**Time table generation using Genetic Algorithm**

**Abstract**

In today’s world there are many school, institutes, colleges and many struggle with keeping their schedule upright. These struggle includes overlapping of teachers, changing the timetable on teacher resignation and addition, and high randomness in their timetable. To deal with these struggles, techniques and AI / ML models are proposed by the researchers. In this study, the work done tries to implement a ML model that can deal with these struggles. Focus is on creating best possible timetable in a reasonable time interval. Time table generation is a NP hard problem that is nor its solution or verifying a solution happens in polynomial time. Work in the research has used the Genetic Algorithm, as it offers reasonable solution to NP hard problems. Genetic Algorithm is widely used and is popular for solving such NP hard problems.

**Introduction**

Genetic Algorithm is search, select and optimize algorithm based on principles of natural evolution. Let understand by real life example, Earth wants the most intelligent, survival expert, excellent adaptor species, so it randomly created the basic organism which had potential to turn into billions of species down the generations. Those organism reproduced choose different form and in end created billion individuals, but in process whose “fitness” (closeness to ideal species) was low, got terminated as individuals or species. So in end only those with great fitness gets more chances to survive, reproduce and grow in number, and now you look human beings are the most widely spread, and easy to find species on earth. So we say human being is the most fit or most closest to ideal species/solution.

GA was introduced first by J.H. Holland in 1992. GA includes chromosome representation, fitness selection, and biological-inspired operators.

**1. Chromosomes:** Generally chromosomes takes binary format but not neccessarily, as it can have your typically data like a timetable data structure containing subject name, teacher name, timing and classroom. In chromosomes, for this research we would be using timetable data structure using subject name and teacher name, because of constraint to have subject taught by teacher expert in that subject only not random one and reducing the algorithm complexity. Chromosomes are a point of solution space. Collection of these chromosome will form timetable. These are processed using genetic function/operators by replacing its population again and again till desired fitness or iteration. The fitness function just maps a possible solution to a number that represents its fitness or survival rate, here in this paper more the fitness number better the chances of its survival. As shown in previous example its totally biological-inspired operators as selection, mutation, and crossover depicting two main thing that is “Survial of fittest while ensuring diversity” .

**2. In selection:** the chromosomes are selected on the basis of its fitness value and choosen as parent for further processing. In crossover operator, a random locus is chosen and it changes the subsequences between two parent’s chromosomes to create off-springs contributing to “Survial of fittest” rule.

**3. In mutation:** some data of the chromosome will be randomly changed on the basis of mutation probability. Therefore, these elements of GA are focused in this paper.

The main contribution of this paper are as follows:

* The general GA used to produce timetable.
* Various types of genetic operators are explored with their pros and cons.
* Comparing the result of different mutation and crossover techniques.
* How to avoid local maxima while implementing GA.
* Finding best generation count and population size for particular situation.

The main aim of this paper is as follows:

1. It presents the solution of timetable generation using GA.
2. Using various combination of mutation, crossover, population size, generation count for optimal solution in reasonable time frame.
3. Introducing all soft and hard constraint conditions to check the fitness of a possible solution.

The paper further is organized as follows:

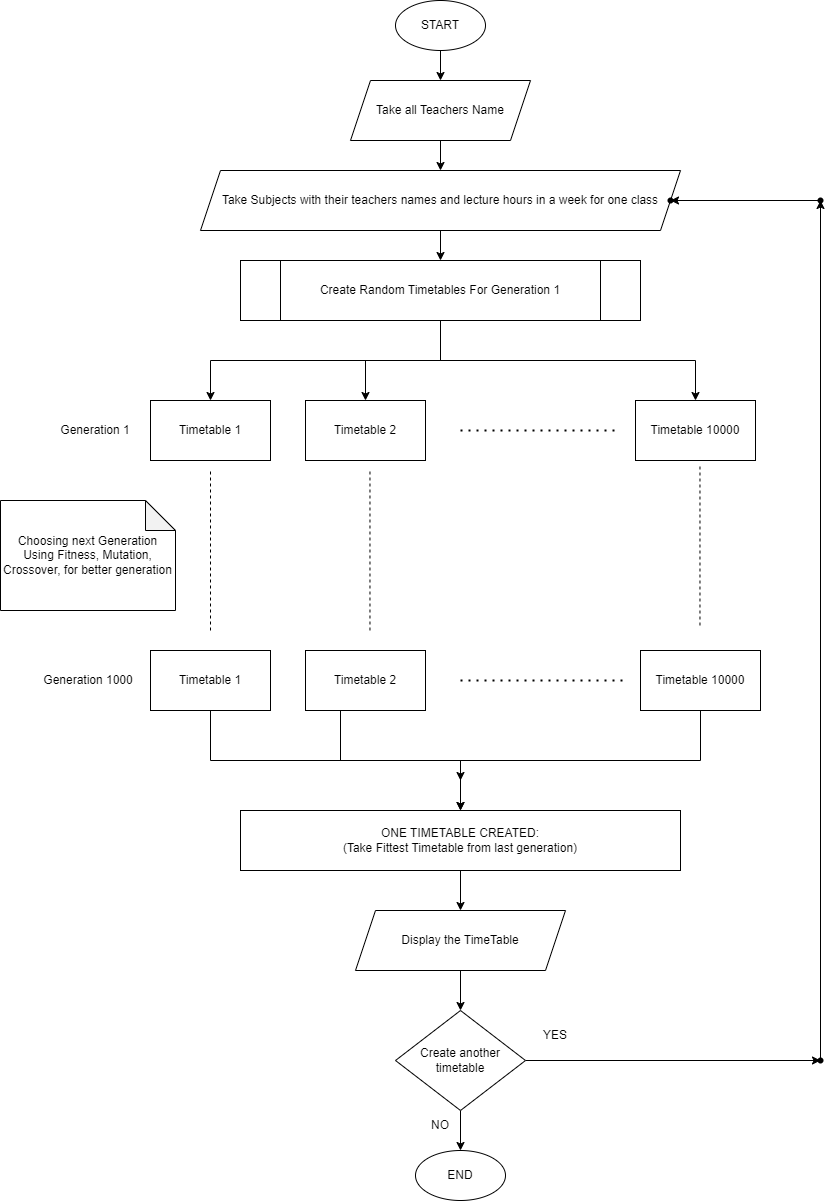
Methodology section for methodology used to do the reasearch. The genetic algorithm and genetic operators are discussed in Background section. Data analysis/experiment. Challenges and future possibilities. Then Discussion and conclusion section.

**Procedure**

1. **Initialization:** Commence the process by generating a diverse population of n chromosomes (timetables), each representing a unique random timetable.
2. **Fitness Evaluation:** Assess the fitness f(x) of each chromosome x within the population to quantify its effectiveness.
3. **Population Evolution:** Iterate through the following steps until the new population is assembled. a. **Parent Selection:** Utilize the tournament method to select two chromosomes, determining a winner as a parent. Repeat this selection process to obtain two parent chromosomes from the population, contributing to the creation of a child for the new population. b. **Crossover Operation:** Employ a crossover probability to recombine genetic material from the selected parents, generating new offspring (children). In the absence of a crossover, the offspring replicates an exact copy of a parent. c. **Mutation Process:** Apply a mutation probability to introduce random changes in the new offspring at each locus (position in the chromosome). d. **Population Integration:** Incorporate the newly generated offspring into the evolving population.
4. **Population Update:** Deploy the recently formed population for another iteration of the algorithm.
5. **Convergence Test:** Check if the termination condition is met; if so, conclude the process and return the best solution identified within the current population.
6. **Iterative Cycle:** Reiterate the algorithm by returning to step 2 for further refinement and enhancement.

**FLOWCHART**

**Created By:** Research Paper Author: Nitin Rana, Ayush Sharma

**Tool Used:**  https://app.diagrams.net/

**Genetic Operators**

Genetic algorithms search for better solutions through genetic operations, which include selection, crossover, and mutation operations.

**A. Selection Operation**

Selection operation involves selecting elitist individuals as parents in the current population, which can generate offspring. The fitness values are used as criteria to judge whether individuals are elitist. Two methods utilized are:

1. **Roulette Wheel Selection:** Chromosomes are selected based on their fitness, with better chromosomes having more chances of being selected.
2. **Rank Selection:** Chromosomes are sorted by fitness, assigned a rank, and selected based on their rank rather than fitness. This method ensures fairness, especially when fitness values differ significantly.
3. **Elitism Selection:** The best chromosome (or a few best chromosomes) is copied to the new population before applying crossover and mutation, preventing the loss of the best-found solution.
4. **Tournament Selection:** In tournament selection, pairs of chromosomes are randomly selected, and the one with the higher fitness is chosen as the winner. This process is repeated until the required number of parents is obtained. Tournament selection provides diversity and is particularly effective when the fitness landscape is rugged or noisy. In our case we will select two parents (two timetables) and create offspring from them.

**B. Crossover Operations**

Crossover operations determine the generation of successors by recombining and mutating selected members of the current population. The primary operator is the crossover, which produces two new offspring from two parent strings. Three types of crossover operators are:

1. **Single-Point Crossover:** Offspring are created by copying segments from one parent up to a randomly chosen point and the remainder from the other parent.
2. **Two-Point Crossover:** Intermediate segments of one parent are substituted into the middle of the second parent string.
3. **Uniform Crossover:** Bits are sampled uniformly from the two parents, creating a crossover mask with random bit strings.

**C. Mutation Operations**

Mutation operations produce offspring from a single parent by introducing small random changes to the bit string. A single bit at a random position is chosen, and its value is changed. Mutation is often performed after crossover.

**Fitness Function**

The fitness function in our timetable generation using a genetic algorithm plays a crucial role in evaluating the quality of candidate schedules. It consists of various components, each addressing specific aspects of the timetable's performance. The overall fitness value is determined by combining these components through weighted summation.

Lecture Hours Fitness: One key aspect of our fitness function ensures that the generated timetable adheres to the specified lecture hours for each subject. The lecHoursFitness component assesses the difference between the scheduled and required lecture hours for each subject. This component is weighted with a factor of 200.

Teacher Overlap Fitness: To prevent scheduling conflicts and ensure effective utilization of teaching resources, we incorporate the teacherOverlapFitness component. It penalizes schedules where teachers are assigned overlapping classes. The weight assigned to this component is 100.

Same Period Per Week Fitness: Maintaining a balanced distribution of subjects throughout the week is crucial. The samePeriodPWFitness component evaluates if a subject is consistently taught in the same period across different days of the week. This component contributes to the overall fitness with a weight of 70.

Same Subject Per Day Fitness: We aim to avoid scenarios where a subject is scheduled more than once in a single day. The sameSubjectPDFitness component penalizes such occurrences and carries a weight of 85.

Same Subject Same Period Per Week Fitness: Lastly, to encourage diversity in the timetable, we introduce the sameSubjectPeriodPWFitness component. It discourages the repetition of the same subject in the same period across consecutive days of the week. The weight assigned to this component is 10.

In the generation of each new population in our genetic algorithm, these fitness components are combined to produce an overall fitness value. This value guides the selection of candidate schedules for crossover and mutation, ultimately driving the evolutionary process towards optimal timetable solutions.

**Fitness Calculation**

One of main factors determining the efficiency of Genetic Algorithm result is how it calculate the fitness of a chromosome (in our case a sample timetable). So to find best fitness function, we tried various preference order of soft constraint of timetable

1. Lecture Hours (LH): This fitnest function account whether subjects’ lecture hours are met in week time or not. The focus is to ensure that the schedule effectively covers the required hours for each subject.

2. Teacher Overlap (TO): This fitness function examines the schedules to identify the overlapping of teaching classes for teachers.

3. Same Period Per Week (SPPW): This metric focuses on verifying if the same subject is scheduled during the same period across multiple days within a week. The aim is to have a regular pattern in timetable.

4. Same Subject Per Day (SSPD): This fitness parameter evaluates whether a subject is scheduled more than once in a single day. It promotes less frequency of same subject in a day.

To evaluate the preferences we should give to these constraints, we checked for every combination of priority for these 4 fitness function, giving us a total of 4! = 24 ways of assigning priority. But TO (Teacher Overlap) is hard constraint so it takes highest priority. So we are left with 3! = 6 total possiblities. We calculated total fitness and times taken for creating 10 timetables using these orders.

O1) TO > LH > SPPW > SSPD : 141105

O2) TO > LH > SSPD > SPPW : 179510

O3) TO > SPPW > LH > SSPD : 151090

O4) TO > SPPW > SSPD > LH : 148820

O5) TO > SSPD > LH > SPPW : 136370

06) TO > SSPD > SPPW > LH : 145590



Although not best but order 2 (ie. TO > LH > SSPD > SPPW : 179510) is better than rest, so we would use that order in our application.

**Conclusion**

In conclusion, the genetic algorithm-based timetable generation system presented here encapsulates an innovative and robust approach to solving the intricate problem of timetable creation. Through a meticulously crafted procedure, involving the initialization of diverse timetables, rigorous fitness evaluations, and the iterative evolution of populations via selection, crossover, and mutation operations, the algorithm demonstrates its adaptability and efficiency in finding optimal solutions.

The utilization of the tournament method for parent selection adds a layer of robustness, allowing for a fair and competitive environment among chromosomes. This, coupled with the integration of single-point crossover, aligns with the unique demands of timetable construction, ensuring the preservation of desirable genetic material in the offspring.

The algorithm's strength lies not only in its ability to navigate the vast solution space but also in addressing potential shortcomings, such as the loss of optimal solutions during population evolution. The incorporation of elitism selection, by copying the best chromosomes to the new population, serves as a strategic safeguard against losing valuable genetic material.

The inclusion of diverse selection methods, namely roulette wheel, rank, and elitism, adds flexibility to the algorithm, accommodating varying scenarios and complexities in the optimization landscape. These selection mechanisms contribute to the algorithm's adaptability, making it well-suited for scenarios where fitness values exhibit significant variations.

The crossover and mutation operations further enhance the exploration-exploitation balance, facilitating the creation of diverse and refined timetables. The algorithm's convergence is systematically tested, ensuring that the process halts when an optimal solution is identified, providing an efficient and effective timetable solution.

In essence, the amalgamation of these components establishes a formidable genetic algorithm tailored for timetable creation, showcasing its prowess in tackling the multifaceted challenges inherent in this domain. As educational institutions grapple with the complexities of scheduling, this algorithm stands as a valuable tool, offering a systematic and intelligent solution to the perennial challenge of timetable generation.

**References**

[1] Leon Bambrick, “Lecture Timetabling Using Genetic Algorithms” Available:http://secretgeek.net/content/bambrilg.pdf

[2] Burke, E.K. and Petrovic, S., 2002. Recent research directions in automated timetabling European Journal of Operational Research, 140(2), pp.266Y280.

[3] Chowdhary A., Kande, P., Dhone, S., Ingle, S., Rushiya, R. And Gawande, D.,2014 Timetable Generation system. International Journal of Computer Science and Mobile Computing, 3(2).

[4] Bhaduri, A., 2009, October. University timetable scheduling using genetic artificial immune network. In Advances in Recent Technologies in Communication and Computing, 2009. ARTCom'09. International Conference on (pp. 289Y292). IEEE.