Volume 2, Issue 2, February 2012 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Genetic Algoritham- An Effective Approach To Solve Real Application Problem

1 MS. A .S. DHANDE, [1]M.E.Student, Sipna's COE, Amravati
[2] PROF DR. S. A. LADHAKE, [2]Principal , Phd .M.E.(Electronics) Sipna's COE, Amravati
[3]MS.S.S. DHANDE [3]Lecturer ,M.E. (Cmps) Sipna's COE, Amravati

ABSTRACT:Genetic algorithms have been applied to a very wide range of practical problems, often with valuable results. This paper surveys just a few examples, to illustrate the diversity of approaches and to point to some of the considerations that have proved important in making applications successful. Because GAs provide a fairly comprehensible way to address a wide range of dffiult engineering and optimization problems producing good if not optimal results, it seems that the technology is finding its way into real-world use much more easily than, say, expert systems did.
Keywords: **Mutation, Crossover, Hill climbing, Random**

1. INTRODUCTION

Genetic algorithms have been applied to a very wide range of practical problems, often with valuable results. This paper surveys just a few examples, to illustrate the diversity of approaches and to point to some of the considerations that have proved important in making applications successful.

The annual construction of an Exam Timetable is a common problem for all institutions of higher education. Scheduling course time tables for large modular courses is a complex problem which often has to be solved in university departments. This is usually done 'by hand', taking several days or week of iterative repair and after feedback from students complaining that the time table is unfair to them in some way.[7] Which we in this research tried solving through conventional method but at a certain level we recognized that solving timetabling problem with increased constraints is very difficult by conventional methods. A highly constrained combinatorial problem, timetabling can also be solved by evolutionary techniques. In this research we are showing evolutionary based genetic algorithm approach as an effective solution to solve course timetabling problem.

Timetable scheduling is the problem of assigning courses or exams to periods and to rooms. There are two types of University schedule: the course timetable and the exam timetable. These are related to each other but can be quite different.

For example, generally, more than one exam will be held in each exam hall at any particular time whereas it would be extremely unlikely that any institution would allow two courses to take place in the same room. Also, the halls are shared between all departments within the institution as opposed to each department using its own rooms. This means that, practically, the exam scheduling process must be carried out centrally by the university. For the purposes of this paper we will consider exams rather than courses.

The methods described, however, would be equally applicable to course timetabling. In course timetabling the hard constraint will be that at a time lecturer cannot take lecture in two different classes and after each lecture at least one lecture relaxation should be given to the lecturer. Timetabling constraints are many and varied. In this research, genetic algorithm approach is applied for solving university course timetabling problem. GA is a way of addressing hard search and optimization problems which provides a good solution although it requires large execution time.[8]

Constraints Involved

The constraints that we treat are classified as hard and soft. Hard constraints are those to which a time table has to adhere in order to be satisfied.

Hard constraints involved are:

1) No participant (lecturer or class) can be in more than two rooms at the same period.

2) No room should be double booked.

3) The room capacity should be large enough to hold each class.

Violating the above constraints will cause the time table to be unfeasible. In addition, we would also like to satisfy as many soft constraints as possible in order to produce a good quality timetable. Soft constraints for this constrained optimization problem are actually the students and lectures preferences which can be as follows.

1) The second time for each subject should not be in the same day.

2) No subject should be allocated to a time period that heads of department don't demand because of other work.

3)The subject should not be allocated to a time period inconvenient for a lecturer.

1. GENETIC ALGORITHM

Genetic Algorithms are powerful general purpose optimization tools which model the principles of evolution [3]. They are often capable of finding globally optimal solutions even in the most complex of search spaces. They operate on a population of coded solutions which are selected according to their quality then used as the basis for a new generation of solutions found by combining (crossover) or altering (mutating) current individuals.

Traditionally, the search mechanism has been domain independent, that is to say the crossover and mutation operators have no knowledge of what a good solution would be . A Genetic Algorithm starts by generating a set (population) of timetables randomly. These are then evaluated according to some sort of criteria. An example would be how many times any student has to sit two exams in a row. On the basis of this evaluation *population members* (timetables) are chosen as parents for the next generation of timetables. By weighting the selection process in favour of the better timetables, the worse timetables are eliminated while at the same time the search is directed towards the most promising areas of the search space (see figure 1).

The crossover operator works by taking two *population members* and combining them somehow to produce one or two offspring. Traditionally this is done by randomly selecting a point (gene) in the coded solution then appending the part of the second solution after that point to solution one up to that point and vice versa. The mutation operator is only applied to one solution at a time and involves the random variation of one particular gene. This adds a limited random element into the search and may reintroduce potentially useful *genetic material* that has been lost earlier in the search

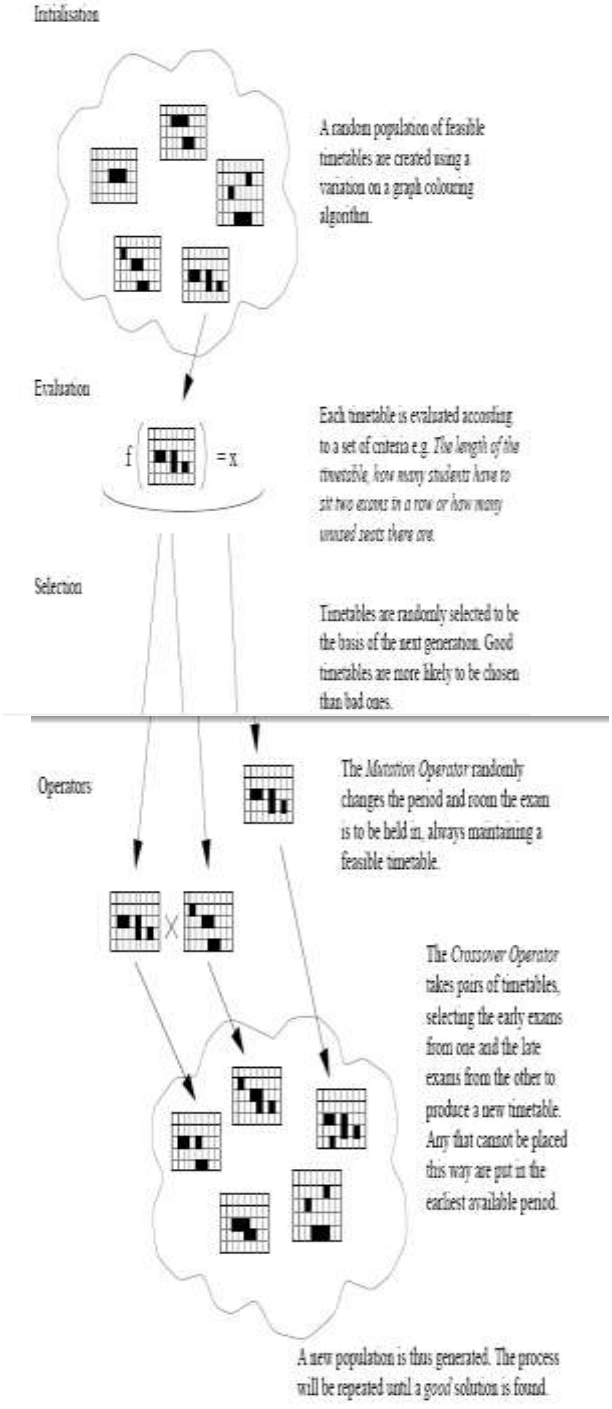

Figure 1: The Genetic Algorithm Process

The genetic algorithm maintains a population of schedules and constructs subsequent generations by adding new random schedules, making copies of high quality schedules in the present generation, applies the mutation operator to schedules, or uses the crossover operator on 2 schedules. When performing mutation or crossover, the schedules are selected with a probability proportional to their fitness, allowing the schedules with high quality to be selected more frequently and make more "offspring".

The free parameters in our GA are the size of the population, number of generations to simulate, and proportions of new generations to be made by the copy, random, mutation, and crossover methods.[4]

Representation: We use the naive representation of schedules.

A schedule contains lists of size containing the room in which each course's exam is to occur and the timing (corresponding to a pair or integers representing the day and time) of each course's exam. We will see that this representation is far from ideal, but there is no clear alternative.

Mutation Operator: The mutation operator takes a probability and a schedule. We loop over the courses, and, with the given probability, each is or is not set to a random room at a random time.

Crossover Operator: Given two schedules, the crossover operator loops over the courses, and chooses at random which schedule to take the room and timing from for that course.[1]

2. OTHER SEARCH ALGORITHM

Random Search: The random search method simply generates random schedules, evaluates their fitness, and remembers the one that it has seen that has highest fitness. This method is meant as a baseline against which to compare the other methods. If a search method doesn't perform significantly better than random search with the same number of fitness evaluations, it is useless.

Hillclimbing Search : The hillclimbing search that we have implemented examines every schedule in the neighbor hood of a given starting schedule, and then continues from the

one with highest fitness, terminating when it has performed as many evaluations as it is allowed to, or it has found a schedule whose neighbor are all equal or inferior to it.

The neighboring schedules are those where only one course meets in either a different room, on a different day, or at a different time. It can be seen that this method is completely deterministic given a starting schedule, and will always return a local maximum.

Mutate Search: Mutate Search starts by generating a random schedule. It makes an alternative schedule (by using the same mutate operator made for the genetic algorithm) with 1% of the data randomized and keeps the better schedule.

This operation is repeated until it has performed as many evaluations as it is allowed to.

This method can be changed to true simulated annealing by varying the degree of modification made to the current best schedule with the number of evaluations remaining. Not being the focus of our investigation, little effort was made to optimize the parameters.[3]

3. TIME TABLE GENERATION

Two fundamental constraints govern the production of a timetable. These are that no student or invigilator can be in more than one place at a time and that there must be sufficient seats to house all the students present. We will call a timetable that satisfies these constraints a *feasible* timetable.

Just because a timetable is feasible, unfortunately, does not mean it is good enough to be used. Many other criteria exist which may be used to judge the quality of a timetable. The most common of these is that a student should not be expected to sit two exams in adjacent examination periods. A particular institution may also wish that only exams of a similar length are scheduled at the same time in the same room or that larger exams come first to allow more time for them to be marked. In the end, the only real way to judge whether a timetable is a good one or not is if the institution will use it. In this paper we will describe a prototype genetic algorithm based timetabling system which will allow feasible timetables to be optimized as the particular institution sees fit through the use of an appropriate graphical user interface.

Genetic Algorithms have been successfully used to schedule exams in a number of cases. Corne et al. use a fairly traditional approach where each gene represents the time at which its particular exam takes place with crossover and mutation operators [3]. This system, with a number of time saving improvements, is in current use at the University of Edinburgh. Paechter takes a different approach where the gene for each exam not only specifies when it is to be taken but also how to search for a new period if, after crossover, the exam is causing a conflict. If the exam

may not be placed in any of the periods then it is left as unscheduled unlike the previous system which allows infeasible timetables.

4. CONCLUSION:

This paper helps to guide how genetic algorithm is a powerful method for solving real world problems. GA helps to solve difficult engineering and optimization problems. Implementation of GA in timetabling solve problems comes in conventional method.

Factors for consideration of Genetic algorithm include:

• Handling many different forms of timetabling constraint while only ever dealing with feasible timetables.

• Generating high-quality solutions despite the increasing intractability which has resulted from modularization.

• Providing a choice of several different good schedules from which the user may choose the best.

• Directing the timetable to the most constrained parts of the timetable so that, if necessary, adjustments may be made manually.

6. REFERENCES

[1] Formulation of Genetic Algorithm to Generate Good Quality Course Timetable-International Journal of Innovation, Management and Technology, Vol. 1, No. 3, August 2010

ISSN: 2010-0248

[2]Solving an Exam Scheduling Problem Using a Genetic Algorithm

[3]A Genetic Algorithm Based University Timetabling System, Edmund Burke, David Elliman and Rupert Weare

Department of Computer Science,University of Nottingham,University Park, Nottingham, NG7 2RD.

[4] David E. Goldberg, 1989, "Genetic Algorithms in search, optimization and machine learning".

[5] Bruns R. (1993) "Knowledge-Augmented Genetic Algorithm for Production Scheduling", IJCAI '93 Workshop on Knowledge based Production Planning, Scheduling and Control.

[6] Applicationsofgeneticalgorithms by PeterRoss andDaveCorne Department ofAI,UniversityofEdinburgh 80SouthBridge,Edinburgh

[7] A.T.Rahmani and N.Ono. A genetic algorithm for channel routing problem. In S.Forrest, editor, Proceedings of ICGA-93, pages 494{498, San Mateo,

[8] [3] C.Lin and P.Hajela. Genetic search strategies in large scale optimisation. In AIAA Structures, Structural Dynamics and Materials Conference,