European Journal of Business and Management
ISSN 2222-1905 (Paper) ISSN 2222-2839 (Online)
Vol.7, No.4, 2015

Genetic Algorithm as a General Approach to Time Tabling Problem

Tarun Jain[1*] Needa Jamil[2]
1.Faculty of Mechanical Engineering, University of Auckland, Auckland, New Zealand
2.Department of Design, Indian Institute of Technology Guwahati, Assam 781039, India
* E-mail of the corresponding author: tjai943@aucklanduni.ac.nz

Abstract
Optimization of Time Table has traditionally and for long been done manually, based on experience of its human solver. Various optimization techniques have been developed for solving specific cases of this problem but no general solution or approach exists as of now. In this paper we present a GA based approach that can be generalized to most of these problems. Timetable for Department of Mechanical Engineering, IIT Guwahati has been solved using this approach as a case study and subsequent results and modifications have been presented in this paper.
Keywords: Genetic Algorithm, Time Table problem

1. Introduction

This paper describes an implementation of genetic algorithm on timetable scheduling problem that is common to all educational institutions. Primary goal of algorithm is to find a feasible solution satisfying all the constraints; modifications for further improvement of solution from the perspective of students and/or teachers have been discussed in short. A general approach has been discussed and then modified for the specific case study of 'Department of Mechanical Engineering, IIT Guwahati'. The core of this genetic algorithm has been developed in MATLAB.

2. Time Table Scheduling Problem, in general

The scheduling problems deal with effective distribution of limited resources. In case of Time Table scheduling, the limited resources can be categorized in following domains:

1. Rooms or equipment
2. Teachers or Instructors
3. Time or Slots

The task i.e. Lecture (or lab) can have fixed number of instances or hours per week and may need any number of above-mentioned limited resources for their realization. The goal is to assign these tasks to their resources while satisfying all the obvious and not so obvious constraints (availability of teacher, feasibility, human factors etc.)

The timetable can be visualized as a 3D structure (or N dimensional structure, depending upon the number of limited resources), with days (Monday to Friday, or Saturday) on one axis, time slots second axis and rooms on third axis. Each task or course can have its own set of attached resources like its Instructor and enrolled students that are not common to all tasks. Hence, our task is simplified, as we need to only assign day, slot and room to each course while checking for all individual and overall constraints.

Now, each course (including labs etc.) can be defined as C_{ij} where 'i' is the course number and j is the instance of course (i.e. 1, 2, 3…. depending on the number of times it should be held in one week). Each course will have its own list of students (or exclusive groups g_1, g_2, g_3 etc.) who will be attending these lectures and a list of instructors. We will assign a day 'd' , slot 's' and room 'r' to each instance of each course to constitute our solution. In general, following constraints should not be violated in a feasible solution:

a) Each group g_i can attend only one class at one time
b) Instructor 'I' can only teach one class at one time
c) Only one class can be held in a room 'r' at one time
d) All courses should have 'j' instances in a week

For our algorithm, we start by giving integer values to all the courses (1, 2, 3…… N_c), teachers or instructors (1, 2, 3…..N_i), rooms (1, 2, 3…..N_r), days (1, 2, 3, 4, 5 for Monday to Friday schedule) and slots (1, 2, 3…….. N_s).

N_c: Total number of courses that need to be scheduled
N_i: Total number of teaching staff or instructors in institution
N_r: Total number of rooms/labs/equipment available
N_s: Total number of time slots (e.g. If minimum time required for any course is 't' and total time available for teaching in one day excluding the breaks is 'T' , then $N_s = T/t$)

3. Genetic Algorithm based approach

Genetic Algorithms are nature-inspired adaptive evolutionary algorithms which can be used for solving complex problems as well as for searching large problem spaces. In GA, every possible solution is thought of as 'individual' and many such individuals or a set of solutions makes up the 'population' after any generation. First set of solutions or the 'initial population' can be generated randomly; individuals are then randomly mated allowing the recombination of genetic material. The resulting individuals can then be mutated with a specific mutation probability. The new population so obtained undergoes a process of natural selection that favors the survival of the best solutions, and provides the basis for a new evolutionary cycle [1].

Let P denote a population of N individuals. Let P (0) be the initial population, randomly generated and P(t) the population at time 't' or the 't^{th}' generation. The GA generates a new population or next generation P (t+1) from P (t) using various genetic operators. The three basic genetic operators are:

1. Reproduction or Selection operator ensures the survival of better solutions, increasing its copies in the next generation.
   One of the most common methods for selection is 'Tournament selection' in which some individuals are randomly picked and the best solution among them is copied to next generation. Such tournaments are repeated till we get our desired size of population.
2. Crossover is a genetic operator that is activated by probability P_c; two parents i.e. individuals are selected at random and some of their characteristics are swapped between one another to make two children for next generation.
3. Mutation is another genetic operator activated by probability P_m; individual is selected randomly and some or all of its characteristics are randomly changed i.e. mutated. This ensures randomness and prevents the algorithm from getting stuck at local optima.

Above mentioned operators are repeated till desired fitness or number of generations is achieved. In following subsections we have discussed the genetic operators and other such parameters for our problem.

3.1 Initial Population

for each course C_i (i goes from 1 to N_c) {
  for **each** course instance j (1 to No. of instances) {
    define day 'd' (random integer from 1 to 5)
    define slot 's' (random integer from 1 to N_s)
    define room 'r' (random integer from 1 to N_r)
  }
}

We can assign the rooms, slots and days a specific value if any of those are pre-defined i.e. frozen for that course. This generates a single individual or solution and is repeated N times to construct our initial population where N is the population size.

The group list 'G' consisting of student lists (g_1, g_2…. g_{Ng}) and instructor 'I' is already attached to each course and need not be defined for individual solutions.

3.2 Fitness

for each day 'd' (1 to 5) {
  for each slot 's' (1 to N_s) {
    Compute C_r (No. of common rooms)
    Compute C_g (No. of common groups)
    Compute C_i (No. of common instructors)
  }
}

Various additional penalty constants can be defined specific to the institution e.g. two instances of same course should not be held on the same day, every instance of a course should be held in same room etc.

Now, the fitness of every individual solution can be defined as the sum of all these penalty constants if we keep the aim of our algorithm to minimize the fitness or vice versa.

3.3 Selection\Reproduction

We have used a modified 'Tournament selection' method for the ease of coding and can be described as follows:

for n =1 to N (Population size) {
  Select 'X' individuals randomly from population
  Copy the individual with least fitness to P(t+1)
}

This generates 'N' individuals for next generation and 'X' should be chosen such that probability for selection of

best solution is moderate.

3.4 Mutation

We have modified the mutation operator to meet the needs of our algorithm as follows:

for each individual {
Activate Mutation by using probability P_{m1}
Once Activated
for each course instance C_{ij} {
Select using probability P_{m2}
Once Selected
randomize day 'd' using probability P_{m3}
randomize slot 's' using probability P_{m4}
randomize room 'r' using probability P_{m5}
}
}

Special courses with frozen slots or rooms should be exempted from mutation.

P_{m1}: Probability of mutation for individual
P_{m2}: Probability of selecting a course for mutation
P_{m3}: Probability of mutation for day 'd'
P_{m4}: Probability of mutation for slot 's'
P_{m5}: Probability of mutation for room 'r'

3.5 Crossover

We have modified the crossover operator as follows:

for each solution pair (1 to ${}^{N}C_{2}$) {
Activate crossover using probability P_{c1}
Once Activated
Select course using probability P_{c2}
Once Selected
Swap day, slot and room among the two solutions
}

Special courses with frozen slots or rooms should be exempted from crossover.
P_{c1}: Probability of crossover for individual pairs
P_{c2}: Probability of selecting a course as crossover site

3.6Solver

Once all of the above mentioned genetic operators have been defined, we can run our algorithm or solver which is described as follows:

Generate initial population P(0)
Calculate fitness for each individual in P(0)
While fitness ≠ 0 or iterations, $t \leq T$ {
Do crossover on P(t)
Do mutation on P(t)
Calculate fitness for each individual in P(t)
Store Minimum fitness of that iteration
Store Global minimum fitness
Do Selection on P(t)
Increase iteration count, t = t+1
}

In the next section we apply this GA based approach to the Timetable scheduling problem for Department of Mechanical Engineering, IIT Guwahati and discuss the practical constraints as well as the results.

4. Case Study

4.1 Constraints

Department of Mechanical Engineering, IIT Guwahati has a 5-day schedule with lecture hours being 8:00 am to 6:00 pm with a 1 hour lunch break and the minimum time required for any lecture is 1 hour. Therefore,

- day 'd' can vary from 1-5
- slot 's' can vary from 1-9

- room 'r' can vary from 1-4 (Only four rooms are available to department for lecture courses; all lab courses have their own respective labs)
- Total number of courses including labs = 39 Total number of Instructors = 30
- Total number of students in department = 390
- Some slots on specific days are frozen by the institute.
- Every lecture course should have 3 instances in a week and should be held in same room on different days.
- Lab courses should occupy 3 consecutive slots on same day.

Above mentioned parameters and constraints are specific to our case study, but the general constraints mentioned before are also valid here.

4.2 Explicit Groups

Other than above mentioned constraints and parameters we also have course-wise student registration list available in order to ensure that no student gets two or more classes scheduled at the same time. Hence, we have defined explicit groups which have following properties:

- All students in this group have enrolled for exactly same courses.
- All students in this group will have the same lecture schedule for a week.

Total number of exclusive groups for our case study = 105

4.3 Modifications

We need to accommodate additional penalty constants like;

UC_r: No. of uncommon rooms allotted to instances of same course.
C_d: No. of common days allotted to instances of same lecture course
F_s: No. of instances that were allotted frozen slots

The slots for lab courses are also fixed since they are held only in afternoon from 2:00 pm – 5:00 pm. Overall fitness function for our case study can be defined as.

$$F = C_r + C_g + C_i + C_d + UC_r + F_s$$

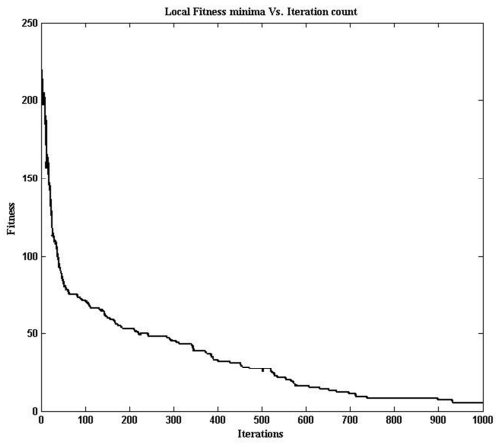
4.4 Results

We have used following values for the constants that are used in our algorithm:

- P_{m1}, P_{m2}, P_{m3}, P_{m4}, P_{m5} = 0.2
- P_{c1} = 0.8 ; P_{c2} = 0.4
- N = 50 (Population size)
- X = 3 (Tournament size)

This algorithm was run for around 1000 iterations (approx.30 minutes on a normal PC) and following results/observations were obtained:

Fig.1. Evolutionary process



4.5 Scope of improvement in algorithm

We have discussed a very general algorithm that can be used to solve any Timetable scheduling problem but there are various modifications which can be added into this algorithm, significantly improving its performance.

Most of these modifications can be specific to institutions but some basic improvements can be in the form of;

- Adaptability, probabilities for mutation and crossover can be changed according to the progress of computation.
- Defining Taboo/impossible moves
- Using (μ+λ) selection after crossover and mutation

5. Conclusion

The Timetable scheduling problem has been tackled through a genetic algorithm based generalized approach that can be used with a few modifications depending on educational institution. Our case study clearly shows the success of this algorithm in finding a feasible solution with a lot of scope in improving the performance using advanced genetic operators.

References

B. Sigl, M. Golub, and V. Mornar (2005) "Solving timetable scheduling problem by using genetic algorithms" , Faculty of electrical engineering and computing, University of Zagreb.

Carrasco, M.P., & Pato, M.V. (2001) 'A multiobjective genetic algorithm for the class/teacher timetabling problem.' In Proceedings of the Practice and theory of Automated Timetabling (PATAT'00), Lecture notes in ComputerScience, Springer, 2079,3-17.

D.E. Goldberg (1989) "Genetic Algorithms in Search", Optimization and Machine Learning, Addison-Wesley.

D. Abramson, J. Abela (1992) "A parallel genetic algorithm for solving the school timetable problem", 15. Australian Computer Science Conference, Hobart.

Datta, D., Deb, K., & Fonseca, C.M. "Multi-objective evolutionary algorithm for university class timetabling problem" , In Evolutionary Scheduling, Springer-Verlag.

The IISTE is a pioneer in the Open-Access hosting service and academic event management. The aim of the firm is Accelerating Global Knowledge Sharing.

More information about the firm can be found on the homepage:
http://www.iiste.org

CALL FOR JOURNAL PAPERS

There are more than 30 peer-reviewed academic journals hosted under the hosting platform.

Prospective authors of journals can find the submission instruction on the following page: http://www.iiste.org/journals/ All the journals articles are available online to the readers all over the world without financial, legal, or technical barriers other than those inseparable from gaining access to the internet itself. Paper version of the journals is also available upon request of readers and authors.

MORE RESOURCES

Book publication information: http://www.iiste.org/book/

Academic conference: http://www.iiste.org/conference/upcoming-conferences-call-for-paper/

IISTE Knowledge Sharing Partners

EBSCO, Index Copernicus, Ulrich's Periodicals Directory, JournalTOCS, PKP Open Archives Harvester, Bielefeld Academic Search Engine, Elektronische Zeitschriftenbibliothek EZB, Open J-Gate, OCLC WorldCat, Universe Digtial Library , NewJour, Google Scholar