International Journal of Computational Engineering Research||Vol, 03||Issue, 4||

A Phased Approach to Solve the University Course Scheduling System

Rohini V,
Department of Computer Science, Christ University, Bangalore

Abstract:
Scheduling is an important activity of our life. Course scheduling is a complicated task faced by every university. This paper presents a phased approach to solve the constraint satisfaction problem of scheduling the resources. Here the idea is to use a genetic algorithm which is a good solution for the NP hard problems like scheduling. The paper starts by defining the problem of scheduling the courses in Christ University, Department of computer science and defining the various complicated constraints available. Then the problem model is explained with the set of resources like professors, rooms, labs, student etc. Finally the paper describes the different phases used to get the final well defined schedule[1].

Keywords: *Constraints, Course scheduling, Genetic algorithm, NP hard problem, Phased, Hard constraints, Soft constraints*

I. INTRODUCTION

Course scheduling in a university is a complex manual work done twice in a year. The problem involves fixing the time slots for the subjects, allocating the resources (lab, class rooms, projectors), allocating the staff satisfying the hard constraints and minimize the cost of a set of soft constraints. The constraints will not be common for all universities. It will be practised according to the constraints of the departments.

II. GENETIC ALGORITHM

In 1975, Holland developed the idea of genetic algorithm idea in his book "Adaptation in natural and artificial systems"[2]. He described how to apply the principles of natural evolution to optimization problems and built the first Genetic Algorithms. Holland's theory has been further developed and now Genetic Algorithms (GAs) stand up as a powerful tool for solving search and optimization.An algorithm is a series of steps for solving a problem. A genetic algorithm is a problem solving method that uses genetics as its model of problem solving. It's a search technique to find approximate solutions to optimization and search problems[4].Basically, an optimization problem looks really simple. One knows the form of all possible solutions corresponding to a specific question. The set of all the solutions that meet this form constitute the search space. The problem consists in finding out the solution that fits the best, i.e. the one with the most payoffs, from all the possible solutions. If it's possible to quickly enumerate all the solutions, the problem does not raise much difficulty. Genetic Algorithms work in a similar way, adapting the simple genetics to algorithmic mechanisms.GA handles a population of possible solutions. Each solution is represented through a chromosome, which is just an abstract representation. Coding all the possible solutions into a chromosome is the first part, but certainly not the most straightforward one of a Genetic Algorithm. A set of reproduction operators has to be determined, too. Reproduction operators are applied directly on the chromosomes and are used to perform mutations and recombinations over solutions of the problem. Appropriate representation and reproduction operators are really something determinant, as the behavior of the GA is extremely dependant on it.

Frequently, it can be extremely difficult to find a representation, which respects the structure of the search space and reproduction operators, which are coherent and relevant according to the properties of the problems.Selection is supposed to be able to compare each individual in the population.Selection is done by using a fitness function. Each chromosome has an associated value corresponding to the fitness of the solution it represents. The fitness should correspond to an evaluation of how good the candidate solution is. The optimal solution is the one, which maximizes the fitness function. Genetic Algorithms deal with the problems that maximize the fitness function. But, if the problem consists in minimizing a cost function, the adaptation is quite easy. Either the cost function can be transformed into a fitness function, for example by inverting it; or the selection can be adapted in such way that they consider individuals with low evaluation functions as better.

Once the reproduction and the fitness function have been properly defined, a Genetic Algorithm is evolved according to the same basic structure. It starts by generating an initial population of chromosomes. This first population must offer a wide diversity of genetic materials. The gene pool should be as large as possible so that any solution of the search space can be engendered. Generally, the initial population is generated randomly.
Then, the genetic algorithm loops over an iteration process to make the population evolve. Each iteration consists of the following steps:

- SELECTION: The first step consists in selecting individuals for reproduction.
- This selection is done randomly with a probability depending on the relative fitness of the
- individuals so that best ones are often chosen for reproduction than poor ones.
- REPRODUCTION: In the second step, offspring are bred by the selected individuals.
- For generating new chromosomes, the algorithm can use both recombination and mutation.
- EVALUATION: Then the fitness of the new chromosomes is evaluated.
- REPLACEMENT: During the last step, individuals from the old population are killed and replaced by the new ones.
- The algorithm is stopped when the population converges toward the optimal solution.
- The basic genetic algorithm is as follows:
- [start] Genetic random population of n chromosomes (suitable solutions for the problem)
- [Fitness] Evaluate the fitness f(x) of each chromosome x in the population
- New population] Create a new population by repeating following steps until the New population is complete
- [selection] select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to get selected).
- [crossover] With a crossover probability, cross over the parents to form new offspring (children). If no crossover was performed, offspring is the exact copy of parents.
- [Mutation] With a mutation probability, mutate new offspring at each locus (position in chromosome)
- [Accepting] Place new offspring in the new population.

2.1 A Simple Genetic Algorithm

- [Replace] Use new generated population for a further sum of the algorithm.
- [Test] If the end condition is satisfied, stop, and return the best solution in current population.
- [Loop] Go to step2 for fitness evaluation.

The Genetic algorithm process is discussed through the GA cycle in Fig. 1.

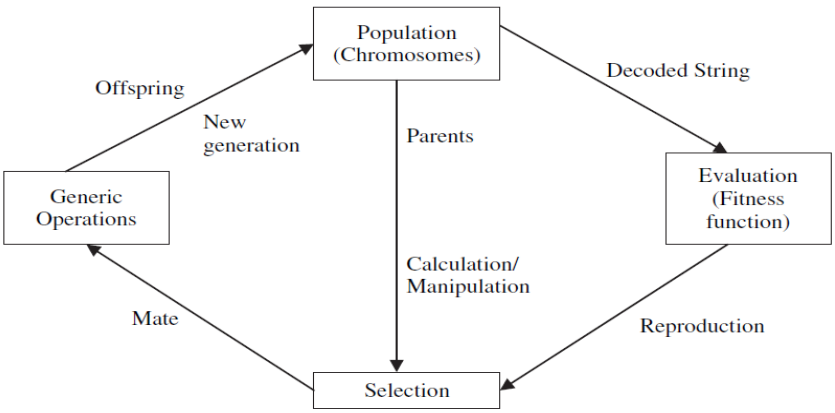

Fig.1 Genetic algorithm cycle

Reproduction is the process by which the genetic material in two or more parent is combined to obtain one or more offspring. In fitness evaluation step, the individual's quality is assessed. Mutation is performed to one individual to produce a new version of it where some of the original genetic material has been randomly changed. Selection process helps to decide which individuals are to be used for reproduction and mutation in order to produce new search points.

The flowchart showing the process of GA is as shown in Fig. 2.

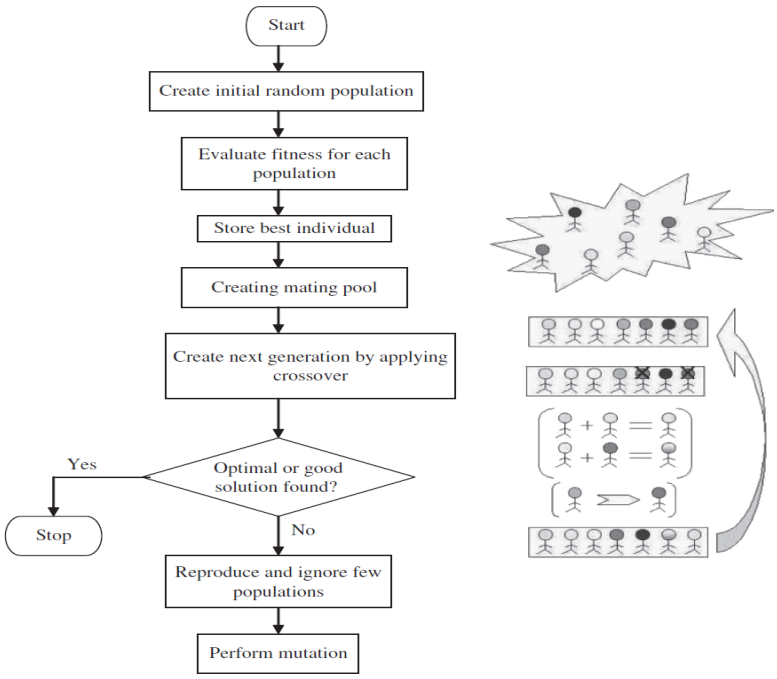

Fig.2 Genetic algorithm flowchart

Before implementing GA, it is important to understand few guidelines for designing a general search algorithm i.e. a global optimization algorithm based on the properties of the fitness landscape and the most common optimization method types:

[1] determinism:A purely deterministic search may have an extremely high variance in solution quality because it may soon get stuck in worst case situations from which it is incapable to escape because of its determinism. This can be avoided, but it is a well-known fact that the observation of the worst-case situation is not guaranteed to be possible in general.

[2] nondeterminism: A stochastic search method usually does not suffer from the above potential worst case ”wolf trap” phenomenon. It is therefore likely that a search method should be stochastic, but it may well contain a substantial portion of determinism, however. In principle it is enough to have as much nondeterminism as to be able to avoid the worst-case wolf traps.

III. EMPIRICAL STUDY

In this section, a real world university course scheduling system is represented and is solved by the genetic algorithm described above.The Department of Computer Science, Christ University, Bangalore is a big department consisting of various courses in computer science as BCA, B.Sc., MCA, M.Sc. and consists of thirty professors. Scheduling the subjects to the various course in the allotted time slot and resources is a complex hard problem. The job involves of scheduling the various subjects to the course , various staff and resources adhering to the constraints stated below. The problem consists of the following entities:

Days, Timeslots, and Periods.:We are given a number of teaching days in the week (typically 6). Each day is split in a fixed number of timeslots of 6 (9 am to 4 pm, except 1 – 2 pm for lunch) and Saturdays only 4 hours (9 am – 1 pm). A period is a pair composed by a day and a timeslot. The total number of scheduling periods is the product of the days times the day timeslots.

Courses and professors: Each course consists of a fixed number of lectures to be scheduled in distinct periods; it is attended by given number of students, and is taught by a teacher. For each course there is a minimum number of days that the lectures of the course should be spread in, moreover there are some periods in which the course cannot be scheduled.

Rooms.:Each room has a capacity, expressed in terms of number of available seats. All rooms are equally suitable for all courses (if large enough). Rooms will have the resources like board(white / black), Projector etc.

The timetables are feasible if and only if the following hard constraints are satisfied:
No instructor teaches more than one course at a given time.
Number of courses taught during any time slot should not exceed the maximum number of classrooms and labs available at that specified time.
Restrictions on the starting time, ending time, lunch time.
Lectures, seminars and labs of each group of students should not overlap.
Practical courses should run on two continuous time slots.
Class timings is 9 -4 with a break of lunch from 1-2 pm.
Subjects of other department should be accommodated in their possible time slot.
While two Elective of one course is happening, one more class should be in the lab.
The following soft constraints should be satisfied if possible:
There should be a gap of at least one hour in the schedule between courses taught by the same instructor.
Course should be spread over evenly throughout the week (Monday to Saturday).
All lectures of a particular course should be assigned to the same room.
Students can attend only one lecture at a time.
All first hour (9-10 am) should be allocated to different subject/faculty.
The problem is solved in a phased approach as follows :[5]
Phase 1: Allot the subjects to the professors.
Phase II : Allot the labs to the courses..
Phase III: Assign lectures to time slots.
Phase IV: Assigns labs and tutorials to days and available time slots in the days.
The flow of data as allocation of subjects of the various courses to the professors are done and given to the committee. The scheduling committee is responsible for allotting the time slots, resources to the various courses satisfying the above mentioned constraints. According to the genetic algorithm method described in the paper and the relationship of those events and resources, a course scheduling system is developed. Finally, a solution which has the best fitness is obtained. Through analysis of the time table prepared, the constraints are satisfied and some soft constraints are not able to satisfy. It is demonstrated that the method in this problem is feasible and widely accepted.

IV. CONCLUSION

University course scheduling system is a NP hard problem[1].In this paper, we have discussed about the real life case study of how the timetabling is done with the constraints and the goal of optimization etc. Genetic algorithm, which is a commonly used evolutionary algorithm is applied to solve the problem described. During the process, genetic representation and a fitness function are defined according to the problem.

REFERENCES

[1] Bratković, Z., Herman, T., Omrčen, V., Čupić, M., Jakobović, D.: University course timetabling with genetic algorithm: A laboratory exercises case study. In: Cotta, C., Cowling, P. (eds.) EvoCOP 2009. LNCS, vol. 5482, pp. 240–251. Springer, Heidelberg (2009)
[2] J.H. Holland , Adaptation in Natural and Artificial systems, Cambridge, The MIT press, 1992
[3] D.Goldberg, Genetic Algorithm in Search, Optimization and Machine Learning", Addison Wesley, 1989
[4] S.N.Sivanandam and S.N.Deepa , Introduction to Genetic Algorithms, Springer, 2007
[5] Minhaz Fahim Zibran, A Multi-phase Approach to University Course Timetabling, University of Lethbridge (Canada), 2007