Timetable Generator Using Genetic Algorithm

Sahil Sarnaik[1], Tanmay More[2], Sanket Shinde[3], Dipesh Shah[4]

[1]Student, Dept. Of Computer Engineering, AISSMS College Of Engineering, Pune, Maharashtra, India
[2]Student, Dept. Of Computer Engineering, AISSMS College Of Engineering, Pune, Maharashtra, India
[3]Student, Dept. Of Computer Engineering, AISSMS College Of Engineering, Pune, Maharashtra, India
[4]Student, Dept. Of Computer Engineering, AISSMS College Of Engineering, Pune, Maharashtra, India

---***---

Abstract- *Timetable planning is a critical aspect of efficient resource management in organizations and businesses. While manual planning suffices for smaller-scale operations, larger-scale endeavors require computer-assisted planning to accurately accommodate various constraints. However, human error can still affect the overall efficiency of the organization. To overcome this challenge, recent trends have seen the use of software developed using genetic algorithms. The rapid advancement of computing technologies that has fueled the progress of artificial intelligence provides answers for the notoriously difficult organizational conundrums known as NP-hard problems, which arise frequently. As a sophisticated category of algorithms modeled after evolutionary mechanisms in nature that hone in on optimal solutions, genetic algorithms emulate the process of natural selection in a particularly ingenious fashion. Existing software applications typically utilize a one-hour timeframe and have several dependencies, such as XAMPP server or an internet connection, to run the program successfully. In contrast, our proposed software aims to introduce a novel approach by utilizing a more optimized half-hour timeframe. This enhancement addresses a significant limitation of existing solutions and enables finer-grained scheduling precision. By employing this improved temporal resolution, our software endeavors to provide more efficient and effective timetable generation. In this regard, this research seeks to leverage evolutionary algorithms, specifically genetic algorithms, combined with adaptive and elite traits to create an artificial intelligence system that can generate a university timetable. The aim is to create the correct and best solution, considering a set of rules, while minimizing human error and enhancing efficiency.*

Keywords- Artificial Intelligence, Genetic Algorithm, Timetable Generation, Evolutionary Algorithm, Optimization

I. INTRODUCTION

The process of scheduling involves creating a plan that meets specific constraints based on the given scenario. In the recent times, the scheduling of timetables has been the subject of extensive research in fields such as Operations Research and AI. Various enhancement methods employed address these problems, with the aim of generating solutions that are either optimal or near-optimal rather than exact. It is widely used in various industries such as transportation planning and complex scheduling for automated factories. While smaller jobs are typically scheduled manually, larger projects require computerized planning.

The increasing computing power has led to the rapid growth of Artificial Intelligence, which is becoming increasingly popular. It is capable of addressing a wide range of problems, including optimizing existing solutions and devising innovative ones that were previously impossible due to several limitations. Artificial Intelligence can offer valuable solutions to organizations and businesses by solving indeterminate polynomial time problems.

GA emulates natural selection as an effective method for optimization improvement. Although there are various types of genetic algorithms, they all follow the same basic principle. The primary aim of this research is developing an Artificial Intelligence system that leverages evolutionary algorithms, specifically a genetic algorithm to generate a university timetable that satisfies specific constraints while being both feasible and optimal. Genetic algorithms have found applications in various areas, including optimized searching and scheduling processes. This algorithm is typically used when optimization is the primary objective. In the case of a timetable problem, the challenge involves determining the precise timing allocation of a collection of activities (like academic sessions and lectures) within a limited time frame and assigning them to specific resources (like instructors, learners and rooms) adhering to certain guidelines.

This research paper employs a genetic algorithm to generate the timetable for Engineering Faculty. The algorithm features a dynamic chromosome size that can be easily adjusted to account for the course numbers in each department.

II. REVIEW OF LITERATURE

As per the findings of Asif Ansari [4], there are multiple approaches available for solving scheduling problems in the field of operational research. These include techniques such as graph coloring and mathematical programming, local search methods, constraint satisfaction manipulation through taboo searches and simulated cooling, as well as backtracking. Genetic algorithm, a type of optimization algorithm, is an evolutionary algorithm that builds upon traditional methods.

According to Immanuel and Chakraborty (2019), the genetic algorithm functions on the concept of "survival of the fittest," where successive generations produce enhanced solutions until an optimal solution is achieved [7].

To efficiently create a timetable for an engineering college, a genetic algorithm proves to be a valuable tool. In their study, G. Alnowaini and A. A. Aljoma [1] introduced a technique that incorporates dynamic chromosome size, specifically designed for this type of scheduling problem. The authors developed a system that allows for the input of various elements, including lecture halls, buildings, lecturers, levels and departments. Additionally, they implemented restrictions to govern the timetable creation process. The main objective of the authors' research was to address the timetabling challenges faced by universities each academic year and mitigate the significant costs associated with generating near-optimal timetables.

The genetic algorithm serves as a useful approach for obtaining an optimal solution to scheduling problems. In the reviewed papers, the system administrator begins by signing into the system and entering the courses along with their respective course codes and units [2]. The administrator can continue adding courses until the required number is reached, and they also have the ability to delete any incorrectly imported courses. Once the courses are entered, the administrator proceeds to input all the rooms to be utilized. Upon entering this information, the system generates the timetable.

To expedite convergence, an adaptive mutation scheme was used. The system design is compatible with various operating systems, starting from Windows Vista. However, for existing systems, the installation of the XAMPP server is necessary. Popular internet browsers such as Internet Explorer, Microsoft Edge, Google Chrome, Opera Mini, and UC Browser are supported. The development of the system utilized the following tools: Supernatural Test 3 and XAMPP for implementing existing software.

Although the genetic algorithm is widely recognized as a highly effective approach for generating schedules, it does not provide a guarantee of achieving a 100% optimal schedule. The degree of optimality is contingent upon the specific constraints applied and the adaptability of the parameters' adjustment function. Despite involving multiple steps and potentially slower execution, the utilization of the genetic algorithm is favored due to its efficiency in generating schedules.

Timetable schedulers created through the utilization of genetic algorithms may not be able to address all given constraints, but they are proficient in handling significant constraints, such as avoiding clashes between two teachers within the same time slot. In the study conducted by Shruthi. B (2020), the developed system generates timetables specifically with a duration of 1 hour, and it achieves an accuracy level of 80% [3].

A Genetic Algorithm (GA) problem involves candidate solutions that are not necessarily optimal but represent possible solutions. These solutions are composed of one or more individual traits known as chromosomes or genotypes, which undergo crossover or mutation operations to generate new solutions for the same problem [7]. Mutation is a divergent operation that aims to explore new regions by altering one or more members of the population, potentially leading to improved solutions outside of the local minimum or maximum space. In the crossover process, two parent solutions are combined to produce offspring for each case [4].

The crossover function plays a crucial role in genetic algorithms (GA). It involves pairing genes from each chromosome using a crossover operator to create new generations. These offspring are then judged based on their fitness scores to determine their qualification for the next generation. Orong, Sison, and Medina [8] proposed a new crossover mechanism known as cross-average crossover. They integrated a rank-based selection approach that enhances the effectiveness of variable minimization, leading to promising results.

Qiongbing and Lixin [8] conducted a separate study that introduced a distinct crossover mechanism known as Same Adjacency crossover in the context of genetic algorithms with variable-length chromosomes. Same Adjacency crossovers aim to find more efficient crossing pairs between parental chromosomes. However, this mechanism requires additional computational work, and the determination of intersection points relies on information from neighboring nodes.

A Java-based system was proposed by Shraddha Shinde, Saraswati Gurav, and Sneha Karme [6] to automate the process of generating timetables. In addition to timetable generation, the system provides additional features allowing users to request leave by specifying the leave date, reason, and alternative faculty. To implement this system, the installation of JSP and

MYSQL is required. The framework developed by the authors appears to have broad applicability to various other scheduling problems beyond timetabling.

According to Shraddha Thakare, Tejal Nikam, and Mamta Patil [5], the key modules in the scheduling procedure include:

1. Data encoding and decoding

2. Initial population

3. Evaluation of population

4. Crossover evolution

5. Mutation

6. New population

These modules are essential in the scheduling process and serve as fundamental steps to generate efficient schedules. The application of these modules extends to various areas and timetable generation of any organization.

After an extensive study of the papers on timetable generation and scheduling, it is apparent that the genetic algorithm plays a vital role in producing optimal schedules while considering all the constraints and rules. This algorithm not only saves time but also eliminates the complexity associated with manual timetable creation and management. Instead of relying on tedious paperwork, students and faculty can access the timetable quickly. By implementing a user-friendly interface and utilizing a well-designed database, the issue of manual timetable creation can be effectively resolved. This approach enhances the efficiency of the scheduling process and provides a convenient solution for students and faculty to access and view timetables.

III. PROBLEM STATEMENT

The process of time scheduling is of the utmost importance as it involves creating schedules that adhere to specific constraints based on the given scenario. The aim is to generate timetables that effectively accommodate all relevant requirements and limitations. While manual scheduling is still prevalent for small-scale operations, larger ones often require computer-assisted solutions. The rapid progress of computational capacities has enabled artificial intelligence to arise as an appealing informatic means for improving existing solutions or exploring novel possibilities that had hitherto remained unexplored owing to a plethora of constraints. In this project, a university timetable scheduler will be developed using a genetic algorithm that incorporates adaptive and elitist traits. The objective is to generate a solution that is both valid and optimal while satisfying certain constraints.

IV. PROPOSED SYSTEM

A. Genetic Algorithm

A genetic algorithm (GA) is an optimization algorithm of a numerical nature that arises from the combination of natural selection and genetics. Unlike conventional procedural methods, the approach employed in this method is of a versatile nature, allowing it to be applied to a broader spectrum of problems. Genetic algorithms are instrumental in solving practical real-life problems on a day-to-day basis. The algorithms are straightforward to comprehend, and the corresponding computer code is simple to implement. Despite the growing number of enthusiasts, genetic algorithm engineering (GA) has not garnered as much attention as artificial neural networks, hill climbing, and various other methods. Its relative lack of attention is not due to any inherent limitations or a dearth of powerful analogies. The concept of evolution, deeply rooted in biodiversity observed in the world around us, serves as a potent and inspiring paradigm for addressing complex problems. From the early stages, the utilization of genetic algorithms became apparent as computer scientists envisioned systems that imitate and replicate various characteristics of life. During the 1950s and 1960s, the concept of employing a collection of solutions to address practical engineering optimization problems was explored repeatedly. John Holland is credited with the essential invention of the genetic algorithm (GA) during the 1960s. The primary motivation behind the development of these algorithms by John Holland was to tackle problems of broad significance and general interest. John Holland's visionary approach to the concept was extensively documented in his 1975 book, "Adaptation in Natural and Man-made Systems" (recently republished with additional content). This publication is highly recommended due to its insightful perspective. According to David (1999), the application of genetic algorithms has demonstrated its efficacy as a potent method for estimating various unknown parameters in physical system models. However, its versatility extends to a wide range of practical optimization issues, particularly those that are of utmost relevance to us, such as the specific scheduling problem within the context of this project.

A genetic algorithm is a computational technique inspired by the natural process of biological evolution, known as natural selection. The process follows an iterative approach and can be effectively visualized through a flowchart. In genetic algorithms, a solution is referred to as a chromosome, rather than an individual. The provided diagram illustrates the fundamental steps of the genetic algorithm. However, in practice, an additional step is included after the evaluation, which involves adjusting the environment in response to the evaluated solutions.

The generation of the population should involve a combination of random selection and greedy approaches. A random point will be selected as the starting point, and the array will be filled with suitable real values, following a deliberate and systematic approach. During the population creation process, it is crucial to ensure that all hard constraints are satisfied. Significant effort will be dedicated to addressing medium constraints, whereas soft constraints may be disregarded entirely.

Evaluation refers to the computation of the fitness value for a chromosome. Fitness is a metric that assesses the quality of the chromosome in relation to the desired solution. This event also serves as a termination criterion, indicating the completion of the process when a certain form or condition is achieved. Fitness calculations involved evaluating each chromosome under various moderate and mild stress levels. This assessment considered factors such as subject alignments, lunch breaks, rest sections, section idle time, people rest instructions, instructor load balancing, and meeting models.

The fitness calculation will largely be based on the number of objects drawn compared to the objects required. However, the calculation will still depend on the rating matrix provided by the user. An evaluation matrix is a set of constraint weights capable of shaping solutions. This is a list containing the priority of the constraints using a one hundred percent (100%) distribution.

Genetic algorithm is one of the most efficient schedule generation methods, though it cannot provide an assurance of a 100% flawless schedule. The degree of optimality achieved by genetic algorithms depends on the specific constraints utilized and the effectiveness of the adaptation function applied to the parameters. Using the genetic algorithm is relatively slow because of the steps involved but being the most efficient schedule generation method, its use is more convenient [2].

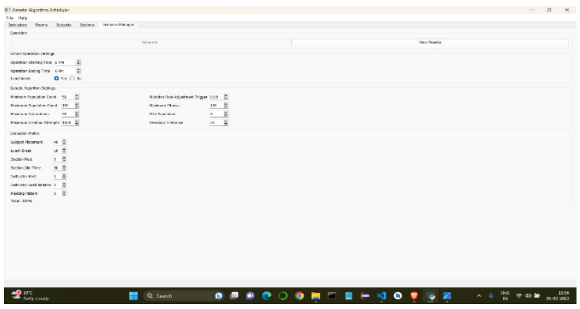

Genetic Algorithm Scheduler
File Help
Instructors | Rooms | Subjects | Sections | Scenario Manager
Operation
Generate | View Results
School Operation Settings
Operation Starting Time 6 AM
Operation Ending Time 6 PM
Lunch Break [✓ Yes] No
Genetic Algorithm Settings
Minimum Population Count 50
Maximum Population Count 100
Maximum Generations 50
Maximum Creation Attempts 1500
Mutation Rate Adjustment Trigger 0.08
Maximum Fitness 100
Elite Population 5
Deviation Tolerance 55
Evaluation Matrix
Subject Placement 40
Lunch Break 10
Section Rest 5
Section Idle Time 30
Instructor Rest 5
Instructor Load Balance 5
Meeting Pattern 5
Total: 100%

Fig.- 1. GUI of the Software

V. METHODOLOGY

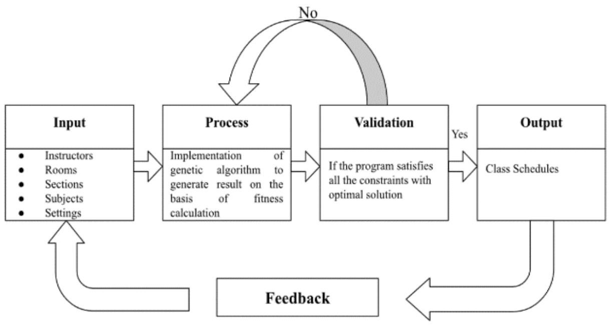

No
Input
Instructors
Rooms
Sections
Subjects
Settings
Process
Implementation of genetic algorithm to generate result on the basis of fitness calculation
Validation
If the program satisfies all the constraints with optimal solution
Yes
Output
Class Schedules
Feedback

Fig.- 2. Process flow

A genetic algorithm is used to generate the timetable. As shown in Figure 2, 4 input fields i.e., Instructors, rooms, sections and subjects are fed. Before starting the program, various settings are fixed like section rest time, instructor rest time, lunch break timing etc. The genetic algorithm is subsequently applied to generate results by evaluating the fitness of individuals. After a candidate solution is generated the program checks if all the constraints are satisfied. If it does not satisfy all the constraints then the process continues unless an optimal solution is generated.

The timetable is encoded as a set of chromosomes or individuals, where each chromosome represents a potential timetable solution. A fitness function is defined that quantifies the quality or fitness of a given timetable solution. The fitness function evaluates how well a timetable satisfies the defined constraints and objectives. It could consider factors like minimizing conflicts, maximizing resource utilization, and meeting preferences. An initialization method is developed to generate an initial population of timetable solutions. This could involve random assignment of classes to timeslots and rooms, ensuring that the initial solutions adhere to the defined constraints. Genetic operators, including crossover and mutation, are employed during the implementation process, to manipulate the solutions in the population. A selection mechanism is designed to choose individuals from the population for reproduction based on their fitness values. Individuals with higher fitness values have a higher probability of being selected, mimicking the natural selection process. Fitness of the newly generated offspring solutions are evaluated using the fitness function. Assess their performance in terms of satisfying the defined constraints and objectives. The solution with highest Fitness is selected as a final solution.

A. Requirement Analysis

1) *Data Format and Collection:* The application takes five major inputs namely Instructors, rooms, subjects,

sections and settings. First three inputs are given to the system in the form of CSV files.

a) *CSV Formatting:* When working with CSV files, it's important to use line 1 of the file as a file indicator, which should be one of the following: "instructors", "rooms", or "subjects". Additionally, the second line of the file should be dedicated to defining the table columns in the same format.

Example of subject CSV file-subjects.csv:

Subjects

Code, title, type, hours, splitable

BI, Business Intelligence1, lec,3,1

HPC, High performance computing, lec,3,1

NLP, Natural language processing, lec,3,1

DL, Deep Learning, lec,2,1

2) *Minimum System Requirements:* Following is the list of minimum system requirements that must be available to run this application.

a) *Operating System*: Windows 7 and Above

b) *CPU:* 2.0+ GHz with multithreading support

c) *RAM:* 1 GB (This still relies on the scenario size and algorithm configuration)

d) *Disk Space:* At least 1 GB Available Space (User generated files are excluded in assessment and minimum space is also subject to application usage)

B. Table Of Constraints

Table 1 shows constraints that are taken into consideration for generating a timetable.

TABLE 1: Table Of Constraints

Sr. No.	Soft Constraints	Medium Constraints	Hard constraints
1.	A set of rules that can be broken without affecting the validity of the output.	A set of guidelines may exist that, if breached, could affect the accuracy of the output. However, these guidelines should only be violated if the scenario is logically impossible or invalid	There may be a set of rules that, if disregarded or broken or not followed, would result in an invalid solution.
2.	Instructors should not get workload.	Sections' subjects are placed on the schedule.	The schedule should ensure that classes do not overlap or clash with each other.
3.	Students should have only 30 minutes break for every two hours of session per day	Sections should have at least 30 minutes vacant time between.	Instructors teach at their available schedule.
4.	Some instructors prefer not to have consecutive lectures.	Sections should have at least 30 minutes vacant time between.	Instructors can only take N number of subjects dependent on their maximum amount of load.
5.	Some instructors have preferred hours for their lectures.		The seating capacity of a classroom should be adequate to accommodate the enrolled number of students for a given course
6.	Instructors should have normalized distributed load based on the instructor pool of subjects.		It is not feasible to schedule two practical classes simultaneously in a single laboratory
7.	Lectures should not exceed three hours (pre-defined time slot).		A student cannot attend more than one class simultaneously

8.	Adequate and separate halls should be allocated for lectures.		Sections will stay in one room unless taking a laboratory subject.
9.	Students should have free periods between lectures.		Sections with subjects that are shared to other sections should produce a schedule that is compatible to all sharing participant.

C. Half Hour Algorithm

Most of the current systems do not provide the flexibility of scheduling a class of half hour duration. Given below is the algorithm for scheduling a half hour class.

```
Algorithm selectTimeDetails (subject, forceRandomMeeting)

  Step 1. Initialize variables
    - meetingPatterns = [[0, 2, 4], [1, 3]]
    - days = [0, 1, 2, 3, 4, 5]
    - hours = data['subjects'][subject][1]

  Step 2. Check if hours can be split into minimum sessions of 1 hour or 2 timeslots
    - if hours > 1.5 and ((hours / 3) % .5 == 0 or (hours / 2) % .5 == 0) and data['subjects'][subject][5]:

        a. If hours is divisible by two and three
          - if (hours / 3) % .5 == 0 and (hours / 2) % .5 == 0:
              - meetingPattern = random. choice(meetingPatterns)
              - if len(meetingPattern) == 3:
                  - meetingPattern = days[0:3] if forceRandomMeeting else meetingPattern
                  - hours = hours / 3
              - else:
                  - meetingPattern = days[0:2] if forceRandomMeeting else meetingPattern
                  - hours = hours / 2
        b. If hours is divisible by three
          - if (hours / 3) % .5 == 0:
              - meetingPattern = days[0:3] if forceRandomMeeting else meetingPatterns[0]
              - hours = hours / 3
        c. If hours is divisible by two
          - else:
              - meetingPattern = days[0:2] if forceRandomMeeting else meetingPatterns[1]
              - hours = hours / 2

  Step 3. Select a random day slot if hours cannot be split
    - else:
      - meetingPattern = [random. randint(0, 6)]

  Step 4. Convert hours into timetable timeslots
    - hours = hours / .5

  Step 5. Select a starting timeslot
    - startingTimeslot = False
    - startingTime = settings['starting_time']
    - endingTime = settings['ending_time']
    - while not startingTimeslot:
      - candidate = random.randint(0, endingTime - startingTime + 1)
```

```
        - if (candidate + hours) < endingTime - startingTime:
            - startingTimeslot = candidate
    Step 6. Generate output
        - return [meetingPattern, startingTimeslot, int(hours)]
```

D. Applications

The main goal of the Timetable generator is to optimize the allocation of resources, minimize conflicts, and improve overall efficiency and productivity in that specific domain.

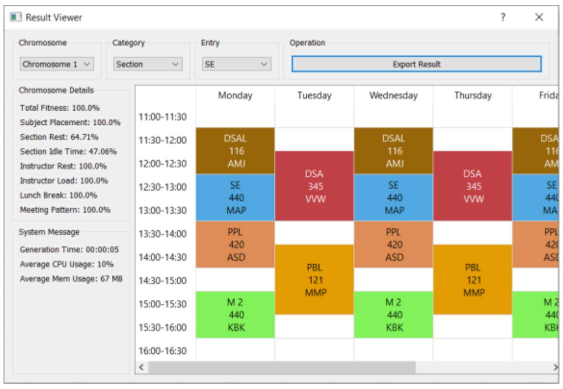
Timetable generator have various applications across different domains. Here are some common applications:

1) *Small scale Applications:*
   a) *Education Institutions:* It is widely used in schools, colleges, and universities to plan and organize class schedules for students and teachers. It helps in optimizing the allocation of resources such as classrooms, teachers, and subjects, ensuring a balanced and efficient timetable for the institution.
   b) *Event Planning*: It can play a significant role in event planning, whether it's conferences, seminars, or festivals. It can help in coordinating multiple activities, sessions, and speakers within a given time frame. Timetable scheduling ensures that all the components of the event are properly organized and synchronized.

2) *Large scale Applications:*
   a) *Public Transportation*: It can be crucial for public transportation systems like buses, trains, and airlines. It involves determining the routes, departure and arrival times, and frequency of services. It can help passengers plan their journeys and ensures smooth operations of public transportation networks.
   b) *Manufacturing and Production Planning*: It can be employed in manufacturing and production industries to plan and coordinate various production activities. It involves scheduling tasks, machine usage, maintenance activities, and material availability. It can help in maximizing production efficiency, meeting deadlines, and reducing downtime.
   c) *Healthcare Services:* It can be used in hospitals and clinics to manage patient appointments, surgeries, and other medical procedures. It helps in minimizing waiting times, optimizing the utilization of healthcare resources, and improving overall patient care.

VI. RESULTS

A schedule is produced as shown in Fig. 3 by a predictive algorithm that utilizes a hybrid approach of the genetic algorithm and particle swarm optimization algorithm to fulfill specific constraints, both mandatory and desirable. The algorithm ensures that subjects are assigned in a sequential manner without overlapping.



Chromosome Details
Total Fitness: 100.0%
Subject Placement: 100.0%
Section Rest: 64.71%
Section Idle Time: 47.06%
Instructor Rest: 100.0%
Instructor Load: 100.0%
Lunch Break: 100.0%
Meeting Pattern: 100.0%

System Message
Generation Time: 00:00:05
Average CPU Usage: 10%
Average Mem Usage: 67 MB

	Monday	Tuesday	Wednesday	Thursday	Frida
11:00-11:30					
11:30-12:00	DSAL 116 AMJ		DSAL 116 AMJ		DSA 11 AM
12:00-12:30		DSA 345 VVW		DSA 345 VVW	
12:30-13:00	SE 440 MAP		SE 440 MAP		SE 44 MA
13:00-13:30					
13:30-14:00	PPL 420 ASD		PPL 420 ASD		PPL 42 ASI
14:00-14:30		PBL 121 MMP		PBL 121 MMP	
14:30-15:00					
15:00-15:30	M 2 440 KBK		M 2 440 KBK		M 2 44 KBI
15:30-16:00					
16:00-16:30					

Fig.- 3. Generated Timetable

Here is a comparison between the outcome of the expected and actual outcome of the proposed system in Fig. 4. All the tests were conducted on a system with configuration of 4Gb RAM, AMD Ryzen 3 processor. Values may change with the size of the dataset and the system's configuration.

TABLE 2: Results Comparison

Metrics	Expected results	Actual Results
total fitness	100	99.98
sub placement	100	100
section rest	90	100
section idle time	23.21	37.29
Instructor Rest	100	97.56
Instructor load	50	63.7
Lunch break	100	100
Meeting pattern	95	100

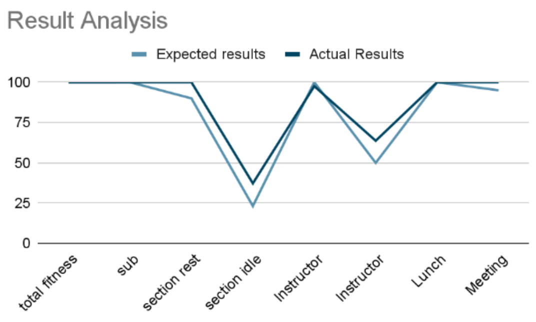


Fig.- 4. Result analysis of expected and actual results

Also, as the Genetic algorithm is based on evolution therefore the following results in the Table. 3, also signifies the generation time for timetable with the same input is comparatively less than the earlier generation meanwhile the average CPU usage is higher and average memory usage is lower.

TABLE 3: Output generation comparison

Metrics	Test1	Test2
Generation time	04:01 min	03:43 min
Average CPU usage	27	70
Average memory usage	110 Mb	74 Mb

In Fig. 5 and Fig. 6, the following results are the comparison of different chromosomes generated in the Generation of test data sets in Table. 4 as well real-world applications Table. 5 respectively. From the following results we can conclude that chromosome 1 is most likely to be the most optimal solution but it also differs on the real-world applications.

TABLE -4: Results data

Metrics		chr 1	chr 2	chr 3	chr 4	chr 5
total fitness		100	99.88	99.87	99.83	99.83
sub placement		100	100	100	100	100
section rest		100	100	100	98.25	98.25
section idle time		23.21	37.29	34.69	36.84	36.84
Instructor Rest		100	97.56	97.37	100	100
Instructor load		61.51	63.7	44.87	56.9	56.9
Lunch break		100	100	100	100	100
Meeting pattern		100	100	100	100	100

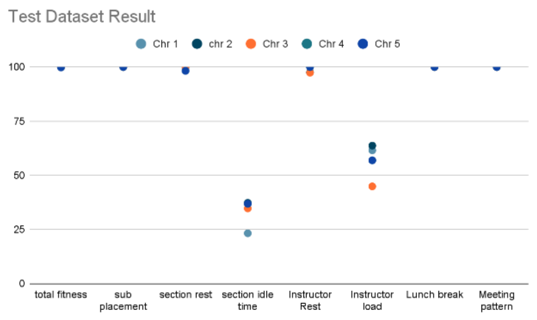


Fig. - 5. Analysis of Chromosome fitness and optimality of testing dataset

TABLE- 5: Results data

Metrics	Chr 1	Chr 2	Chr 3	Chr 4	Chr 5
Total Fitness	100%	95.27%	95.27%	95.27%	95.27%
Subject Placement	100%	94.74%	94.74%	94.74%	94.74%
Section Rest	64.71%	82.35%	53.33%	53.33%	64.71%
Section Idle Time	47.06%	41.18%	60%	40%	58.82%
Instructor Rest	100%	100%	100%	100%	100%
Instructor Load	100%	75%	100%	100%	100%
Lunch Break	100%	100%	100%	100%	100%
Meeting Pattern	100%	93.33%	100%	100%	100%

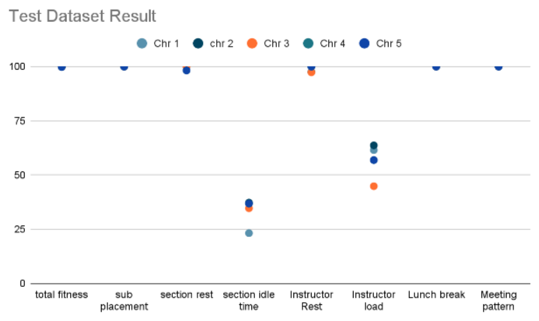


Fig.- 6. Analysis of Chromosome fitness and optimality of real-life applications

VII. CONCLUSION

The use of the model is to measure intelligence ability. It is observed from our results that the system has overcome the limitations of the existing systems and has performed more efficiently. Efficiency compared to the model; the system can generate solutions with at least 80% exercise (as the maximum solution for any situation). Timetable generated using genetic algorithm provides optimal outputs which enhances the in better management of classes and organization.

Moreover, the genetic algorithm-based timetable generation has had a profound impact on educational institutions' day-to-day operations. The optimized outputs provided by the algorithm enable administrators to allocate resources more effectively, minimize conflicts in scheduling, and maximize student and teacher productivity. Consequently, the intricate machinations and multifaceted operations of academic establishments have benefitted from fluid functionality, an amplified immersion of pupils, and advanced total execution.

Additionally, the optimized timetable generated by the genetic algorithm has potential applications in areas such as event management, transportation logistics, and healthcare scheduling. If the system can adeptly apportion assets and mitigate frictions, it may heighten functioning, decrease expenditures, and better the total organizational proficiency in these realms.

REFERENCES

[1] G. Alnowaini and A. A. Aljomai, "Genetic Algorithm for Solving University Course Timetabling Problem Using Dynamic Chromosomes," *IEEE International Conference of Technology, Science and Administration (ICTSA)*, Taiz, Yemen, 2021, pp. 1-6, doi: 10.1109/ICTSA52017.2021.9406539.

[2] K. Williams, "Automatic Timetable Generation Using Genetic Algorithm," *IISTE: Computer Engineering and Intelligent Systems*, vol. 10, no. 4, 2019, doi: 10.7176/CEIS/10-4-04.

[3] B. Shruthi, "Automatic Time Table Generator Using Genetic Algorithm," *Journal of Engineering Science*,vol. 11, no. 5, May 2020, ISSN 0377-9254.

[4] A. Ansari, "Automatic Time-Table – Implementation Using Genetic Algorithm," *Journal of Computing Technologies*, vol. 6, no. 4, 2017, ISSN: 2278-3814.

[5] S. Thakare, T. Nikam, and M. Patil, "Automated Timetable Generation using Genetic Algorithm," *International Journal of Engineering Research & Technology (IJERT)*, vol. 9, no. 7, Jul. 2020, ISSN: 2278-0181.

[6] S. Shinde, S. Gurav and S. Karme, "Automatic Timetable Generation Using Genetic Algorithm," *International Journal of Scientific & Engineering Research*, vol. 9, no. 4, Apr. 2018, ISSN 2229-5518.

[7] S. D. Immanuel and U. K. Chakraborty, "Genetic Algorithm: An Approach on Optimization," *2019 International Conference on Communication and Electronics Systems (ICCES)*, Coimbatore, India, 2019, pp. 701-708, doi: 10.1109/ICCES45898.2019.9002372.

[8] M. Y. Orong, A. M. Sison and R. P. Medina, "A new crossover mechanism for genetic algorithm with rank-based selection method," *2018 5th International Conference on Business and Industrial Research (ICBIR)*, Bangkok, Thailand, 2018, pp. 83-88, doi: 10.1109/ICBIR.2018.8391171.

[9] I. K. Gupta, "A hybrid GA-GSA algorithm to solve multidimensional knapsack problem," *2018 4th International Conference on Recent Advances in Information Technology (RAIT)*, Dhanbad, India, 2018, pp. 1-6, doi: 10.1109/RAIT.2018.8389069.

[10] S. Markal, S. Ghorpade and D. Chalke, "Timetable Generator," *IOSR: Journal of Computer Engineering (IOSR-JCE)*, vol. 22, no. 2, pp. 29-33, Mar-Apr. 2020, doi: 10.9790/0661-2202022933.

[11] D. Mittal, H. Doshi, M. Sunasra, and R. Nagpure, "Automatic Timetable Generation using Genetic Algorithm," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 4, no. 2, Feb. 2015. doi: 10.17148/IJARCCE.2015.4254

[12] T. Roobasurya "Automatic Timetable Generation Using Genetic Algorithms," *International Journal of Research Publication and Reviews*, vol. 2, no. 4, pp. 19-24, 2021.ISSN 2582-7421.

[13] Y. S. Chaudhari, V. W. Dmello, S. S. Shah and P. Bhangale, "Autonomous Timetable System Using Genetic Algorithm," *2022 4th International Conference on Smart Systems and Inventive Technology (ICSSIT)*, Tirunelveli, India, 2022, pp. 1687-1694, doi: 10.1109/ICSSIT53264.2022.9716370.

[14] M. Gaikwad, A. Gaikwad, M. Chaudhary, D. Sawarkar, M. Bhargava, and V. Anaspure, "Auto Timetable Generator," *International Research Journal of Modernization in Engineering Technology and Science*, vol. 4, no. 5, pp. 2976-2981, May 2022. e-ISSN: 2582-5208.

[15] A. Salaskar, R. Malla, A. Singh, P. Sahani, and M. Khorajiya, "Timetable Generator using Genetic Algorithm," *International Journal of Creative Research Thoughts (IJCRT)*, vol. 11, no. 4, pp. a646-a648, April 2023. ISSN: 2320-2882.

[16] A. Ansari and S. Bojewar, "A Timetable Prediction for Technical Educational System Using Genetic Algorithm – An Over View," *International Journal of Scientific and Research Publications*, vol. 4, no. 12, December 2014. ISSN: 2250-3153

[17] K. R. Rashmi and A. M. B., "Automated University Timetable Generation using Prediction Algorithm," *International Research Journal of Engineering and Technology (IRJET)*, vol. 08, no. 06, pp. 2345-2350, June 2021. ISSN: 2395-0056.