Class Schedule Generation using Evolutionary Algorithms

Mohit Kumar Kakkar¹, Jajji Singla¹, Neha Garg¹, Gourav Gupta¹, Prateek Srivastava¹, Ajay Kumar²
¹Chitkara University Institute of Engineering and Technology, Chitkara University, Punjab, India.
²Department of Mechanical Engineering, Faculty of Engineering and Technology, Shree Guru Gobind Singh Tricentenary University, Gurgaon, India.

E-mail address: prateek.srivastava@chitkara.edu.in, ajay.kumar30886@gmail.com

Abstract Timetabling problem is known as an NP-hard problem that centres around finding an optimised allocation of subjects onto a finite available number of slots and spaces. It is perhaps the most challenging issues looked by colleges around the globe. Every academic institution faces a problem when preparing courses and exam plans. There are many restrictions raised while preparing a timetable. This paper proposed a method based on the evolutionary algorithms to solve the constrained timetable problem, which helps to create theory as well as lab schedule for universities. A smart adaptive mutation scheme is used to speed up convergence and chromosome format is also problem specific. Here in this paper two algorithms are compared in respect of Timetabling problems. Using GA (Genetic Algorithm) and MA (Memetic algorithm), we optimised the output by selecting the best solution from the available options to present a comprehensive curriculum system.

1. Introduction
Finding a feasible class schedule in large university departments is a constant challenge. Timetable creation is a problem that needs to be placed in multiple time intervals to prevent conflicts (courses, exams, classes, etc.) when using a specific time slot. While generating the schedule for an institution, many constraints arise. These restrictions or constraints are basically divided into two types; one with time restrictions, room allocation, known as strict restrictions, and the other with faculty preference restrictions. Here, strict restrictions on this timetable issue cannot be avoided. We will discuss these strict restrictions in the "system constraints" section. These types of restrictions may vary in institutions. Manually solving this problem takes much time, and these solutions are not feasible in most ways. It may find many conflicts. The automatic schedule generator can help in minimising the valuable time consumed for the department of any institute or university.

This paper focuses on the issues of time table of course (curriculum) scheduling at the university level. Students and faculty members face many problems when preparing courses and exam plans. The goals of this work are:

- To create a suitable timetable system for the required courses.
- To find feasible solutions under many constraints.
The primary step to achieve the goals mentioned above is to discover the problems faced by faculty and staff when preparing the plan, collect relevant information about the problems to analyse it. From the demand point of view, the ultimate goal is to design the format of the timetable and utilise the algorithms to get the desired output in the form of a specific timetable.

2. Literature Survey
During the past few decades, numerous articles have been published in conferences and magazines relating to the automation of the timetable generator. The first automated time search was started by Gotlieb[1] in 1963. Dimopoulou and Milliotis [2] proposed programming procedures to handle this kind of scheduling problem. Due to complexity, most researchers are considering heuristic algorithms, such as Dave Corne et al. [3] utilised GA. An automatic timetable survey was presented by Schaerf [4].

Some researchers have recently focused on logic programming because they want to solve this problem naturally to represent constraints. However, there were some shortcomings with the unfulfilled performance [5]. Mooney and Lardin have used Tabu Search [6]. In 1967, Welsh et al. [7] proposed a graphic colouring method to solve the problem, which is of NP-Hard type.

Masri Ayob[8] described an exam timetable issue at Kebangsaan University in Malaysia. He proposed a useful objective function. Michael W. Carter [9] presented a tool called EXAMINE, which has already been used by universities of Toronto and Carleton. Cooper and Kingstone [10] have used simulated annealing. Kendall and Hussin [11] proposed a tabu search meta-heuristic method for ETP. Recently, many researchers [12] have applied different evolutionary computing algorithms like cuckoo search to the scheduling problem. Tavarani [13] used novel operators for quantum evolutionary algorithm for solving problem regarding timetabling. Kenekayoro [14] applied Machine Learning to solve Timetabling problem for university. Tavakoli et.al.[15] Proposed a novel algorithm for timetabling problem with the quality of courses rendered approach.

3. Proposed Approach
Genetic algorithm is an intelligent research technique used for solving NP-hard problems. It mimics the process of natural selection. It starts with a population of genes. Here we have compared our output from GA with the MA (memetic algorithm) also, which is basically an extension of the GA. It uses a local search technique to reduce the likelihood of premature convergence.

![Figure 1. General Genetic algorithm](image-url)
3.1 Genetic Algorithm Operators

**Chromosome:** It is a set of variables or values which define a solution to the problem that we are trying to solve with genetic algorithm. The set of all solutions is known as the population. The chromosome is generally represented as a binary or string variables.

**Initial population:** It is the collection of chromosomes. As bits represent chromosomes, the initial population is generally taken as a binary no.

**Mutation:** Mutation is the process of the random chance of a gene in the solution string of population and is very helpful in creating a solution or chromosome.

**Crossover:** A crossover is a process of using two chromosomes for intermixing them.

**Fitness function:** A fitness function is used to evaluate the fitness of a solution. A fitness value is assigned to each chromosome, and the optimal solution is basically that chromosome which has a maximum value of fitness.

For our system, we require data (which is to be entered by the users), system constraints and applications of genetic algorithm in it.

3.2 Input Data:

**Groups:** Format of this data of the class group is like [1, 'GP-10'] and contains the group’s name.

**Instructors:** Format of this data of instructors is like [E10,'XYZ'] and contains ID and name of the teacher.

**Rooms:** Format of this data of lecture room is like [[0,'PL-001',50,0] which contains room no., its capacity, and 0 for theory class or 1 for the lab.

**Courses:** Format of this data of courses is like ["MK-1501",1] which contains course code, and 0 for theory class or 1 for the lab.

![Figure 2. Flow chart of Time table Generator](image-url)
3.3 Parameters required for coding
We have used following parameters values for both GA and MA.
Population size = 100
Mutation probability = 0.025
Crossover probability = 0.90
Max. no. of generations = 500

Figure 3. General structure of Time table scheduling via genetic algorithm.

3.4 System Constraints:
System constraints are divided into two categories:
i. Hard Constraints: The time table schedule is subjected to some conditions or constraints that are unavoidable and known as hard constraints to get a valid result:
   - A student should appear in a session of class at one time.
   - A teacher can appear in a class for the respective course at one time.
   - At a particular time, one classroom should be booked for one class only.
   - If there is any requirement of equipment, for example, audio-visual equipment or projectors then it should be appropriately filled.

ii. Soft Constraints: These are the constraints that are not of much greater concern, or these can be avoided, but still are taken into account to obtain the satisfactory output.
   - Teaching hours or workload of every teacher should well spread over a week.
   - There should not be consecutive lectures of the same teacher in the class.
   - Assign fixed time slots for particular subjects.
   - Minimise continuous lectures of the same course in a day.
3.5 Coding Part:
As it is not possible to put all the coding portion here, so we are providing crossover (cross) and mutation (mut) functions using GA in Python.

```python
def Cross(p1, p2):
c = np.zeros((coursesLen, 3), dtype=int)
for i in range(len(p1)):
    if np.random.uniform(0.0, 1.0) < 0.5:
        c[i] = copy.copy(p1[i])
    else:
        c[i] = copy.copy(p2[i])
return c

def mut(chrom, mp):
    for i in range(len(chrom)):
        if np.random.uniform(0.0, 1.0) < mp:
            if np.random.random() < 0.5:
                randomRoom = random.randint(0, roomsLen-1)
                chrom[i][1] = randomRoom
            else:
                randomTimeSlot = random.randint(0, totalTimeSlot-1)
                chrom[i][2] = randomTimeSlot
    return chrom
```

Table 1. Timetable of a teacher

| Timing    | Monday       | Tuesday      | Wednesday    | Thursday     | Friday       |
|-----------|--------------|--------------|--------------|--------------|--------------|
| 09:00-09:55|              |              |              |              |              |
| 09:55-10:50|              |              | AM-101, TG-001|              |              |
| 10:50-11:45| AM-101, TG-001| AM-101, TG-001| AM-101, TG-001|              |              |
| 11:45-12:40|              |              |              |              |              |
| 12:40-01:35| AM-101, TG-002|              |              |              |              |
| 01:35-02:30|              |              |              | AM-101, TG-002|              |
| 02:30-03:25| AM-101, TG-001|              |              |              |              |
| 03:25-04:20|              |              |              |              |              |
Table 2. Timetable of a particular DAY

| Timing       | TG-001  | TG-002  | TG-003  | TG-004  | TG-005  |
|--------------|---------|---------|---------|---------|---------|
| 09:00-09:55  | EG-102,GP-04 |         | EG-102,GP-05 |         | PH-101,GP-01 |
| 09:55-10:50  |         | AM-101,GP-1 | PH-101,GP-03 | EG-102,GP-05 |         |
| 10:50-11:45  | AM-101,GP-05 |         |         |         |         |
| 11:45-12:40  |         | PH-101,GP-05 |         |         |         |
| 12:40-01:35  |         | AM-101, GP-1 | AM-103,GP-1 | PH-101,GP-01 |         |
| 01:35-02:30  | AM-101, GP-1 | AM-103,GP-07 |         |         |         |
| 02:30-03:25  |         |         |         |         |         |
| 03:25-04:20  | EG-102,GP-02 | AM-103,GP-07 | EG-102,GP-05 |         |         |

Table 3. Timetable of a particular class/group

| Timing       | Monday | Tuesday | Wednesday | Thursday | Friday |
|--------------|--------|---------|-----------|----------|--------|
| 09:00-09:55  | BE-105,TG-005 |       | SE-105,TG-005 |       | SE-105,TG-005 |
| 09:55-10:50  |        | BE-105,TG-005 | AM-121,TG-005 | AM-121, TG-005 |       |
| 10:50-11:45  | AM-121,TG-005 | SE-105,TG-005 | AM-121, TG-005 |       |       |
| 11:45-12:40  |        | BE-105,TG-005 |        | SE-105,TG-004 |       |
| 12:40-01:35  |        |        |         |          |       |
| 01:35-02:30  |        |        |         |          |       |
| 02:30-03:25  |        |        |         |          |       |
| 03:25-04:20  |        |        |         |          |       |

Figure 4. GA performance
4. **Conclusion:** Genetic algorithm used to generate a timetable is an evolutionary algorithm, and it is very much useful for this kind of NP-hard problem of timetable generation. By using this technique, much time can be saved as it is much more convenient than conventional methods. It prevents the occurrence of errors in formulating timetables manually. It is a very timesaving concept. Here in this paper Two evolutionary algorithms are compared in respect of Timetabling problems.
Table 1-3, depicted the output of our program (generated in Python), showing the schedule of a teacher, on a particular day and of a specific class respectively.

In figure 4, we can see the performance of our algorithm as best result can be achieved just after 25 iterations. In figure 5 and 6, it can be seen easily that GA and MA are both capable of approaching the level where no collision occurred. However, in case of MA, when there is no clash, the evolution happened very slowly, which can most likely be due to a local limit, as a result of which it has to wait for a suitable mutation. While in a GA, after we achieved a no-clash timetable, the evolution continued at the same rate, and we are always capable of obtaining a more acceptable mutation.

References
[1] Gottlieb, C. C. (1963). The construction of class-teacher timetables. In IFIP congress (Vol. 62, pp. 73-77).
[2] Dimopoulou, P.M.M. (2001) Implementation of a University Course and Examination Time- tabling System. European Journal of Operational Research, 130, 202-213. https://doi.org/10.1016/S0377-2217(00)00052-7.
[3] Ross, P., Corne, D. and Hart, E. (2003) Genetic Algorithms and Timetabling. Springer Berlin Heidelberg, 755-771.
[4] Schaerf, A. (1999) A Survey of Automated Timetabling. Artificial Intelligence, 13, 87-127.
[5] Muller, T. and Bartak, R. (2002) Interactive Timetabling: Concepts, Techniques and Practical Results. Master Thesis, KTIML MFF UK, Prague, September 2001, 58-72.
[6] Mooney, E.L. and Rardin, R.L. (1993) Tabu Search for a Class of Scheduling Problems. Annals of Operations Research, 41, 253-278. https://doi.org/10.1007/BF02023077.
[7] Welsh, D.J.A. and Powell, M.B. (1967) An Upper Bound for the Chromatic Number of a Graph and Its Application to Timetabling Problems. The Computer Journal, 10, 85-86. https://doi.org/10.1093/comjnl/10.1.85.
[8] Ayob, M., Malik, A.M.A. and Kendall, G. (2007) Solving a Practical Examination Timetabling Problem: A Case Study. Series Volume: 4707. Springer Berlin Heidelberg, 611-624.
[9] Carter, M.W., Laporte, G. and Lee, S.Y. (1996) Examination Timetabling: Algorithmic Strategies and Applications. The Journal of the Operational Research Society, 47, 373-383. https://doi.org/10.1057/jors.1996.37.
[10] Cooper, T.B. and Kingston, J.H. (1993) The Solution of Real Instances of the Timetabling Problem. The Computer Journal, 36, 645-653. https://doi.org/10.1093/comjnl/36.7.645.
[11] Kendall, G. and Hussian, N.M. (2005) A Tabu Search Hyper-Heuristic Approach to the Examination Timetabling Problem at the Mara University of Technology. Series Volume: 3616, Springer Berlin Heidelberg, 270-293.
[12] Theppakorn, T., & Pongcharoen, P. (2020). Performance improvement strategies on Cuckoo Search algorithms for solving the university course timetabling problem. Expert Systems with Applications, 161, 113732.
[13] Tayarani-N, M. H. (2020). Novel operators for quantum evolutionary algorithm in solving timetabling problem. Evolutionary Intelligence, 1-25.
[14] Kenekayoro, P. (2020). Incorporating Machine Learning to Evaluate Solutions to the University Course Timetabling Problem. arXiv preprint arXiv:2010.00826.
[15] Tavakoli, M. M., Shirouyehzad, H., Lotfi, F. H., & Najafi, S. E. (2020). Proposing a novel heuristic algorithm for university course timetabling problem with the quality of courses rendered approach; a case study. Alexandria Engineering Journal, 59(5), 3355-3367.