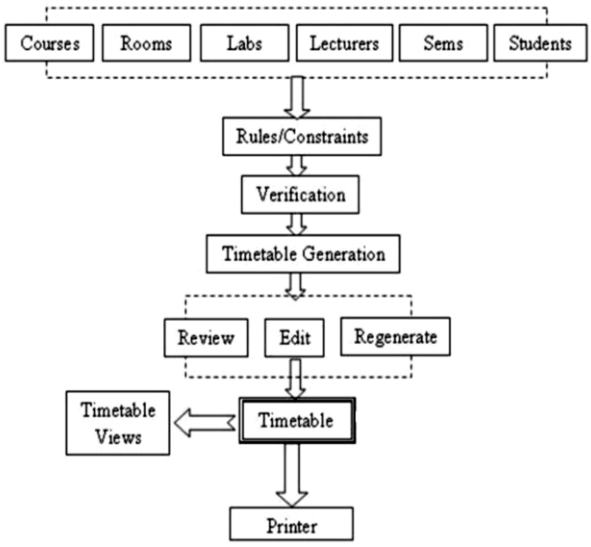

Figure 1: General view of Smart Timetable

Literature Survey

Trying to develop a software which helps to generate Timetable for an Institution automatically. By looking at the existing system we can understand that timetable generation is done manually. Manually adjust the timetable when any of the faculty is absent, and this is the big challenge for Automatic Timetable Generator that managing the timetable automatically when any of the faculty is absent. As we know all institutions) organizations have its own timetable, managing and maintaining these will not be difficult. Considering workload with this scheduling will make it more complex. As mentioned, when timetable generation is being done, it should consider the maximum and minimum workload that is in a college. In those cases timetable generation will become more complex. Also, it is a time consuming process.

III. CONCLUSION

As discussed, an evolutionary algorithm, genetics algorithm for time tabling has been proposed. The intention of the algorithm to generate a time-table schedule automatically is satisfied. The algorithm incorporates a number of techniques, aimed to improve the efficiency of the search operation. By automating this process with the help of computer assistance timetable generator can save a lot of precious time of administrators who are involved in creating and managing various timetables of the institutes.

IV. REFERENCES

[1]. Boehm B, "A Spiral Model of Software Developmentand Enhancement", ACM SIGSOFT Software Engineering Notes, ACM, 11(4):14-24, August 1986

[2]. Boehm B, "A Spiral Model of Software Development and Enhancement", IEEE Computer, IEEE, 21(5):61-72, May 1988

[3]. Boehm, B, "Spiral Development: Experience, Principles,and Refinements", Special Report CMU/SEI-2000-SR-008, July 2000

[4]. D. Abramson. Constructing school timetables using simulated annealing: sequential and parallel algorithms. Manage. Sci., 37(1):98–113, January 1991.

[5]. David Abramson and J Abela. A parallel genetic algorithm for solving the school timetabling problem. In 15 Australian Computer Science Conference, 1992.

[6]. Enrique Alba. Parallel Met heuristics: A New Class of Algorithms. Wiley- Interscience, 2005. www.tutorialspoint.com

[7]. Georgios Varsamopoulos "How to Write a Technical Paper: Structure and Style of the Epitome of your Research'"

[8]. AnujaChowdhary, Priyanka Kakde, ShrutiDhoke, Sonali Ingle, RupalRushiya, Dinesh Gawande "TIMETABLE GENERATION SYSTEM'" A paper published in IJCSMC Vol. 3, Issue. 2, February 2014.

[9]. M.Lalena, "Traveling Salesman Problem using GeneticAlgorithm'" retrieved from www.lalena.com/AI/T/.

[10]. Y. Has an A Bahanrum, O. Maharum, "A Job-Shop Scheduling Problem using Genetic Algorithm'" Proceedings of the Second IMT-GT Regional Conferenceon Mathematics, Statistics and Applications. University Sains Malaysia, Penang June 13-15, 2006.

[11]. J .J. Moreira, "A System for Automatic Construction forExamination Timetable Using Genetic Algorithm'". The Techne Polytechnic Studies Review Journal, Vol.6 No.9 2008.

[12]. V.T. Matthew, "Genetic Algorithm. Department of CivilEngineering'", Indian Institute of Technology, Bombay, Mumbai, 2005.

[13]. P. Ross, D. Corne, "Applications of (GA) Genetic Algorithms'", Department of Artificial Intelligence,University of Edinburgh, 2003. retrieved from www.citeseerx.ist.psu.edu/viewdoc/download?

[14]. Mosaic Space Blog, "The Practice and Theory of Automated Timetabling'" PATAT 2010, Mosaic SpaceBlog, University and college planning and management retrieved, from http://mosaicd.com/blog, 2011, Last accessed date 21st January 2012.

[15]. D. G. Maere, (2010). "How Working Group AutomatedTimetabling was founded'", retrieved from http://www.asap.ac.nott.ac.uk/, 2010, Last accessed date9th December 2011.