Automatic Time-Table –Implementation using Genetic Algorithm

Asif Ansari[#1], Ajit Parab[#2]
[#1] Lecturer, Computer Technology, BGIT
[#1]ansariasif23@gmail.com
[#2] HOD, Computer Technology, BGIT

Abstract: A college timetable is a temporal arrangement of a set of classes and classrooms in which all given constraints are satisfied. Timetabling has long been known to belong to the class of problems called NP hard. This project introduces a practical timetabling algorithm capable of taking care of both Hard and soft constraints effectively, used in an automated timetabling system. The Genetic Algorithm is main component of project which produces the XML based weekly timetable sheet as the output. The project takes various inputs from the user such as Teacher List, Course List, Semester List, Room List, Day List and Timeslot as well as various rules, facts and constraints using Genetic Algorithm, which are stored in XML based knowledge base. This knowledge base serves as input to our Timetable Generator Algorithm. Further benefits of choosing these frameworks are explained in later part of report with practically acceptable results.

Keywords: Genetic Algorithm, XML, hard constraints, soft constraints.

I. INTRODUCTION

Even though most college administrative work has been computerized, the lecture timetable scheduling is still mostly done manually due to its inherent difficulties. The manual lecture-timetable scheduling demands considerable time and efforts. The lecture-timetable scheduling is a constraint satisfaction problem in which we find a solution that satisfies the given set of constraints. The college lecture-timetabling problem asks us to find some time slots and classrooms which satisfy the constraints imposed on offered courses, lecturers, classrooms and so on. Since the problem is a combinatorial optimization problem belonging to NP-hard class [1], the computation time for timetabling tends to grow exponentially as the number of variables increase. There have been a number of approaches made in the past decades to the problem of constructing timetables for colleges and schools. Timetabling problems may be solved by different methods inherited from operations research such as graph coloring and mathematical programming, from local search procedures such as tabu search and simulated annealing, or from backtracking-based constraint satisfaction manipulation In our project, timetabling problem is formulated as a constraint satisfaction problem and we proposed a practical timetabling algorithm which is capable of taking care of both hard and soft constraints.

II. TIMETABLING

A timetable construction is an NP-complete scheduling problem. It is not a standard job-shop problem because of the additional classroom allocation. It is large and highly constrained, but above all the problem differs greatly for different colleges and educational institutions. It is difficult to write a universal program, suitable for all imaginable timetabling problems. Although manual construction of timetables is time-consuming, it is still widespread, because of the lack of appropriate computer programs.

A timetabling algorithm can use different strategies to get a solution without violation of hard constraints. Violations can either be avoided from the outset or penalized to lead the algorithm towards better solutions and introduce repair mechanisms.

The considered college is a polytechnic college which has the following characteristics on its course administration.

1. The classes for students are scheduled in the weekday's that can be specified as 5 to 6 working days
2. The types of lectures are:

- Theory lecture
- Tutorial

3. The class size of theory lectures is from 40 to 60.
4. A minimum timeslot is a 1 Hour interval.
5. For theory classes takes 1 Timeslot and tutorial takes 1 timeslot.

III. PROPOSED SYSTEM

The System proposes an optimized technique to automate time table generation system. Time table generation system involves various challenging constraints of resources including faculties, rooms, time slots etc. The proposed technique filters out the best of active rules and Genetic algorithm to generate the optimized solution. Genetic Algorithm and Active Rules together form a complete sphere for developing a system, which needs to satisfy various constraints.

Active Rules provide "event-condition-action" model for the implementation of any rule based system. Planning timetable is one of the most complex and error prone applications. There are still serious problems like generation of high cost time table are occurring while scheduling and these problems are repeating frequently. Therefore there is a great requirement for an application distributing the course evenly and without collisions. The aim is here to develop a simple, easily understandable, efficient and portable application which could automatically generate good quality time table within a second.

Active rules are described for the knowledge of intelligent agents (i.e. Constraints), GAs are described and their use in optimizing rule based agent is proposed, methods are apply to the problem of optimizing some results of this application are presented and finally, some conclusion and possible direction for future research are presented. The structure of time table generator consist Input Data Module, relation between the input data module, time slots module, applying active rules and GA module then extract the reports.

Input Format

Each instance is in a single file, containing a file header and four sections: courses, rooms, curricula, and constraints. The header provides all scalar values and each section provides the arrays for that specific aspect of the problem. The exact format is shown by the following example

Name	Courses	Rooms	Days	Period/ Day	Curricula	Constraints

Subject Name	Lecturer Name	Hours/Week	Working Days	Strength of class

Class Room No	Capacity

Course Name	Number of Courses	List of Courses(subject)

Figure 1: Input Format

Constraints

Constraints can be divided in to two parts:

1. Hard constraints
2. Soft constraints

Hard constraints

Lectures: All lectures of a course must be scheduled, and they must be assigned to distinct periods.
Room Occupancy: Two lectures cannot take place in the same room at the same time.
Conflicts: Lectures of courses in the same curriculum or taught by the same teacher must be all scheduled in different periods.
Availabilities: If the teacher of the course is not available to teach that course at a given period, then no lectures of the course can be scheduled at that period.

Soft constraints

Room Capacity: For each lecture, the number of students that attend the course must be less or equal than the number of seats of all the rooms that host its lectures.
Minimum Working Days: The lectures of each course must be spread into a minimum number of days.
Room Stability: All lectures of a course should be given in the same room.

System Structure

Figure 2 below shows The System Structure of the timetable Generator.

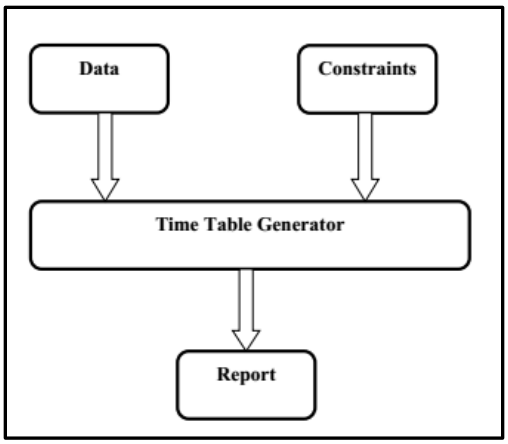

Figure 2: System Structure

Active Rules are based upon an Event-Condition-Action architecture. The meaning of an ECA rule is: "when an event occurs check the condition and if it is true execute the action". There is an event language for defining events and for specifying composite events from a set of primitive ones. The condition part of an ECA rule formulates in which state the database has to be, in order for the action to be executed. The action part of an ECA rule usually starts a new transaction which when executed may trigger new ECA rules. The system will then select the rule with the highest priority to fire, or will arbitrarily select a rule to fire if there is more than one with the same priority.

IV. GENETIC ALGORITHM

Genetic algorithms are methods of solving problems based upon an abstraction of the process of Natural Selection. They attempt to mimic nature by evolving solutions to problems rather than designing them. Genetic algorithms work by analogy with Natural Selection as follows. First, a population pool of chromosomes is maintained. The chromosomes are strings of symbols or numbers. There is good precedence for this since humans are defined in DNA using a four-symbol alphabet. The chromosomes are also called the genotype (the coding of the solution), as opposed to the phenotype (the solution itself). In the Genetic algorithm, a pool of chromosomes is maintained, which are strings. These chromosomes must be evaluated for fitness. Poor solutions are purged and small changes are made to existing solutions and then allow "natural selection" to take its course, evolving the gene pool so that steadily better solutions are discovered. The basic outline of a Genetic Algorithm is as follows: Initialize pool randomly

For each generation
{
Select good solutions to breed new population
Create new solutions from parents
Evaluate new solutions for fitness
Replace old population with new ones
}

The randomly assigned initial pool is presumably pretty poor. However, successive generations improve, for a number of reasons:

1. Selection:

The process is realized in the system using Movable Lecture Selection Filter. This module decide Decides on keeping or discarding a selection (which is a Planning Entity, a planning Value, a Move or a Selector). If works on Boolean value. If the selection is accepted it returns TRUE and if the selections is rejected it returns FALSE. This will never return NULL value. Proposed System.

2. . Mutation.

Mutation is a divergence operation. It is intended to occasionally break one or more members of a population out of a local minimum/maximum space and potentially discover a better minimum/maximum space. The mutation of the chromosome is realized by the call various functions such as Lecturer Difficulty Weight, Period Strength, and Room Strength. It can be realized in the following possibilities. The first two cases are random changes of the chromosome: Exchange of randomly chosen elements of Room and Time the two randomly chosen lectures are exchanged in time periods, and the another two are exchanged in their rooms. The randomly chosen lecture is substituted by the randomly chosen time period, and to another lecture is randomly chosen room.

3. Crossover:

The process is realized in the system by using courseconflict.java module if conflict arises in any of the resources, then the resources are interchanged so as to avoid the conflict. That is either of the resources is moved to the right or to the left to avoid the conflict. The new descendant is started by crossing of two parents for every case. The descendant receives

random part of field Room and Time for the first parent, the rest it receives from the second parent. The cross point is various for the two fields.

Algorithm
// Initialise generation 0:
k := 0;
Pk := a population of n randomly-generated individuals;
// Evaluate Pk:
Compute fitness(i) for each i∈ Pk;
do
{ // Create generation k + 1:
// 1. Copy:
Select (1 −) × n members of Pk and insert into Pk+1;
// 2. Crossover:
Select × n members of Pk; pair them up; produce offspring; insert the offspring into Pk+1;
// 3. Mutate:
Select μ × n members of Pk+1; invert a randomly-selected bit in each;
// Evaluate Pk+1:
Compute fitness(i) for each i∈ Pk;
// Increment:
k := k + 1;
}
while fitness of fittest individual in Pk is not high enough; return the fittest individual from Pk;

V. CONCLUSION

The GA in timetabling framework has been shown to be successful on several real problems. It has been shown that the genetic algorithm perform better in finding areas of interest even in a complex, real-world scene. One could argue that the genetic algorithm can find a local optimum and then stop. This is always a danger with a genetic algorithm, but again it depends on the search space. In this time table generation approach, there are many good solutions and the genetic algorithm will find one of them. In extreme cases where there is only one good solution the genetic algorithm may fail, but again it can be restarted by the Active Rules with many chances to find a better solution. One could also argue that this architecture is not powerful enough since it does not work based on an event/action language. However there is nothing to prevent this architecture from being a subset of a rich and powerful event/action language. In such a case it can be used to pick the rule to be fired when there are no other criteria available for rule selection. In other cases it may be better to let the genetic algorithm pick the rule to be fired, instead of having many conditions which will complicate the active rule set and consequently increase design, test and maintenance times. The benefits of this approach are simplified design and reduced development and maintenance times of rule-based agents in the face of dynamically evolving environments.

REFERENCES

[1] Michael R. Garey and David S. Johnson. Computers and intractability: a guide to the theory of np-completenesspage no 13-14. 1979.

[2] S. Petrovic S. and E. Burke. University timetabling. In Handbook of scheduling: Algorithms, models, and performance analysis, pages 1-23. CRC Press, Boca Raton, FL, 2004.

[3] H. L. Fang. Genetic algorithms in timetabling scheduling. Ph.D. thesis, University of Edinburgh, Edinburgh, UK, 1994.

[4] S. A. MirHassani. A computational approach to enhancing course timetabling with integer programming. Applied Mathematics and Computation, 175: 814-822, 2006.

[5] M. L. Pinedo. Planning and scheduling in manufacturing and services. In Springer Series in operations research, pages 3-8. Springer, New York, 2005.

[6] J. Clausen. Branch and bound algorithms- principles and examples, 1999. http://citeseerx.ist.psu.edu/.

[7] M. Friedrich, I. Hofsäß and S. Wekeck. Timetable-Based Transit Assignment Using Branch & Bound, 2009. http://cgi.ptv.de/download/traffic/library/2001%20TRB%20 Timetable%20Transit%20Assignment.pdf.

[8] T. E. Morton and D. W. Pentico. Heuristic scheduling systems. Wiley Series in Engineering & Technology Management, New York, 1993.

[9] R. Montemanni. Timetabling: Guided Simulated Annealing + Local Searches, 2003. http://www.idsia.ch/Files/ttcomp2002/montemanni.pdf.

[10] D. Abramson, M. Krishnamoorthy and H. Dang. Simulated Annealing Cooling Schedules for the School Timetabling Problem, 1997. http://citeseerx.ist.psu.edu /viewdoc/summary?doi=10.1.1.35.994.

[11] F. Glover and M. Laguna. Tabu Search, 2009.http://www.dei.unipd.it/~fisch/ricop/tabu_search_glover_laguna.pdf.

[12] A. Hertz, E. Taillard E. and D. A. de Werra. Tutorial OnTabu Search, 2009. http://www.cs.colostate.edu/~whitley/ CS640/hertz92tutorial.pdf

[13] A. Schaerf. Tabu Search Techniques for Large High-School Timetabling Problems, 1996. http://citeseerx.ist.psu.edu/ viewdoc/summary?doi=10.1.1.20.9007.

[14] W. Legierski, Constraint-Based Reasoning for Timetabling. AI-METH 2002-Artificial Intelligence Methods (November 13-15, 2002), 2002. http://www.aiforum. org/data/22-cons.pdf

