**Time Table Management System****using Genetic Algorithm**

Ashwini P Devadiga, Chaitra R K, Kripa, Manasa S, Dhanya Shenoy

Shri Madhwa Vadiraja Institute of Technology & Management, Bantakal-574115, Udupi

**ABSTRACT - Timetabling is the assigning of an event to a particular time slot in a timetable. Timetable construction is a hardworking and complicated task when there are a large number of courses and limited resources. As a result, some institutes tend to solve this issue manually even when the results may not always be fully optimal. Many solutions exist in the search space of a timetabling problem, but few of them are not feasible. Genetic Algorithm is a meta-heuristic algorithm that has been successfully applied to many optimization problems such as scheduling and timetabling problems. By using Genetic algorithm, we are able to reduce the time required to generate timetable which is more accurate, precise and free of human errors. Here the genetic algorithm is applied in the development of a viable timetabling system in which timetable is generated based on user specified constraint and requirements.**

**INDEX TERMS*:*** *Genetic Algorithm,**Timetable, Constraints*

**I. INTRODUCTION**

A timetable is a kind of schedule that set out times at which specific events are intended to occur. When specific requirements need to be followed, assigning task becomes hard to implement in timetabling problem. Timetable scheduling is a very important and tedious job in every educational institution. Nowadays colleges offer many courses and each course has many subjects to be taught. The faculty has to teach for more than one subject, this creates a difficulty in generating timetable and increases the paperwork which in turn consumes more time.The basic idea of timetabling problem is the assignment of educational events (lectures, laboratories, tutorials) in the given set of time slot which can reduce the overhead and increase the chance of error. The course scheduling system proposed to produce timetables that truly satisfy the needs of the user and increase teacher satisfaction. A typical way to solve a scheduling problem is to manually arrange events to be held into timeslots in a timetable [1]. The toughest thing about timetable is the endless list of paperwork. With timetabling software, it is possible to automatically generate timetables effectively. The timetabling problem is also called Constraint Satisfaction problem in which we obtain a solution that satisfies the given requirements. Requirements under consideration may be divided into two groups; i.e., hard constraints and soft constraints. Hard constraints are those that must satisfy, that is if the hard constraints are violated the timetable is no longer valid. Soft constraints are those, if we violate them in scheduling output is still valid. Each

institution has their own set of constraints. Accuracy and efficiency of scheduling system depends on information from the input of the timetabling system. Timetable scheduling problem in colleges is different from the timetable scheduling at school level. At the college level, a student may not get same number of courses each day. For example, they can have 3 subjects on Monday, but there may be only 2 subjects on Tuesday, and there is no class on Wednesday. Since every college has its own timetabling problem, the available software packages may not generate the best time table that satisfies all the constraints [2]. Hence an approach is needed to develop a timetable scheduling system which can be customized to fit any college timetabling problem.

There exist a lot of timetable problems such as:

* University timetable
* Railway timetable
* Exam timetable
* Sports timetable

There are many solutions in the search space of a timetabling problem but few of them are not feasible. Feasible solutions are those that do not violate the hard constraints and try to satisfy the soft constraints. Therefore, effectively modelling of practical problems close to the real-world situation is important. Many studies have been conducted in comparing the high performance and efficiency of the timetable.

# II. LITERATURE REVIEW

Solutions to timetabling problems have been proposed since 1980s. Research in this area is still active, which indicates that there are many problems in timetabling that needs to be solved. The problem was first studied by Gotlieb (1962), who formulated a class-teacher timetabling problem by considering that each lecture contained one group of students, one teacher and any number of times which could be chosen freely. Since then the problem is being studied using different methods under different conditions.

The paper [3] discusses about finding a feasible lecture/tutorial timetable in educational establishments. This paper presents evolutionary algorithm (EA) based approach to solving heavily constrained university timetabling problem. The approach uses a problem-specific chromosome representation. Heuristic and context-based reasoning have been used for obtaining feasible timetable in a reasonable computing time. An intelligent adaptive mutation scheme has been employed for speeding up the convergence.

The main approaches used in the paper [4] are Local search procedures and Constraint programming. Here the search panel has been used, which will give the response to the user queries and generates the error messages wherever necessary. JavaScript and MySQL database has been used for the implementation of the timetable. However, there are certain limitations where the user has to format the system after it is prepared, which can be solved using the logical approach. And there are certain other limitations that can be solved using Evolutionary algorithms.

The proposed system [5] uses the genetic algorithm to solve the timetable problem. This paper compares the two versions of genetic algorithm with and without local search, both to handmade timetable and other approaches based on the simulated annealing and Tabu search. Some difficulties are encountered when applying Genetic algorithms to constrained combinatorial optimization problem. The most relevant of them is crossover and mutation operators may generate infeasible solutions. However by changing the representation of the solution, crossover can be applied consistently. This approach is useful generalization of the Genetic algorithm and can be applied to other highly constrained combinatorial problems.

Create a population of creatures.

Evaluate the fitness of each creature.

While the population is not fit enough:

{

Kill all relatively unfit creatures.

While population size< max;

{

Select two population members. Combine their genetic material to create a new creature.

Cause a few random mutations on the new creature.

Evaluate the new creature and place it in the population.

}}

# Figure 1: Genetic Algorithm

The paper [6] focuses on particle swarm optimization (PSO) for finding optimal solutions to the problem of course timetabling. It is a technique based on particles, each particle has velocity in search space in order to find feasible solution and adjust its position according its own previous experience and neighbours’ experiences. It processes the search system using populations of particles, which corresponds to the use of individuals in genetic algorithms. In this paper the soft limitations which were not taken into account, a fact which is one of the disadvantages of their proposed method.

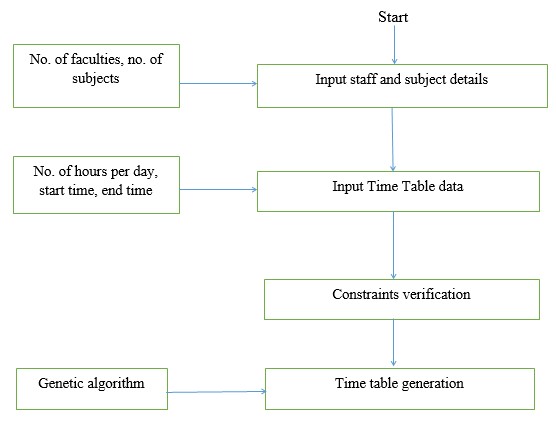
In [7] Simulated annealing approach is proposed to solve time tabling problem. Constraint satisfaction methods are used for an initial timetable satisfying all the hard constraints and some of the soft constraints. Subsequently, for later steps simulated annealing was used in order to optimize a given objective function. Simulated annealing is an iterative method that accepts a new solution if its cost is lower than the cost of current solution in each iteration. However, simulated annealing algorithm is less efficient in terms of the runtime because it needs more time to discover and evaluate the neighbours in different periods of the program.

The paper [8] focusing on the applications of the ant colony optimization to highly constrained realworld instance of the College course timetabling problem. The approach uses a MAX-MIN Ant System, since such system shown great promise on

various different problems. A colony of ants is used. At each algorithm iteration, each ant constructs a complete timetable. In each of the generated solutions, all of the hard constraints are satisfied. The main issue in applying ant colony optimization to problem is to find an appropriate representation that can be used by the artificial ants to build solution and theoretically cannot predict the best solution it should be experimentally done.

The paper [9] proposed a technique for smart autogenerate scheduling specifically for the educational sector. This uses Expert system by considering its abilities to imitate, extract and integrate between knowledge and human expert with data. This uses visual basics as a main coding language in developing auto-generate scheduling. At the final process it generates a report which checks the effectiveness of the system. If there is conflict between classes, missing lecturer and tutorial slot, a report on group, lecturer and classroom will be display for manual slotting. Data integrity is main drawback of expert system since system need to be updated manually.

**III. SYSTEM ARCHITECTURE & DESIGN**

The staff and subject details collected using web pages i.e., an input to the system. On receiving details of faculties, subjects and timetable data start allotting subjects. Final time table is generated by applying genetic algorithm. The proposed methodology consists of four steps: Input staff and subject details, Input Timetable data, Constraint verification, Time table generation using genetic algorithm.

**Figure 2: Methodology of Timetable system**

1. *Input Faculty and Subject Details:*

Input data is collected by the admin which include faculty and subject details respectively. This data is collected by two separate sub modules named :Add faculty and Add subject in our system.

**Add Faculty:** In this sub module we insert details of faculty such as faculty name and email. And we also provide a unique faculty id which helps in referencing throughout our system and it is also acts as a login credentials.

**Add Subject:** In this sub module we insert details of subjects that are in our curriculum such as subject code, subject name for the respective semester and section. We try to store the theory and elective subjects separately in our database so that it becomes easy for us in future.

The details collected by the admin are stored in database and used by the algorithm to generate efficient timetable

*2.Input Timetable Data:*

This step will take various inputs like number of subjects, faculties, semester, section and on relying on this input it will generate possible timetable for working days of a week. In our system admin has the access to generate the time table. The input to generate a timetable is collected by module named generate timetable. In this module admin can insert the faculty name, subject code, subject name, semester and sem. This data is obtained from database of faculty and subject details which is added by the admin. Here the specific details are obtained like particular faculty handling a subject, different subjects for each semester, number of hours per week for a subject.

*3.Constraint Verification:*

Constraints ensure that every aspect is arranged in a timetable for that timetable to be correct. Some constraints are verified for this system such as

* + Same faculty cannot be allotted to both sections for the same subject.
  + For a Faculty simultaneous classes are not allotted.
  + Not more than two classes are allotted for a subject per day.
  + For the same subject code different subjects cannot be mapped or added.
  + Faculty ID is unique to a faculty. Same Faculty ID cannot be allotted for different faculties.

*4.Timetable Generation:*

The last step is time table generation using genetic algorithm. Before we can use a genetic algorithm to solve a problem, a way must be found of encoding any potential solution to the problem. This could be as a string of real numbers or, as is more typically the case, a binary bit string. We will refer to this bit string from now on as the chromosome. A typical chromosome may look like this:

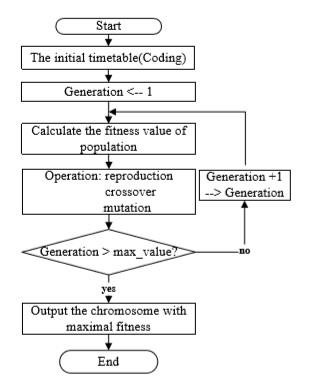
10010101110101001010011101101110111111101

At the beginning of a run of a genetic algorithm a large population of random chromosomes is created. Each one, when decoded will represent a different solution to the problem at hand. Let's say there are N chromosomes in the initial population. Then, the following steps are repeated until a solution is found

* + Test each chromosome to see how good it is at solving the problem at hand and assign a Fitness Score accordingly. The fitness score is a measure of how good that chromosome is at solving the problem to hand.
  + Select two members from the current population. The chance of being selected is proportional to the chromosome’s fitness. Roulette Wheel Selection is a commonly used method.
  + Dependent on the Crossover rate crossover the bits from each chosen chromosome at a randomly chosen point
  + Step through the chosen chromosomes bits and flip dependent on the Mutation rate.
  + Repeat step 2, 3, 4 until a new population of N members has been created.
  + Keep repeating until required fitness is achieved.

**Genetic Algorithm**

* + First of all, an initial generation of chromosomes is created randomly and their fitness value is analyzed.
  + New Generations are created after this. For each generation, it performs following basic operations:
  1. First of all, preserve few fittest chromosomes from the previous generation as it is. This is called Elitism and is necessary to preserve desired characteristics in the coming generations.
  2. Randomly select a pair of chromosomes from the previous generation. Roulette wheel selection method has been used here in this project.
  3. Perform crossover depending on the crossover rate which is pretty high usually. Here single point crossover has been used.
  4. Perform mutation on the more fit chromosome so obtained depending on the mutation rate which is kept pretty small usually.
* Now analyze the fitness of the new generation of chromosomes and order them according to fitness values.
* Repeat creating new generations unless chromosomes of desired fitness value i.e. fitness=1, are obtained.



**Figure 3: Flowchart of Genetic algorithm [10]**

**Objects of Time Table Schedular**

*Students Group*

The StudentGroup class has the ID, name of the student group, number of subjects, array of subject names and hours of study required for each subject per week. It also contains the id of teachers who will teach those subjects.

*Teacher*

It is a class to hold the faculty information. It has an id, name of the faculty, subject that he/she teaches and an in which represents the no. of batches assigned to the teacher.

*Slot*

A slot here is the most basic unit of Genetic algorithm. It represents a single characteristic of a Gene.

*Gene*

It is the main constituent of a Chromosome and is made up of a sequence of slot numbers. It represents a Time table of a single class group.

*Time Table*

This class’ object holds an array of Slot. This is basically a class to generate new slots initially for each Student group.

*SchedulerMain*

This is the main class of the algorithm which invokes other classes and calls methods for crossover, mutation, selection etc.

*Chromosome*

A chromosome here is a collection or an array of Genes. It is the main class of algorithm and it undergoes crossover and mutation to furnish fitter individuals.

*Utility*

It is basically for testing purpose only. It contains some methods like printSlots() which help to keep track of algorithm through console.

**Terminologies Used**

*Permutation Encoding*

In permutation encoding, every chromosome is a string of numbers, which represents number in a sequence. Other encoding techniques are Binary encoding Value encoding tree Encoding.

*Elitism*

A practical variant of the general process of constructing a new population is to allow the best organism(s) from the current generation to carry over to the next, unaltered.

*Roulette Wheel Selection*

It is a selection procedure in which the possibility of selection of a chromosome is directly proportional to its fitness.

*Single Point Crossover*

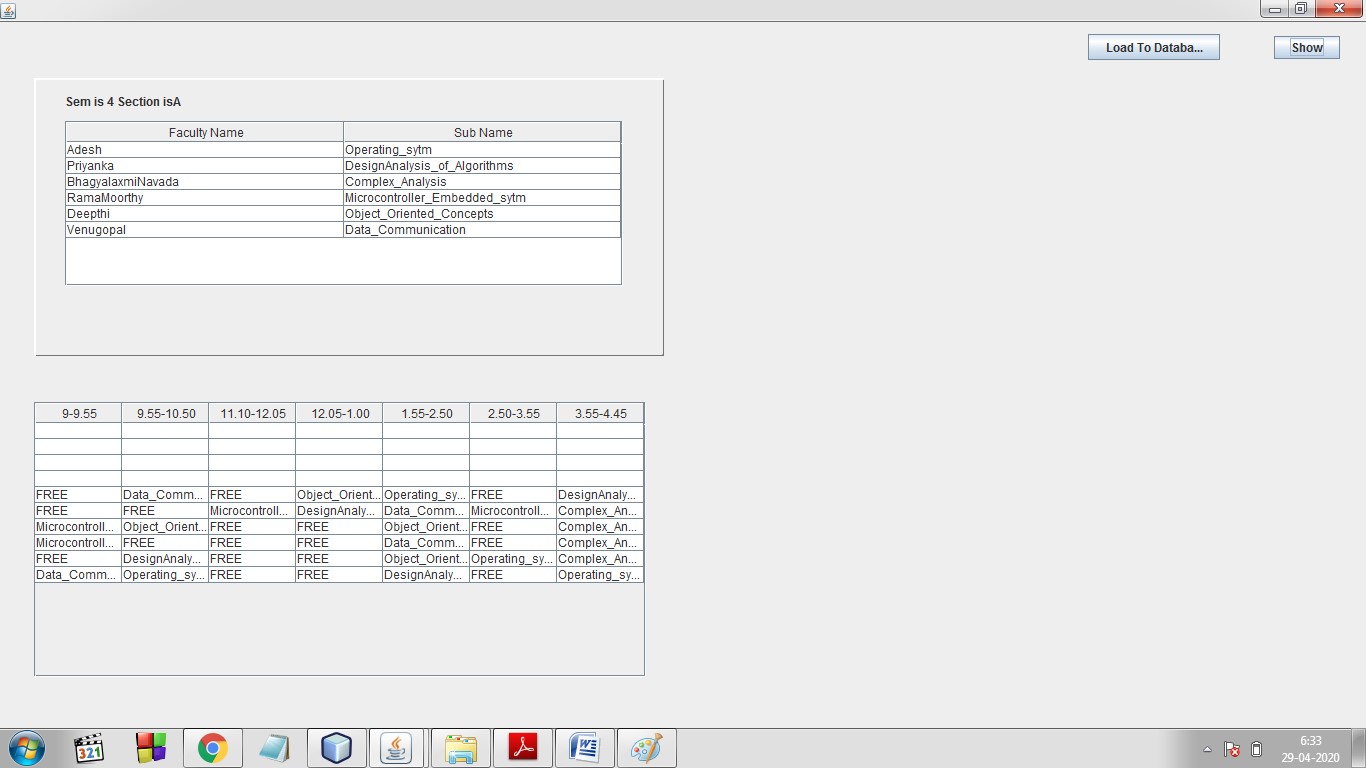
It is that type of crossover between two chromosomes in which the chromosomes are broken at a single point and then crossed.

*Swap Mutation*

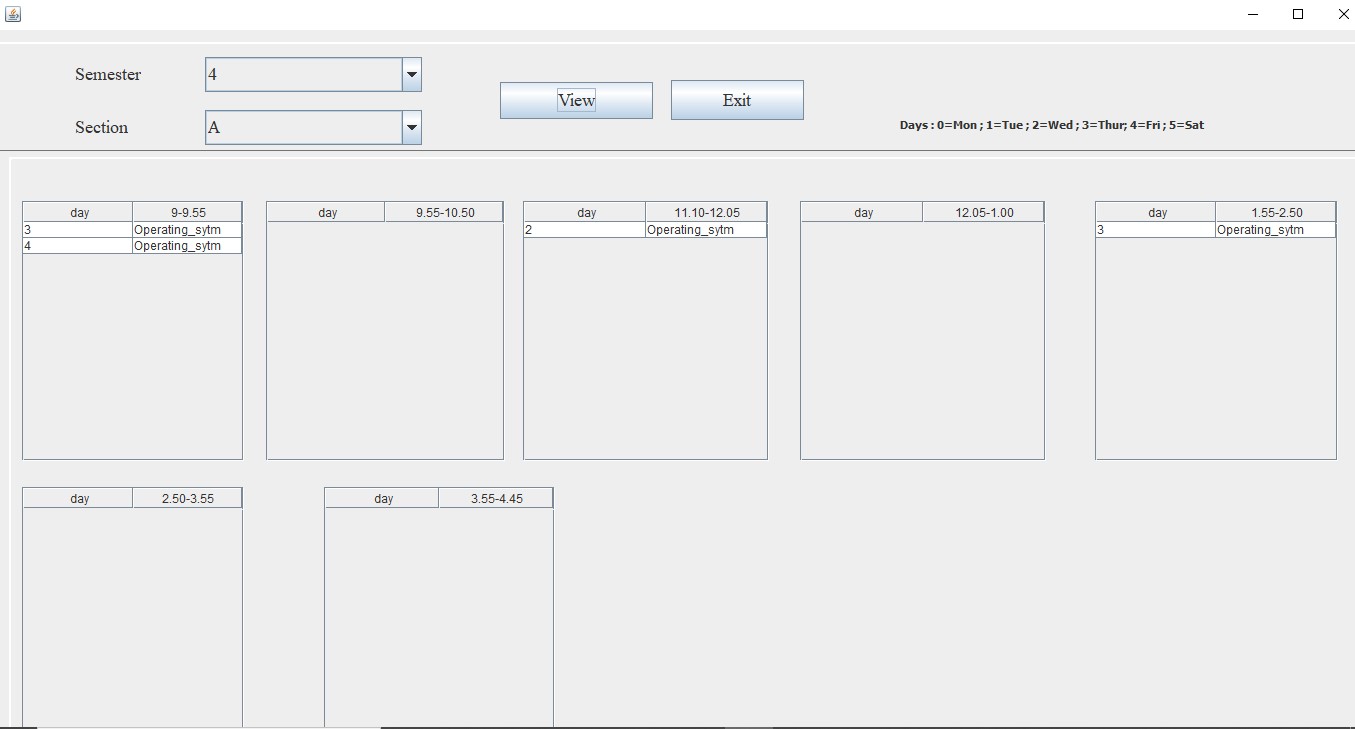
It is the type of mutation technique in which the chromosomes are so mutated that two portions of the chromosome get exchanged resulting in a new chromosome.

# IV. RESULTS

We have created several pages for taking inputs such as subject details, faculty details and timetable data. There are two modules: admin and faculty module. Admin can collect timetable data and generate timetable. This data is stored in the backend database. Genetic algorithm is used to generate the final timetable. The timetable includes both class timetable and personal timetable for the faculty as shown in Figure 4 and Figure 5. Figure4 shows the class timetable which can be viewed by both admin and faculty. Figure 5 shows a page where faculty can view his personal timetable by selecting the semester and section.



# Figure 4: Class timetable



**Figure 5: Personal timetable for the faculty**

# V. CONCLUSION

The process of Time Table generation has been fully automated with this software. Using Genetics Algorithm, a number of trade-off solutions, in terms of multiple objectives of the problem, could be obtained very easily. Moreover, each of the obtained solutions has been found much better than a manually prepared solution which is in use.

Though this web-app serves as a basic time table generator, there is a lot more which could be done to make this project even better in terms of consideration of soft constraints like professor giving preference to particular class. The future upgradations can be classroom size considerations, lab facility available and multiple subject selection for faculty. It can also have some add-on features such as Print option for individual faculty.

**References**

1. Shengxiang Yang, Sadaf Naseem Jat, “Genetic Algorithms With Guided and Local Search Strategies for University Course Timetabling”, Vol. 41, No. 1, January 2011 IEEE.
2. Xiao Fei Yang, Masri Ayob, Mohd Zakree Ahmad Nazri, “An Investigation of Timetable Satisfaction Factors For A Practical University Course Timetabling Problem”, 2017 ICEEI.
3. Dipti Srinivasan, Tian Houseow, Jian Xin Xu “Automated Time Table Generation Using Multiple Context Reasoning for University Models”,2002 IEEE.
4. V. Abhinaya, K. Sahithi, K. Akaanksha “Online Application of Automatic Time-Table Generator”, vol. 06, Issue 02, Feb 2019.

1. Alberto Colorni, Marco Dorigo, Vittorio Maniezzo “Genetic Algorithm to Solve the Timetable problem”,1994
2. Andeep Kumar, Kawaljeet Singh, Neeraj Sharma “Automated Time Table Generator Using Particle Swarm Optimization”, vol. 1, Issue 9, 2013 September.
3. E. Aycan and T. Ayav “Solving the Course Scheduling Problem Using Simulated Annealing”, 2019 IEEE.
4. Priyanka Gore, Poonam Sonawane, Suneha P Potdar “Time Table Generation Using Ant Colony Optimization Algorithm”, vol.5, Issue 3, March 2017.
5. Nur Iqtiyani Ilham, E.H.Mat Saat, N.H.Abdul Rahman, Farah Yasmin Abdul Rahman, Nurhani Kasuan “Auto-Generate Scheduling System Based on Expert System”, 2017 IEEE conference.
6. Ping Guo, Jing-xin Chen, Lin-Zhu “The Design and Implementation of timetable system based on genetic algorithm”, 2011 IEEE Conference.