International Journal for Research in Engineering Application & Management (IJREAM)
ISSN : 2454-9150 Vol-07, Issue-01, APR 2021

Automatic Timetable Generator using Genetic Algorithm

Suman S Mhalsank[1], Jenaan Kasmani[2], Sanket Wani[3] ,Sunil Wankhade[4]

[1,2,3,4]Department of Information Technology, MCT's Rajiv Gandhi Institute of Technology, Mumbai, Maharashtra, India

[1] sumanmhalsank2000@gmail.com, [2]jenaan.official@gmail.com , [3]wanisanket08@gmail.com, [4]sunil.wankhade@mctrgit.ac.in

Abstract - Timetable creation is a very tedious and time-consuming task. Years back when computers were not used to solve complex problems due to computational power, Scheduling classes was often done by humans which proved to be efficient but not perfect. Today with the explosion of Artificial intelligence and an exponential increase in computing power, using computers to solve new set of problems that previously could only be solved by humans became easy. Scheduling is one of the problems that can now be solved by computers with solutions acceptable or even better. To resolve the complexity and to reduce the efforts of generating timetables manually, some technologies can be implemented. However, scheduling is known to be a non-deterministic polynomial-time (NP) complete problem where, in order to find the best solution, every possible combination should be executed. Genetic Algorithm (search heuristic that is inspired by Charles Darwin's theory of natural evolution) is fit for scheduling problems as it creates a solution over time based on rules and criteria. This algorithm reflects the process of natural selection where the fittest individuals are selected for reproduction to produce offspring of the next generation using which we can optimize the Scheduling problem.

Keywords — Adaptive-Elitist Genetic Algorithm, Evolutionary Algorithm, Genetic Algorithm, Machine Learning, Python, Scheduling.

I. INTRODUCTION

Timetable scheduling is the way toward making schedules that fit the imperative of the situation. It is utilized in an extent of industry from booking transportations up to making complex timetables for profoundly enhanced robotized industrial facilities. Lion's share of limited scope scheduling are done physically while bigger activities require PC helped planning.

Artificial intelligence is one of the rising computing solutions due to an increase in computing power. It tends to be applied to various kinds of issues which can help optimize existing solutions or create never been attempted solutions because of multiple limitations. Artificial Intelligence helps to solve non-deterministic polynomial-time issues that organizations and businesses will consistently have. NP is the set of decision problems for which the problem instances, where the answer is "yes", have proofs verifiable in polynomial time by a deterministic Turing Machine.

Genetic algorithm is a metaheuristic that mimics the process of natural selection. It can be performed in multiple different ways with different types but it will all follow the same concept. This project aims to create an artificial intelligence through the use of evolutionary algorithm, specifically genetic algorithm combined with adaptive and elitist traits that can generate a university schedule timetable with the goal of generating a valid and as optimal as possible solution with certain constraints

II. LITERATURE SURVEY

NP-Complete Complexity Class

The knapsack problem belongs to the complexity class of non-deterministic polynomial-time (NP) hard. These are problems where there are no efficient solution algorithms yet. The computational time only grows as the size grows according to Stephen A. Cook [6]. According to standard defining organization's computer scientist, Paul E. Black [7], NP-Complete is a complexity class where solutions can be verified quickly and if there is a quick algorithm to solve this class of problem, it can be used to solve other NP problems too. In an article by J.D. Ullman [8], he said

 DOI : 10.35291/2454-9150.2021.0149

that finding the most optimal schedule or solving scheduling itself belongs to the NP-Complete problem.

Ways on Solving University Class Scheduling

There are multiple approaches towards solving scheduling problems according to Ellis Cohen and Jarurat Ousingsawat [9]. Some are only feasible up to a certain level such as the constraints they can handle, objectives to meet and computation time. The most common approaches are construction algorithms which greedy algorithm is included and search algorithms where tabu search, simulated annealing, particle swarm optimization, genetic algorithm and many more from J Zhou, P E D Love, X Wang, K L Teo and Z Irani's review on scheduling [10].

Genetic Algorithms: An Overview

According to Melanie Mitchell's lesson [11], genetic algorithm is a program that mimics the process of biological evolution in order to solve problems and to model evolutionary systems. The vast search space has inspired the creation of genetic algorithm. Biological entities over the years have adapted to their environment to be more efficient and effective. It has then become part of the early waves of artificial intelligence wherein rules and behavior are manually encoded. The algorithm is still in use in the modern period mostly for optimization problems.

III. PROPOSED SYSTEM

This paper aims to create an artificial intelligence that can create timetable schedules. This only covers processing input data and generating human sensible results. The process to be used in creating schedules is adaptive-elitist genetic algorithm. A type of evolutionary algorithm that involves keeping the best set of solutions over next generations to preserve high-level solutions but at the same time, variables such as the population count, mutation rate and selection pressure, change in order to avoid premature convergence that leads to poor output. Some Hard Constraint are Instructors teach one class at a time, Instructors can only take *N* amount of subjects to have less load, Sections attend only one class at a time. Some Medium Constraints are Sections' subjects are placed on the schedule, Sections should have at least 30 minutes vacant time for a lunch break. An example of Soft Constraint is Students should have only 30 minutes break for every two hours of session per day. More constraints can be added according to the user's choice

CONCEPTUAL FRAMEWORK

The program can easily be described as a scheduling computing software wherein you input basic data sets and it will output a structured result. In Fig 1: Simplified Model of the Program as we can see, The application takes five major input; Instructors, rooms and subjects component supports the addition of entry using the application and importation.

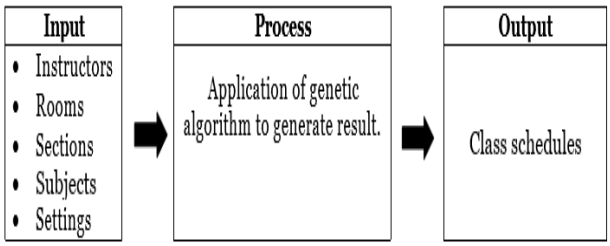

Fig 1: Simplified Model of the Program

The sections component allows the creation of sections with a special feature of sharing subject with other sections and the scenario manager component handles running configuration. There would be two types of output; using the result viewer of the application which can view the top five solutions of the last generated scenario and exporting the selected solution which will produce three comma-separated values (CSV) files. The application will be using the Python language.

FLOW OF THE WORK

The repetitive process of GA is best conceptualized using a flowchart which is Fig 2: Genetic Algorithm Flowchart .

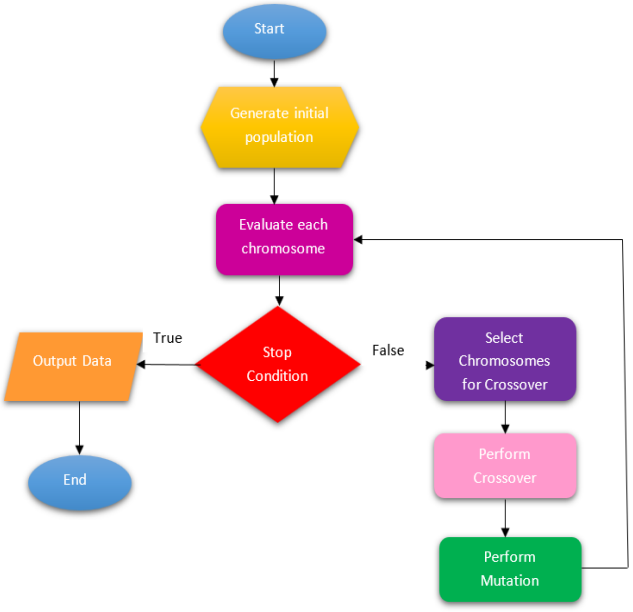

Fig 2: Genetic Algorithm Flowchart

Generation of the population is done by random selection with a mix of greedy approach (Random filling). During the generation of population, all hard constraints are to be met. Evaluation which is the process of calculating the Fitness of a Chromosome triggers the ending process given that a certain fitness is met. Calculating the fitness is done by assessing each chromosome to the selected constraints and the evaluation matrix (set of constraint weights based on

 DOI : 10.35291/2454-9150.2021.0149

prioritization of constraints using a distribution of a hundred percent (100%) provided by the user).

We included seven factors to be distributed in the 100% of the Evaluation Matrix they are the subject placement, lunch break, section rest, section idle time, instructor rest, instructor load balance and meeting pattern.

The total fitness can be calculated using the formula below where:

y = chromosome

f = functions that follow the evaluation matrix

fW = Evaluation weight

$$Fitness(y) = \sum (f(y) * fW)$$

$$Fitness(y) = (a(y) * aW) + (b(y) * bW) + (c(y) * cW) + (d(y) * dW) + (e(y) * eW) + (f(y) * fW) + (g(y) * gW)$$

The Next step is Adaption (changing running variables to cater for better results) of artificial intelligence to its current performance to avoid focusing on one set of the problem which may cause pre-mature convergence. Adaption performs two important tasks which are population alignment and mutation rate adjustment.

POPULATION ALIGNMENT

Population (A set of proposed solutions). We can change the running variables before the generation of the solution. The GA settings "Minimum Population" and "Maximum Population" define the limit of population change. Calculation of population alignment will show how much population change is to be done using a series of formulas.

MUTATION RATE ADJUSTMENT

The mutation rate is the chance for each chromosome to get random change. When a generation completes without triggering the adjustment, the mutation rate will decrement by 0.5% else it will be increased by 0.5%. Calculation of mutation rate adjustment trigger follows a formula

After evaluation, the fittest chromosomes are picked to participate in reproduction. Considering the pros and cons, out of all the ways of selecting chromosome, Elitism is used here which is the most appropriate type of selection will help avoid early convergence and promotes diverse solutions. As an elitist variant of the genetic algorithm, the top *n%* of the population is guaranteed to proceed in the next generation with the same genes depending on the settings. The rest of the selection process is to be done by implementing multiple tournaments on the population. The participants in the tournament are selected randomly with a quantity of 4% of the existing population but capped at twenty-five (25). The tournament will run until enough pairs are picked. The mating pool consists of selected chromosomes that will undergo crossover (Production of offspring based on genes). Like selection, to have a better outcome, Here out of all the ways Davis' order crossover works better for permutation-based problems. Mutation (altering the gene/s of a chromosome) to maintain the variety of solutions. Mutation is the process of. Usually, the mutation rate is low and fixed. However, adaptive genetic algorithm means that the mutation rate may vary depending on the performance of the population.

Once the genetic algorithm terminates either by force or meeting the end criteria, the top 5 fittest chromosomes will be displayed for user review. The user may pick any from the proposed solution and get its output in a CSV file.

GENETIC ALGORITHM SETTINGS

Due to the limitation of computational power, the configuration of genetic algorithm within the application is not one for all types where it can cater to every scenario.

Name	Limit
Minimum Population Count	100-200
Maximum Population Count	Minimum Population Count-200
Maximum Generations	50-150
Maximum Creation Attempts	1,500-4,500
Mutation Rate Adjustment Trigger	0.00-1.00 where value is divided by five (5)
Maximum Fitness	90-100
Elite Population	0-10%
Deviation Tolerance	50-75%

Fig 3: Genetic Algorithm Random Settings

To counter the limitation and still be able to support scalability of usage, the algorithm's running configuration can be altered according to operators' device capability or preference. The setting properties include in the Fig 3: Genetic Algorithm Random Settings includes random values of the settings that can be changed according to the user's choice to get the best optimal result.

IV. DESIGN AND METHODOLOGY

The paper falls under optimization problem and evaluation of performance in an extensive verification is not a feasible solution. It is important to select an appropriate design and methodology to get the viability of the work. The selected method for collecting and analyzing data (e.g. subjects, instructors and rooms) for testing the system is quantitative. These datasets will be profiled using pandas after being used in the system.

 DOI : 10.35291/2454-9150.2021.0149

Iterative development as a method for creating the software fits for continuously changing environment for artificial intelligence development. This enables the developer to learn every iteration and apply it to the future instance. This methodology is fit for the project as the system relies on tweaking and managing constraints. Initial planning has been done by designing system architecture, data schematic and models. Upon entering an iteration, the feature to be done is decided and planned. There are two types of iterations used; adding a feature or a model and tweaking values. Each iteration's goal is then implemented and then manually tested.

The selected evaluation methodology for the project is a combination of Monte Carlo methods and surrogate modeling. It is an experimental methodology as evaluation for evolutionary computation based scheduling systems are yet to be done. Implementation of Monte Carlo method and surrogate modelling for evaluation of artificial intelligence based scheduling system, follow the following steps:

(1) Generation of a set of solutions using Monte Carlo method with genetic algorithm settings as variables that can be tweaked.

(2) Creation of a surrogate model which only uses the evaluation matrix as basis.

(3) Computation of each set of solutions generated by the Monte Carlo method difference to surrogate model.

(4) Providing the mean of the result as basis for performance of the artificial intelligence (Scenario based).

Usage of the Monte Carlo method in the system is for the generation of random genetic algorithm configuration. The surrogate modeling usage for this project makes use of an imaginary 1-dimensional metric basing on the evaluation matrix. The model will serve as a basis for evaluating the closeness of the solution to the ideal one. There will be three models Balanced Distribution, Subject Placement and Tight Constraints For each model, there will be three scenarios. For each scenario, there will be three randomly generated value of settings It is important to note that there will only be one time generation of random values for settings and the three generated values will be used all throughout the scenarios to prevent bias in result. For every setting, there will be five generated result set. In total, there will be 135 solution set for evaluation which will be abstracted per level.

V. RESULTS

Judging by the performance of results from Fig 4: Timetable of a Week (1) and Fig 5: Timetable of a Week (2), the system was able to generate solutions that have at least 80% fitness (basing onset of highest solutions per scenario). The problem lies in the inability of artificial intelligence to perform minor corrections on adjusting schedules Nevertheless, the algorithm was able to cater to a majority of the entries on the evaluation matrix and perform to its capability despite limitations imposed by the algorithm's configuration. It is important to note that the generated random values for settings were limited, therefore, the system was not able to perform at its peak. The results of the application are not guaranteed to be the best possible solution for the scenario. The quality of the result relies heavily on running preference. Due to its stochastic nature, results may vary from poor to excellent. The system does not guarantee that every solution's hard constraint will be met especially when the presented scenario is logically impossible or intensely tight.

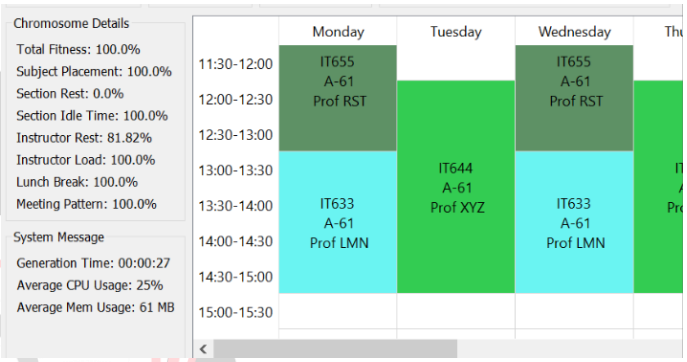

Fig 4: Timetable of a Week (1)

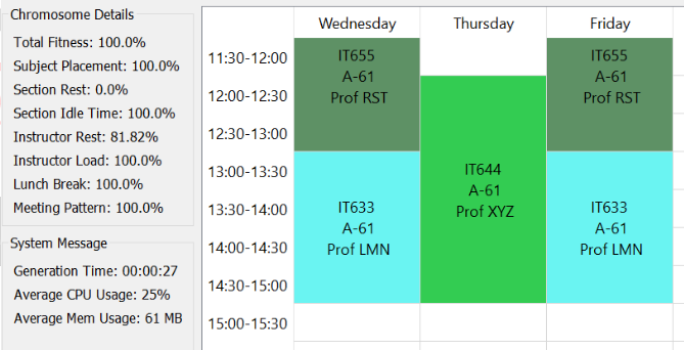

Fig 5: Timetable of a Week (2)

In the Fig 4: Timetable of a Week (1) i.e Monday to Wednesday and Fig 5: Timetable of a Week (2) i.e Thursday to Friday, we see the week's timetable is formed based on the inputs given and the constraints to be met. The Time Slots (Rows) and the week days (Columns) are displayed, Professors are allotted classes and timings, the subjects they teach are mentioned. The Blocks displayed include the Subject name, Classroom and the Professor name. The height of those blocks indicate the starting hour and the ending hour, the professor will teach. This is an example of one Chromosome shown and the details of the chromosomes that include it's fitness along with the seven factors we included in our Evaluation Matrix.

 DOI : 10.35291/2454-9150.2021.0149

Below that the application also shows the Generation Time required to generate the timetable, the CPU and the memory usages. Four more chromosomes are generated by the application, it's the user's choice to select the chromosome they think is perfect. Due to the stochastic nature of the algorithm, it will generate different chromosomes everytime with varying chromosome details.

VI. CONCLUSION

The outcomes have shown that this framework/system can provide valid solutions that can be utilized. Nonetheless, it does not give total automation. There are still situations that would require the administrator to change a few entries to make an ideal solution. The system was likewise intended to be basic and clear. This takes out any confusion brought about by dispersed user interface controls and makes use of software fully utilized. The straightforwardness and simplicity of the system and introduction of configurable algorithm's goal and performance reduced the requirement for such a lot of constraints as solutions are made dynamically. This enables users to effortlessly utilize and explore different avenues regarding the application until they discover the ideal fit for their scenario. A large number of combinations for testing to find an accurate evaluation for the application has proven to be far from possibility. However, it very well may be reasoned that from the models provided, the system had the option to create results that despite being imperfect, remains valid and acceptable given the number of constraints imposed on it. The solutions that the system will provide will intensely rely on the running configuration and evaluation matrix. One may track down an ideal solution if the application was given sufficient time and computing power. The complete evaluation for the system will remain hard to solve as the freedom for the configuration of the algorithm has provided a large number of combinations. It can likewise be deduced that evaluation using other methodologies will yield the same amount of result.

Acknowledgment

We would like to acknowledge and thank the following:

Our God Almighty, for making all of this happen.

Beloved Parents who always supported us in all their capabilities and served as our inspiration for success in life.

MCT's Rajiv Gandhi Institute of Technology's faculty who supported us whenever we needed assistance, which made the research project possible.

Our friends and peers who have shown their support and help in times of distress.

REFERENCES

[1] "The Usage of Operations Management", Universal Class. Retrieved from https://www.universalclass.com/articles/business/the-usage- of-operations-management.htm

[2] R. Dan Reid and Nada R. Sanders, "Operations Management: Chapter 15", 2010. Retrieved from www.csus.edu/indiv/b/blakeh/mgmt/documents/opm101chapter 15

[3] Paulyn Navarrete and Illiana Tan, "Persistent issues faced during enlistment", January 2015.Retrieved from http://thelasallian.com/2015/01/29/in-review-persistent-issues-faced-during-enlistment/

[4] Careen L. Malahay and Tweeny M. Malinao, "Elementary school shortens classes to accommodate pupils", June 2012. Retrieved from http://newsinfo.inquirer.net/207699/elementary-school-shortens- classes-to-accommodate-pupils

[5] Carsten Murawski and Peter Bossaerts, "How Humans Solve Complex Problems: The Case of the Knapsack Problem", October 2016. Retrieved from https://www.nature.com/articles/srep34851/

[6] Stephen A. Cook, "An Overview of Computational Complexity", June 1983. Retrieved from https://dl.acm.org/citation.cfm?doid=358141.358144

[7] Paul E. Black, "NP-Complete", December 2016. Retrieved from https://xlinux.nist.gov/dads/HTML/npcomplete.html

[8] J.D. Ullman, "NP-Complete-Scheduling-Problems", June 1975. Retrieved from https://www.sciencedirect.com/science/article/pii/S0022000075800

[9] Ellis Cohen and Jarurat Ousingsawat, "An Introduction to Algorithms for Solving Schedule-Related Problems", January 2015. Retrieved from

http://www.project.net/introduction-algorithms-solving-schedule-related-problems

[10] J J Zhou, P E D Love, X Wang, K L Teo and Z Irani, "A review of methods and algorithms for optimizing construction scheduling", March 2013. Retrieved from https://link.springer.com/article/10.1057/jors.2012.174

[11] Melanie Mitchell, "Genetic Algorithms: An Overview", 1995. Retrieved from http://ohm.ecce.admu.edu.ph/wiki/pub/Main/ResearchProjects/mit chell_GA_tutorial.pdf

 DOI : 10.35291/2454-9150.2021.0149