



Optimizing Timetable Generation for Educational Institute Using Genetic Algorithm

Virti Shah^{1*}, Aastha Gadhvi², Neel Oza³, Monali Sankhe⁴ and Monika Mangla⁵

^{1,2,3,4,5} Information Technology, DJSCE, Mumbai, India
virt1709@gmail.com, aasthagadhvi123@gmail.com, neelo2003@gmail.com,
monali.sankhe@djsce.ac.in, monika.mangla@djsce.ac.in

Abstract. The timetable generation process is a critical component of educational institutions, characterized by its inherent complexity and NP-hard nature due to a multitude of constraints. This paper provides a comprehensive comparative analysis of various methods employed in timetable generation, with a particular focus on Genetic Algorithms (GAs). GAs offer a promising approach to solving combinatorial optimization problems efficiently, which makes them well suited for addressing the challenges of timetable scheduling. We evaluate the performance of GA-based methods in comparison with heuristic algorithms, examining key factors such as solution quality, computational efficiency, and scalability. By doing so, we aim to provide valuable insights into the effectiveness and suitability of GA-based strategies for timetable generation systems. This study highlights the strengths and limitations of GAs in handling the intricacies of scheduling tasks, contributing to a deeper understanding of their potential in overcoming the challenges inherent in timetable generation. The findings offer significant implications for future research and practical applications in this domain.

Keywords: Hereditary calculation, Dynamic Principles, Rule based specialists, asset planning, heuristic calculations, Genetic Algorithm.

1 Introduction

Timetable scheduling for higher education is complex due to the diverse expertise of teaching staff and limited resources like time slots and classrooms [1]. The process involves meeting hard constraints, which are mandatory, and accommodating soft constraints, which are desirable but flexible [2].

At Dwarkadas J. Sanghvi College of Engineering, manual timetable creation is time-consuming and inefficient. Genetic Algorithms (GAs), inspired by natural selection, offer a robust solution [3]. Introduced by John Holland in the 1970s, GAs optimize solutions by evolving potential answers through selection and mutation, eliminating weaker candidates and improving performance over generations [4].

GAs excel over traditional AI methods by adapting to changes and noisy data. They rely on defining an objective function, genetic representation, and genetic operators to evolve solutions effectively, enabling further improvements and optimal results.

© The Author(s) 2025

S. Bhalerao et al. (eds.), *Proceedings of the International Conference on Recent Advancement and Modernization in Sustainable Intelligent Technologies & Applications (RAMSITA-2025)*, Advances in Intelligent Systems Research 192,

https://doi.org/10.2991/978-94-6463-716-8_37

2 Literature Review

Timetable planning is an extremely constrained NP-hard problem, often referred to as the timetabling issue. Developing a computational solution for this problem requires addressing several complex constraints. This document explores three methods used to generate efficient timetables: Genetic Algorithms, Heuristic Methods, Graph Coloring Algorithm, and Priority Queue.

Various approaches have been developed to optimize the allocation of courses, lecturers, and venues. One method ranks courses by factors such as study level, scope, and class size to prioritize scheduling [5]. Another key aspect is considering lecturers' availability, ranked based on their engagement weights to resolve conflicts. These systems often incorporate ICT advancements, such as web-based platforms and spatial information systems, allowing for dynamic updates and remote access to timetables [6].

In tackling constraints, provisions for students' activities and other institutional events are integrated into the scheduling process. This helps in minimizing disruptions and ensuring that all identified constraints are satisfied. The literature also highlights the necessity of a systematic approach to timetable management, particularly in institutions with shared facilities and limited resources. The systems developed often include features like automated alerts and adaptable scheduling tools to accommodate unforeseen changes [7].

A prevalent method for timetable optimization is the use of Genetic Algorithms (GAs), which mimic natural selection to evolve solutions that meet various constraints. GAs are often combined with heuristic methods to improve efficiency and adaptability to dynamic scheduling changes [8].

Another approach is the Graph Coloring algorithm, which assigns distinct "colors" to nodes in a graph to prevent adjacent nodes from sharing the same color. This method helps avoid timetable clashes, especially in scenarios with limited resources like classrooms or instructors [9]. Hybrid algorithms, combining GAs with Graph Coloring, aim to leverage the strengths of both techniques for more robust and adaptable scheduling [10].

Timetable generation algorithms illustrate the complexity of this NP-hard problem, which requires balancing various constraints. GAs are popular due to their robustness and ability to evolve solutions, often complemented by heuristic methods like rule-based approaches to ensure feasibility and optimization. A growing trend is the hybridization of GAs with other techniques, such as Bacterial Foraging Optimization (BFOA) [11], to enhance search and convergence capabilities, providing more efficient solutions to timetable scheduling.

2.1 Constraints:

1. Teacher must not be present in more than one classroom at the same time
2. Classes and Labs should not clash.
3. Free time distribution for Teachers and Students.
4. Classrooms or labs must not be left unallocated.
5. Even distribution of working hours of Teachers and Students.

Table 1. Comparison of Algorithms

Aspect	Genetic Algorithm	Heuristic	Graph Coloring	Priority Queue
Performance	Near-optimal solutions in complex search spaces	Quick solutions based on thumb	Adaptable scheduling, mild to high time complexity	Efficient retrieval of highest-priority items
Time Taken	Moderate high, converge quickly	Typically fast, may prioritize speed over optimality	Moderate high, depends on graph complexity	Efficient retrieval, time complexity depends on implementation
Deployment Status	Widely deployed in various domains, including timetabling	Widely used in due simplicity speed	Utilized in scheduling and problems	Essential data structure in many algorithms and Applications
Constraints Considered	Can handle various constraints effectively	Can handle a subset of constraints, may struggle with complexity	Can handle a related resource allocation and conflict avoidance	Can manage constraints by prioritizing tasks or resources
Availability of Resources Handled	Can handle large number of suitable resources encoding	May struggle with large-scale problems	Suitable moderate number resources	Efficiently manages large number of resources by prioritization

GAs are a popular choice for timetable generation systems due to their ability to efficiently search for optimal solutions in large solution spaces. In the context of timetable generation, GAs can handle complex constraints, preferences, and various objectives [12]. They allow for the exploration of a wide range of possible timetables, enabling the system to find near-optimal or even optimal schedules that meet diverse criteria.

3 Use of Genetic Algorithm to Generate the Automatic Timetable

Timetable planning is described as an extremely forced NP-hard problem. Most experts refer to it as the timetabling issue. To develop a good computation to understand this problem, some complex imperatives have to be addressed.

Four boundaries are accepted as information in the proposed paper: Speakers' names, individual Subject: Course names for the class Room: class names and boundaries for each time stretch; start and end times of the term

Event Condition-Action (ECA) rules operate on the principle: "When an event occurs, check the condition, and if true, execute the action." The event triggers the rule, the condition validates it, and the action executes the response, potentially creating new ECA rules [13]. In this context, Genetic Algorithms (GAs) play a pivotal role in selecting optimal actions by maintaining a population of chromosomes represented as strings of symbols or numbers. These chromosomes, also known as genotypes, are evaluated for fitness, and less effective ones are discarded [14]. This evolutionary process refines solutions over iterations, enabling dynamic responses to events and improving planning efficiency by reducing testing and turnaround times [15].

The authors propose a modified approach where GAs prioritize actions dynamically for various tasks triggered by events, favoring distributed arrangements, stack adjustments, and fault management. This method maximizes scheduling efficiency by exploring multiple configurations and selecting the best one, although challenges arise when only one optimal solution exists [16]. In such cases, the algorithm can resume using Dynamic Rules to identify superior configurations. This streamlined approach minimizes development and support times while enhancing rule-based systems' adaptability and effectiveness [17].

4 Proposed Work

This paper proposes a strategy to solve the timetabling issue, considering various constraints such as room, teacher, course, and number of hours. The strategy dynamically adjusts resources based on their complexity, using a heuristic approach to manage and check constraints, which can be manually modified.

Two methods are discussed: the first method prepares the timetable after course registration, while the second creates the timetable first, followed by registration. The first method was suitable when resources were abundant, whereas the second was used when resources were limited. Constraints are categorized as "Hard" and "Soft"; hard constraints are unavoidable, such as the inability of a teacher or student to be in two places at once, while soft constraints can be disregarded if impractical.

Heuristic algorithms form the foundation of the suggested algorithm. The algorithm schedules resources and controls restrictions one at a time using values as input. The following are the algorithm's primary characteristics which is also depicted as System Architecture in Fig 1:

- The system generates weekly timetables for teachers, students, and rooms.

- It evenly distributes the lecture workload across time slots.
- It ensures fair course distribution for students.
- Users can log in securely with appropriate access levels.
- The system detects and resolves conflicts like overlapping schedules.
- It optimizes the timetable based on preferences and minimizes conflicts.
- It tracks publication adjustments and marks non-modifiable courses.
- It groups lectures within the same time frame on weekdays.
- Programming techniques ensure efficient operation despite complexity.
- It prioritizes time slots, adjusting schedules and shifting unmovable lectures.

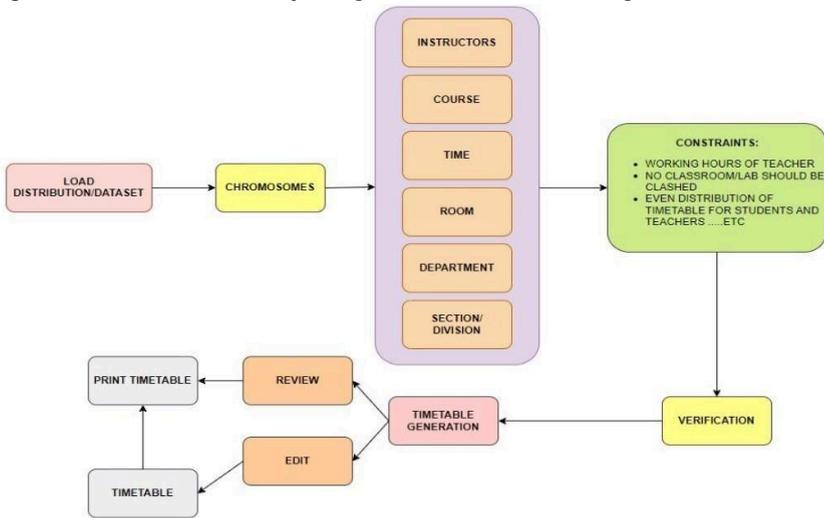


Fig. 1.System Architecture

The algorithm was put into practice by taking the subsequent actions:

- Chromosomes capture course, teacher, subject, practical, and hours info.
- The evaluation function minimizes conflicts, maximizes resource use, and satisfies constraints.
- Genetic operators like crossover, selection, and mutation evolve the population.
- Generations iterate, applying selection, crossover, and mutation to create offspring.
- A cutoff point is set based on generations or solution quality.
- The best-performing chromosome (timetable) is selected as the solution.
- The timetable is tested for feasibility and effectiveness against scenarios.

This timetable generation system boasts cutting-edge tools and techniques, empowering users with unmatched flexibility. No detail is overlooked. Timetables are dynamically generated for diverse levels, including campus, department, class and even individual teachers. The algorithm employs a bottom-up strategy.

In essence, this system prioritizes user control and adaptability, using advanced algorithms to generate comprehensive and conflict-free timetables at various levels, making it a powerful tool for universities.

5 Design and Implementation

This section presents a comprehensive strategy for lecture-timetable planning using Genetic Algorithms (GAs), treating it as a constraint satisfaction problem. Various methods, including graph coloring, local search (e.g., simulated annealing, tabu search), and backtracking, have been proposed for generating timetables. GAs stands out by breaking the planning process into essential modules to meet specified requirements effectively.

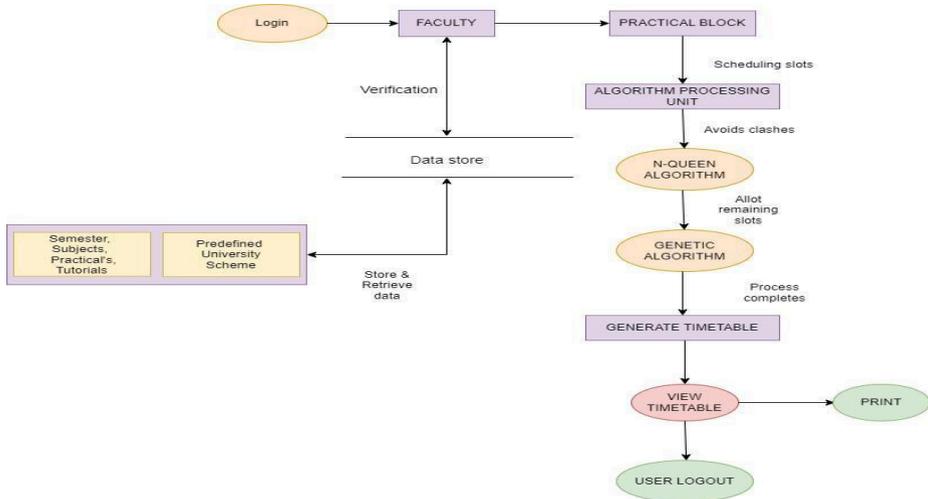


Fig.2.Flow Chart

5.1 Flow Chart

Flow Chart as shown in fig. 2. Below is the description of the flowchart:

1. Faculty begins by logging into the system.
2. Ensures only authorized users can access the system.
3. Stores and retrieves necessary information: Semester details, subjects, practical, tutorials.
4. Allocates practical slots while avoiding slot clashes through the
5. N-Queen Algorithm: Assigns remaining slots, ensuring no conflicts.
6. Genetic Algorithm: Optimizes the timetable by processing and refining allocations.
7. Completes the process by generating the final timetable.
8. Faculty can view the generated timetable and can print the timetable if required.
9. Faculty logs out after completing their tasks.

5.2 Population Evaluation

Population evaluation involves assessing the quality of a solution based on defined criteria. This is a central aspect of GAs, where the fitness of solutions is evaluated to

determine their effectiveness. A fitness value between 0 and 1 is used, with 1 indicating the best solution. The fitness function distinguishes between better and worse solutions, helping guide the algorithm towards optimal solutions.

5.3 Fitness Function

The fitness function is defined as:

$$F(X) = \frac{\rho_1(x) - \rho_V(x)}{\rho_1(xWT) - \rho_V(xWT)} \tag{1}$$

Where, $\rho_1(x)$ = Average density of the sequence of x
 $\rho_V(x)$ = Analogue for the poor phase

A timeline that is being assessed is indicated by X. W is the total number of limitations. T is the total fitness value.

5.4 Crossover Evolution

Crossover evolution is a technique used to generate new populations from existing ones. It involves two parent chromosomes from which new offspring chromosomes are created. This process is visualized in Fig 3, which shows how the chromosomes are divided and recombined to produce new genetic material.

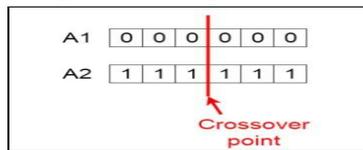


Fig.3.Crossover Point

5.5 Encoding and Decoding of Data

The initial step in a Genetic Algorithm (GA) is encoding data. This involves converting a solution into a chromosome-like string to facilitate the algorithm’s processing. Data is typically encoded as binary strings, representing different qualities or features. This encoding simplifies the manipulation and analysis of solutions within the algorithm.

5.6 The Initial Population

The first stage of a GA involves generating an initial population of solutions. This is done by creating a set number of random individuals based on constraints and requirements. Generally speaking, a larger population produces greater results, but it also takes more time and resources. Conversely, a smaller population may evolve more quickly but might not provide as robust solutions.

5.7 Mutation

Mutation is a mechanism that introduces variation into the population by randomly altering the values of genes within a chromosome. This process generates new and potentially better solutions by exploring different aspects of the solution space. Mutation operates independently of other solutions, allowing for unique arrangements to be considered in the optimization process.

6 Result and Analysis

6.1 Login Page

The login page allows students and teachers to access the application by entering their usernames and passwords shown in Fig 4. This serves as a primary interface for users to provide input, either through text or forms, which is then processed by the system's main functions.



Fig.4. Login Page

6.2 Adding Teachers and Subjects

This feature enables the addition of teachers and subjects to manage faculty data. It includes fields like Faculty Number, Name, Designation, Contact Number, Email ID, and the number of classes assigned to each teacher as shown in Fig 5 and Fig 6. This module is crucial for invoking other processes such as crossover, mutation, and selection within the system.

6.3 Adding Classrooms

Classrooms are fundamental components within the system's Genetic Algorithm. They represent key units with specific attributes essential for managing the educational environment as shown in Fig 7. This module includes the printclasses() method for monitoring algorithm performance and verifying system processes.

6.4 Generating Timetables

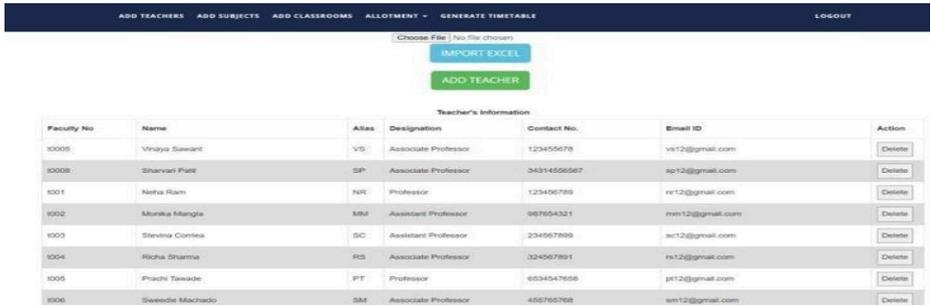


Fig.5. Add Teacher

Here, the user can add the details of Teachers or can also upload Excel sheet.

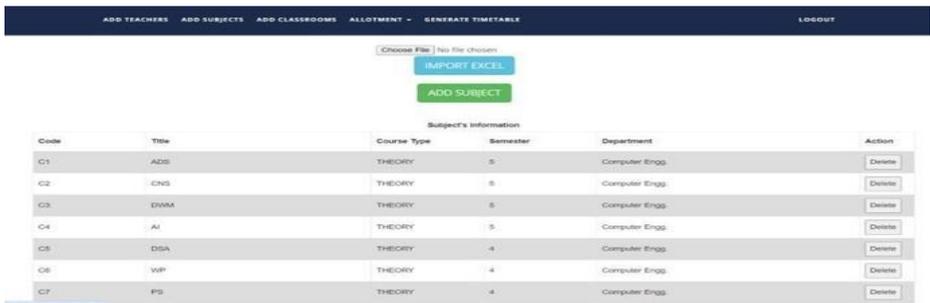


Fig.6. Add Subject

Here, the user can add the details of Subjects.

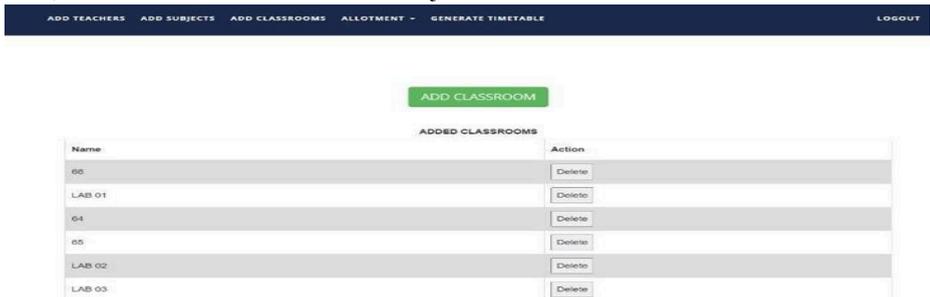


Fig.7. Add Classroom

Here, the user can add the details of Classrooms.

This function initializes a set of “spaces” for each class, effectively generating a new timetable structure. It takes into account various constraints and inputs to optimize the scheduling process. The desired output is shown in Fig 8 and Fig 9.

INFORMATION TECHNOLOGY DEPARTMENT							
WEEKDAYS	MONDAY	TUESDAY	WENESDAY	THURSDAY	FRIDAY	SATURDAY	1:30-2:30
8:00-9:00	C1 SM (64)	C6 RS (64)	C5 SC(64)	LUNCH	C4 VS (64)	C1_1 NR, SC, PT	C1_1 NR, SC, PT
9:00-10:00	C2 ARJ (65)	C1 NR (65)	C6 SC (65)	LUNCH	C5 RS (65)	C2_2 RS, PT, ARJ	C2_2 RS, PT, ARJ
10:00-11:00	C3 SV (64)	C2 VS (64)	C1 NR (64)	LUNCH	C6 SC (64)	C3_1 SR, AJ, SC	C3_1 SP, AJ, SC
11:00-11:30	C4 PT (64)	C3 SV (64)	C2 VS (64)	LUNCH	C1 NR (64)	C4_1 ARJ, MM, SM	C4_1 ARJ, MM, SM
11:30-12:30	C5 RS (64)	C4 PT (64)	C3 SV (64)	LUNCH	C2 VS (64)	C5_1 RS, ARJ, VS	C5_1 RS, ARJ, VS
12:30-1:30	C6 SC(64)	C5 RS (65)	C4 PT (65)	LUNCH	C3 SV (65)	C1_1 NR, SC, PT	C1_1 NR, SC, PT

Fig.8. Generated Timetable

Here, the user views the generated timetable for Class I2 (Sem 6).

Hello Sreedie Machado								LOGOUT
Sreedie Machado								
WEEKDAYS	8:00-9:00	9:00-10:00	10:00-11:00	11:00-11:30	11:30-12:30	12:30-1:30	1:30-2:30	
MONDAY	C1 (64)	-	S3 (66)	LUNCH	-	-	-	
TUESDAY	-	-	-	LUNCH	-	-	-	
WEDNESDAY	S3 (66)	C3 (65)	-	LUNCH	-	-	-	
THURSDAY	-	-	-	LUNCH	C3 65	C4_1 Lab 03	C4_1 Lab 03	
FRIDAY	-	-	C3 65	LUNCH	-	-	-	
SATURDAY	C4_1 Lab 03	C4_1 Lab 03	-	LUNCH	-	-	-	

S3_3_AJ
S11_1_A03B Lab

Fig.9. Generated Timetable (Teacher)

Here, the user views the generated timetable for Professor.

Below Table-2 shows the test cases passed based on various inputs. Various testing strategies is used in order to test the system as follows:

- a. **Functionality Testing:** Core features like user login, resource management, and timetable generation were verified, with all test cases passing without defects.
- b. **Usability Testing:** Users praised the system’s intuitive interface, navigation, and responsive design across devices.
- c. **Performance Testing:** The system met performance expectations, completing the generation timetable efficiently under normal load.
- d. **Regression Testing:** Existing functionalities remained stable after updates, with no regressions observed.
- e. **Boundary Testing:** The system handled edge cases and boundary conditions effectively, demonstrating robustness and error-handling capabilities.

Table 2. Test Cases and Results

Test ID	Case Description	Input Data	Expected Output	Actual Output	Status
TC-001	User Login	Username, Password	Login Successful	Login Successful	Pass
TC-002	Add Teacher	Teacher Details	Teacher Added Successfully	Teacher Added	Pass
TC-003	Add Course	Course Details	Course Added Successfully	Course Added	Pass
TC-004	Add Subject	Subject Details	Subject Added Successfully	Subject Added	Pass
TC-005	Add Classroom	Classroom Details	Classroom Added Successfully	Classroom Added	Pass
TC-006	Generate Timetable	Schedule Parameters	Timetable Generated Successfully	Timetable Generated	Pass
TC-007	Validate Teacher Assignment	Course, Teacher ID	Teacher Assigned Correctly	Teacher Assigned	Pass
TC-008	Validate Course Scheduling	Course, Time Slot	Course Scheduled Correctly	Course Scheduled	Pass
TC-009	Validate Classroom Assignment	Classroom, Time Slot	Classroom Assigned Correctly	Classroom Assigned	Pass
TC-010	Check Timetable Conflicts	Teacher, Classroom, Time	No Conflicts Detected	No Conflicts	Pass

7 Conclusion

This research introduces a novel timetable generation system leveraging Genetic Algorithms (GAs). GAs efficiently explore large solution spaces by encoding timetables as chromosomes and evolving them through selection, crossover, and mutation. Compared to graph coloring and heuristic methods, GAs adapt better to complex constraints, dynamic scenarios, and large datasets. While graph coloring algorithms often struggle with evolving requirements, and heuristics provide quick but suboptimal solutions, GAs consistently deliver optimal or near-optimal results. The system achieved fewer constraint violations (5 vs. 20) and higher teacher preference satisfaction (95% vs. 70%) compared to alternatives. Priority queues, though efficient for specific tasks, lack the scalability and adaptability of GAs for timetabling.

The GA-based system demonstrates flexibility in handling dynamic adjustments and complex preferences, achieving improved timetable optimality while maintaining runtime efficiency comparable to priority queue approaches. While computational complexity for large datasets remains a challenge, this research highlights the scalability and robustness of GAs for automated timetable generation. Future work

could explore GA variations, integrate deeper user preferences, and address application-specific constraints for enhanced performance and applicability.

References

1. H. Alghamdi, T. Alsubait, H. Alhakami, and A. Baz, "A Review of Optimization Algorithms for University Timetable Scheduling", *Eng. Technol. Appl. Sci. Res.*, vol. 10, no. 6, pp. 6410–6417, Dec. 2020.
2. H. Kuttymammi and L. Ying Ying, "Timetable Scheduling System using Genetic Algorithm for School of Computing (tsuGA)", *Int J Innov Comp*, vol. 11, no. 2, pp. 67–72, Oct. 2021.
3. Vishal Nandal, SushmaYadav, and Kamlesh Dahiya "Graph Coloring based Scheduling Algorithm to Automatically Generate College Course Timetable" *IEEE 11th International Conference on Cloud Computing, Data Science Engineering (Confluence 2021)*.
4. Dipesh Mittal, Hiral Doshi, Mohammed Sunasra, and Renuka Nagpure "Automatic Timetable Generation using Genetic Algorithm" *International Journal of Advanced Research in Computer and Communication Engineering*, Vol. 4, Issue 2, February 2015.
5. Constance Kalu, Simeon Ozuomba, Sylvester Isreal Umama "Development of Mechanism for Handling Conflicts and Constraints in University Timetable Management System" *Communications on Applied Electronics (CAE)*—ISSN: 2394-4714 Foundation of Computer Science FCS, New York, USA Volume 7— No. 24, December 2018 – www.caeaccess.org
6. Anisha Jain, Ganapathy S C Aiyer, HarshitaGoel, Rishabh Bhandari "A Literature Review on Timetable generation algorithms based on Genetic Algorithm and Heuristic approach" *International Journal of Advanced Research in Computer and Communication Engineering* Vol. 4, Issue 4, April 2015 Copyright to IJARCCCE doi:10.17148/IJARCCCE.2015.4437159
7. Barkha Narang, Ambika Gupta, and Rashmi Bansal, "Use of Active Rule and Genetic Algorithm to Generate Automatic Time-Table," in *International Journal of Advances in Engineering Sciences* Vol.3 (3), July, 2013.
8. Tahir Afzal Malik, Hikmat Ullah Khan, and Sajjad Sadiq, "Dynamic Time TableGeneration Conforming Constraints a Novel Approach," in *ICCT 2012*.
9. Om Prakash Verma, Rohan Garg, and Vikram Singh Bisht, "Optimal Time-Table Generation by hybridized Bacterial Foraging and GAs," in *International Conference on Communication Systems and Network Technologies*, 2012.
10. Ashish Jain, Suresh Jain, and P.K. Chande, "Formulation of Genetic Algorithm to Generate Good Quality Course Timetable," in *International Journal of Innovation, Management and Technology*, vol. 1, no. 3, Aug. 2010, pp. 248-251.
11. E.K. Burke, J.P. Newall, and R.F. Weare, "A Memetic Algorithm for University Exam Timetabling," in *The Practice and Theory of Automated Timetabling*, E.K. Burke and P. Ross, Eds., Springer-Verlag, Berlin, 1996, pp. 241-250.
12. J. Thompson and K.A. Dowsland, "Variants of Simulated Annealing for Examination and Timetabling Problem," in *Annals of Operations Research*, vol. 63, 1996, pp. 105-128.
13. E.K. Burke, K. Jackson, J. H. Kingston, and R.E. Weare, "Automated University Time tabling: The State of the Art," in *The Computer Journal*, vol. 40, no. 9, 1997.
14. Alberto Colorni, Marco Dorigo, and Vittorio Maniezzo, "A Genetic Algorithm to Solve the Timetable Problem," in *Computational Optimization and Applications*, submitted to the journal.
15. D.E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, Massachusetts, 1989.

16. M.R. Garey and D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, in W.H. Freeman and Company, 1979.
17. J.H. Holland, Adaptation in Natural and Artificial Systems, in The University of Michigan Press, Ann Arbor, Michigan, 1975. Reprinted by MIT, 1992.

Open Access This chapter is licensed under the terms of the Creative Commons Attribution-NonCommercial 4.0 International License (<http://creativecommons.org/licenses/by-nc/4.0/>), which permits any noncommercial use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

