TIME TABLE SCHEDULING USING "GENETIC ALGORITHM"

Chetan Kale[1], Sarfaraj Hodekar[2], Avinash Pawar[3]
chetank400@gmail.com,cooldudester7276@gmail.com,sarfarazsam@rediffmail.com
[1,2,3,] *Students, Department of Computer Engineering,*
RMCET, Ambav(Devrukh),Mumbai University
[4*] Prof. V. S. More
vikasmore08@gmail.com
[4] *Lecturer,Department of Computer Engineering,*
Rajendra Mane College of Engineering & Technology, Ambav(Devrukh),Mumbai University

ABSTARCT:
The problem of time table scheduling for classes in any academic environment is a major challenge due to the large number of students and courses that students have to offer and the limited number of classrooms which is usually available for such number of courses. How then do we resolve the problem in order to avoid clashes of courses and conflicts of interest as well as reduce stress due to long hours of lecture time and complex examination time table? This project work presents a very efficient method called genetic algorithm for solving such problems of timetabling. This project is specifically dedicated to the application of the algorithm in timetable scheduling. It demonstrates the scope of the algorithm and gives a practical example of how the algorithm works. The rightful application of the algorithm will make problems of time table scheduling a thing of the past. Timetable scheduling using genetic algorithm takes configuration file as input in which all required fields of timetable are mentioned. Genetic algorithm takes this input, process it and display most fit item without conflicts.

Keywords: *Genetic Algorithms (GAs), Timetable Generator, Fitness function, Mutation.*

1 . INTRODUCTION

1.1 Overview

This project explains an example usage of Genetic Algorithms (GAs) for finding optimal solutions to the problem of Timetable scheduling. There are two objectives in this. First, to provide a detailed introduction to the topic of Genetic Algorithms, their method and their variations. The second objective is to apply them to the problem of Automated Timetable scheduling. In this project, configuration file is given to genetic algorithm as an input and genetic algorithm resolves conflict and generate a timetable automatically.

1.2 Brief Description

A timetable scheduling using genetic algorithm which resolves all the conflicts and clashes in timetable efficiently. Any problem ` has a set of valid results. This is said to form the solution space. In an optimization problem, the goal is to find results that maximize a set of criteria. A Genetic Algorithm is a method for efficiently searching the solution space. Timetable scheduling is complex optimization problem. Such problems where no efficient algorithm is known are ideal applications for genetic algorithms to be used to search a solution space. It is also important to realize that such scheduling is a real world problem that has immediate application in various forms of timetabling.

1.3 Problem Definition

The project consist of a genetic algorithm approach to a timetable scheduling problem. There are many conflicts in bad timetables which is hard to resolve ,hence using genetic algorithm it can be resolved easily and automatically generate a timetable. the configuration file is given as input which consist of all the necessary fields required such as no. of teachers ,no. of subjects, no. of working days, class timings ,etc. Genetic algorithm processes this input by genetic methods such as crossover mutation ,etc. and automatically generate timetable satisfying all constraints and resolving all conflicts.

2 SYSTEM ARCHITECTURE

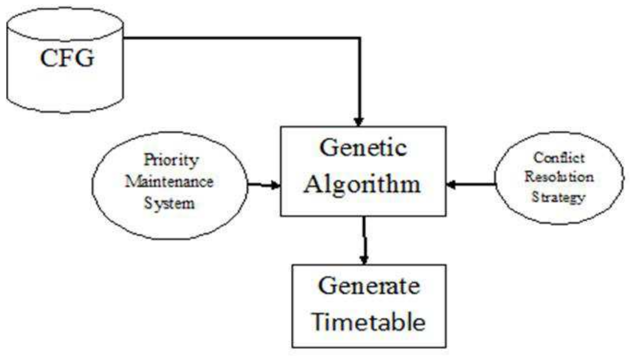

Fig 1: System Architecture

System architecture consist of a configuration file in which provide all the input fields required for timetable. Genetic algorithm take this input and performs operation on it, resolve conflicts, satisfy priority and automatically generate timetable.

2.1 Usecase Diagrams

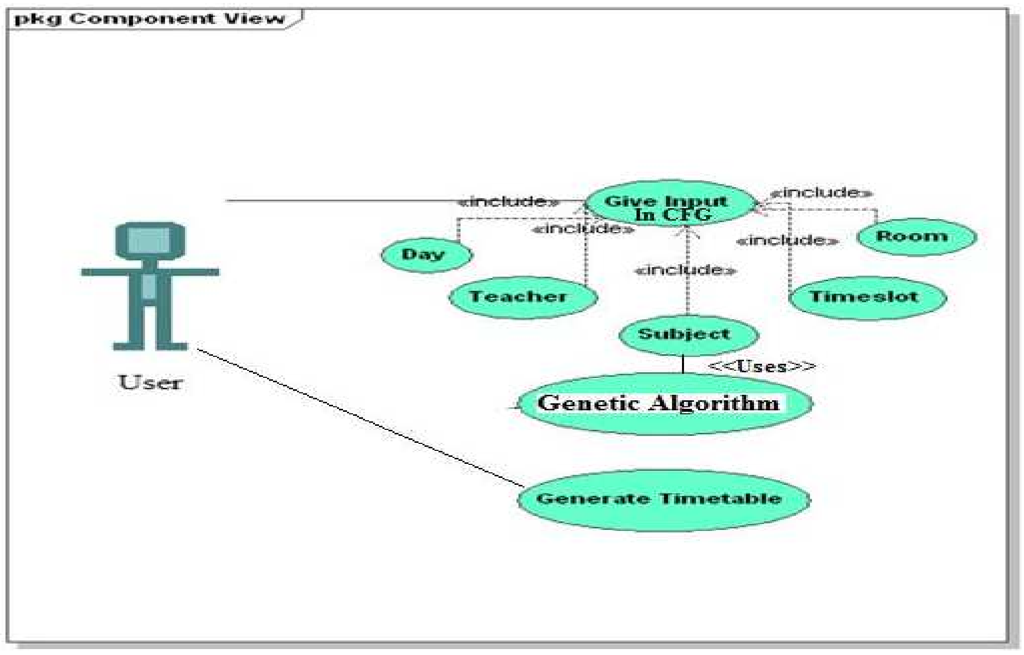

Fig 2: Usecase Diagram(9)

2.2 Algorithm

1.Create a population of objects (creatures) //Initialization

2.Evaluate the fitness of each object //analyzing

3.While the population is not fit enough //Fitness check

4.Delete all unfit objects //Removing unfit objects

5.While population size ¡max: //size check

6.Select two best populations

7.Create new objects

8.Random mutations

9.Evaluate and place in population //breeding

3 Flow of Genetic Algorithm

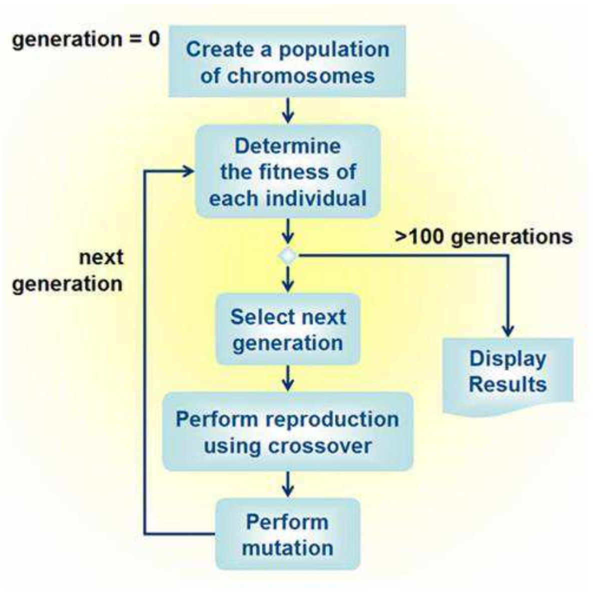
generation = 0
Create a population of chromosomes
Determine the fitness of each individual
next generation
>100 generations
Select next generation
Display Results
Perform reproduction using crossover
Perform mutation

Fig 3 : Flow of Genetic Algorithm[7]

3.1 Encoding of chromosomes:

certain amount of art is involved in selecting a good decoding technique when a problem is being attacked. The first place one starts when implementing a computer program is often in choosing data types and that is where the major variation occurs in different approaches. In genetic algorithm, chromosomes are encoded as a string of binary digits. A number of properties of binary encoding work to provide simple, effective and elegant GAs. There are, however, many other ways to represent a creatures genes, which can have their own implicit advantages.

3.2 Population Size:

The first step in a GA is to initialize an entire population of chromosomes. The size of this population must be chosen. Depending on the available computing techniques, different sizes are optimal. If the population size chosen is too small then there is not enough exploration of the global search space, although convergence is quicker. If the population size is too large then time will be wasted by dealing with more data than is required and convergence times will become considerably larger (Goldberg, 1989).

3.3 Evaluating a chromosome:

Random populations are almost always extremely unfit (Davis, 1991). In order to determine which are fitter than others, each creature must be evaluated. In order to evaluate a creature, some knowledge must be known about the environment in which it survives. This environment is the partially encoded (or partially decoded) description of the problem (Gen and Cheng). Depending on the way we structure the method of evaluating a chromosome we can either aim to generate the least costly population or the most fit; it is a question of minimizing cost or maximizing fitness.

3.4 Initializing a Population:

There are two general techniques for initializing a population. A population of creatures (all of the genetic information about all of the creatures in the colony) can be loaded from secondary storage. This data will then provide a starting point for the directed evolution. More commonly the GA can start with a random population. This is a full sized population of creatures whose genetic makeup is determined by a random process (Davis, 1991).

3.5 Methods of Selecting for Extinction or for Breeding:

Once a full population of creatures is established, each with a measure of fitness (or of cost) we can find an overall fitness. If the overall fitness is not yet as high as is desired a portion of the least fit creatures in the population can be selected for extinction. In various GAs, the method of selecting creatures for breeding is handled in different ways .Holland’s original model uses a method where the healthiest are most likely to breed (Gen and Cheng, 1997). Other methods select any two creatures at random for breeding.

3.6 Crossover:

Once parents have been chosen, breeding itself can then take place. A new creature is produced by selecting, for each gene in the chromosome, an allele from either the mother or the father. The process of combining the genes can be performed in a number of ways. The simplest method of combination is called single point cross-over. This can be best demonstrated using genes encoded in binary, though the process is translatable to almost any gene representation (Davis, 1991). The process of crossover can be performed with more than one crossover point

Two parents have already been selected:

PARENT1: 1011010101010010010010011100111001101010111011101

PARENT2: 0101001110110101011101010010011010110010100101110

Choose a crossover point:

PARENT1: 1011010101010010 010010011100111001101010111011101

PARENT2: 0101001110110101 011101010010011010110010100101110

Perform crossover to produce a child:

CHILD: 1011010101010010 011101010010011010110010100101110

Which then becomes, a whole new chromosome:

CHILD: 10110101010100100111010100100110101100101001010110

FIGURE : An Example of Crossover with Fully Encoded Genes

3.7 Mutation:

After crossover is performed and before the child is released into the wild, there is a chance that it will undergo mutation. The chance of this occurring is referred to as the mutation rate. This is usually kept quite small (Davis, 1991). The purpose of mutation is to inject noise, and, in particular, new alleles, into the population.

3.8 Constraint Data

In order to test for each of the types of hard constraint it is necessary to store sufficient detail about the timetable requirements. This means that information concerning all lecturers, classrooms and classes must be maintained. The way in which each of these data types are implemented will now be given in detail. A lecturer is a structured data type with one field- an availability timetable. An availability timetable is an array which indicates (using 1s and 0s) whether the lecturer is available or unavailable to lecture during each hour in the week. In this way prior commitments can be taken into consideration. A class is a structured type with three fields. Each class has a lecturer number and a number indicating the code number of the group of related classes to which it belongs. For example, core first year engineering subjects might form one set of related classes, and would each have the same number in their related class field. Each class can only be listed as belonging to one set of related classes. This is a simplification which ignores more complex relations between classes, for example, where classes are studied by more than one faculty. The set of all lecturers is stored as an array. Similarly, there are arrays of classes and of classrooms. Each of these elements is identified uniquely by its position in the relevant array.

3.9 Repair Strategy

A repair strategy is used which ensures that all classes appear exactly once. For robustness, this is done in two stages. Firstly, any classes which appear more than once are (non deterministically) altered such that they appear only once. Secondly, any classes which did not appear at all are booked to a spare space (regardless of room size, etc) If this repair strategy is applied to an empty timetable the result is a timetable with each class booked to a random time and place. As such, the repair strategy is also used for initial a random population. The use of the repair strategy ensures that each class is booked exactly once. Hence, the number of hard constraints which must be considered when timetables are being evaluated is further reduced. It has so far been shown that the hard constraints classrooms must not be double booked and every class must be scheduled exactly once have been satisfied by non genetic means.

4 Conclusion:

Genetic algorithm resolves the timetable scheduling efficiently. It resolve conflicts and automatically generate timetable. Genetic algorithm gives a optimal solution to problems. We can implement this time table scheduling using genetic algorithm application in our college also.

ACKNOWLEDGMENT

We render our sincere thanks to Prof. V. S. More (Lecturer, Computer department) RMCET,Devrukh Mumbai University for his guidance and encouragement.

References:

[1]. Anandaraj soundarya Raja Murugan,”*University Timetabling Using Genetic Algorithm*”, *Dalarna University, March 2009*

[2].Branimir Sigl, Marin Golub, Vedran Mornar” *Solving Timetable Scheduling Problem by Using Genetic Algorithms*”, *Faculty of Electrical Engineering and Computing, University of Zagreb*

[3]”Genetic algorithm”, Wikipedia(2011), the free Encyclopaedia. http://en.wikipedia.org/wiki/Mainpage.

[4]. Jiří Voráč, Ivo, Vondrák, Karel Vlček” *SCHOOL TIMETABLE GENERATING USING GENETIC ALGORITHM*” *VSB – Technical University of Ostrava Czech Republic*

[5].Leon Bambrick,”*Lecture Timetabling Using Genetic Algorithms*”, *University of Queensland*

[6]. M. Grobner, et al., "A standard framework for timetablingproblems ," Proc. International Conference on the Practice and Theory of Automated Timetabling 2002, LNCS 2740, pp.24-38, 2003.

[7]. Salwani Abdullah, Hamza Turabieh” Generating University Course Timetable Using

Genetic Algorithms and Local Search" *Center for Artificial Intelligence Technology Univesiti Kebangsaan Malaysia*, Third 2008 International Conference on Convergence and Hybrid Information Technology

[8]. Sehraneh Ghaemi, Mohammad Taghi Vakili, Ali Aghagolzadeh" *USING A GENETIC ALGORITHM OPTIMIZER TOOL TO SOLVE UNIVERSITY TIMETABLE SCHEDULING PROBLEM" Faculty of Electrical & Computer Engineering University Of Tabriz*

[9]. Use Case Diagram the free Encyclopaedia . http://sciencedirect/wiki/Mainpage

[10]. Tom V. Mathew, "Genetic Algorithm" *Indian Institute of Technology Bombay*