Design of a Smart Timetable Generator Using Artificial Intelligence for NEP 2020 Multidisciplinary Learning Models

Affrina Prince A.[1], Gokul Rama Subbiah [2], Sanjay V.[3]

Artificial Intelligence and Data Science, Bannari Amman Institute of Technology, Sathyamangalam, India

E-mail: [1]affrinaprince.ad22@bitsathy.ac.in, [2]gokularama.al22@bitsathy.ac.in, [3]sanjay.al22@bitsathy.ac.in

Abstract

Timetable generation is an extremely complicated problem of combinatorial optimization with numerous challenging limitations constraints. As a result, newly introduced requirements under the National Education Policy (NEP) 2020 for academic institutions (i.e., credit flexibility, multiple options for completion of courses across other disciplines and more dynamic schedules) has become complicated to generate a timetable. In this study, the research developed a newly adaptive GA (Genetic Algorithm) for the generation of a course schedule in a highly efficient and scalable machine-learning-based parameter modifications to implement NEP 2020 credit flexibility constraints and to modify both the probability of crossover and mutation dynamically based on the predicted velocity of the GA (Genetic Algorithm) to converge as indicated by a supervised learning-based model. The work developed the GA (Genetic Algorithm) to operate using Python for backend processes, React for user interface development and MySQL for data management. The experimental evaluation of a medium-sized dataset (3 departments, 96 courses, 54 instructors, 24 classrooms and 40 weekly time slots) indicates that the method resulted in better training convergence and execution than traditional GA-based solutions. Therefore, the methodology has the potential for increased speed of convergence, fewer constraint violations and is suitable for today's higher education scheduling environments.

Extended version of article received from National Conference on Intelligent Networks and Systems for Innovation using Generative AI in High-End Technologies
DOI: https://doi.org/10.36548/jitdw.2026.1.003
Received: 15.01.2026. received in revised form: 17.02.2026, accepted: 09.03.2026, published: 20.03.2026

Keywords: Genetic Algorithm, Machine Learning, NEP 2020, Timetable Generation, Constraint Optimization, Adaptive Parameter Tuning.

1. Introduction

Timetable creation at higher education institutions is a difficult and computationally demanding task that has been researched within the fields of combinatorial optimization. Timetable creation requires allocating courses, assigning instructors, providing rooms and times for those classes so that various institutional constraints are maximizing the resource use [1] [3]. Because there are many dependent and independent variables in this process, the number of ways to generate a timetable increases exponentially based on how many classes are offered every year to students, making the overall difficulty level of creating a timetable NP-hard. Previous methods of manual timetable generation have always been time-consuming, inefficient and not adaptable enough to respond to the continually evolving academic sector. As a result of this demand for automated and smart systems that can manage large volumes of data while also ensuring accuracy and consistency in scheduling, researchers are putting more emphasis on these systems [12] [4].

The introduction of the National Education Policy (NEP) 2020 has fundamentally modified the structure of education in large institutions. NEP 2020 promotes more flexible learning models that incorporate credit-based courses, each aligned to a particular discipline, multiple entry/exit points and elective courses from various disciplines. Although these advancements will improve both the quality and accessibility of educational services at these institutions [5-8]. This will also include additional levels of complexity among other constraints involved in creating a timetable. Educational institutions are facing increasing numbers of scheduling constraints due to course overlap, course prerequisites and total credits earned by students. Thus, while constructing a schedule without conflicting with other classes, institutions must now accommodate the above constraints simultaneously. Advanced optimization techniques and methods will be needed to ensure that current scheduling conflicts are resolved and new regulatory constraints are addressed [2,16].

This will include using computational methods such as Genetic Algorithms (GA) to produce optimal solutions in the wide area of schedule creation. The use of GA begins with encoding scheduling solutions and using evolutionary operations (selection, crossover, mutation) to evolve solutions over time. There are, several limitations associated with traditional forms of GA such as low convergence rates, premature convergence and sensitivity to GA parameters. A

major innovation of this research will be the ability to adaptively change GAs (i.e., GA parameter settings) dynamically throughout the execution of the GAs based on convergence behaviur to achieve more appropriate searching and avoid being trapped in local optima. Furthermore, this approach will feature modelling of NCAPPE 2020 (National Centre for Academic Publishing and Publishing Education) regulatory constraints. This ensures the Developed Schedule / Timetable is both optimally configured for the Educational Institution and compliant with current regulations. When integrating evolutionary optimization techniques with smart parameter adaptation techniques, the proposed method provides a scalable, effective and robust solution for automated schedule / timetable generation in multi-disciplinary teaching and learning situations.

2. Related Work

Automated timetable generation has been widely studied as a combinatorial optimization problem and several computational techniques have been proposed over the past decades. Early approaches based on heuristic and rule-based methods [5], where scheduling decisions were made using predefined priority rules. Although computationally inexpensive, these techniques aren't changeable and they struggle with having high volumes of connections between constraints in larger organizations. CSP-based models have been studied and utilized extensively [18], [19].

CSP methods view scheduling as a collection of variables, domains and constraints. The use of backtracking and constraint propagation methods to generate solutions for scheduling. However, the performance of CSP methods decreases the number of variables and co-dependencies increases (e.g., in multidisciplinary institutions, multiple schedulers access the same resources). Metaheuristic optimization methods have demonstrated that they outperform pure heuristic methods and typically require a greater degree of parameter tuning while also being less stable when there are changes in dynamic academic policies. For example, simulated annealing models have been developed to provide probabilistic means of leaving from local optima that have a greater development conflict, ant colony-based models use signal tracks to give guidance in making decisions about which regions of the search space to explore [12-15].

Genetic Algorithms have become one of the most frequently utilized and effective approaches to generating timetables, since they have a population-based search advantage, thereby providing a more robust approach to solve multi-constraint optimization problems [1], [4], [10]. The Genetic Algorithm (GA) may be used as a general method of encoding a schedule as a chromosome (i.e., timetable) and subsequently evolve it over time by using the typical GA

operators like selection, crossover and mutation. In general, there have been many studies that compare the use of GA to other conventional methods of heuristics and constraint satisfaction problems (CSP) and all report better solution quality and lower conflict rates across both medium and large datasets when GA is used instead of another method of optimisation. Research has also indicated that hybrid models may further enhance the performance of GA [11]. For example, researchers have started to develop hybrid GAs that combine local search techniques, such as hill climbing or memetic algorithms to enhance the refinement of offspring solutions before replacing existing members of the original parent population. The use of hybrid GAs provides additional exploitation capabilities and refinements of solutions.

However, most existing systems use fixed crossover and mutation rates that limit their ability to adapt based on how many constraints are present (i.e., cluster size and density of constraints) within various datasets. By applying fixed parameters, premature convergence or excessive exploratory behaviour can occur, resulting in inefficient optimisation of GA. Recent attempts by researchers have focused on how the use of machine learning methods can allow for the adaptive control of the parameters used in evolutionary algorithms [17]. Some researchers have researched to use reinforcement learning and regression to dynamically modify the parameters of Genetic Algorithms (GAs) through trends in convergence [8], [9]. These methods have not been widely adopted for the scheduling of educational learners and there are few studies that have been published in the application of GAs for the generation of timetables that consider the various policy-based structural constraints described by the National Education Policy (NEP) 2020. These include credit flexibility, the selection of interdisciplinary courses and the use of a modular academic structure.

Many researchers have focused their efforts on reducing the conflict and optimizing the resource allocation of scheduled learners while not focusing on compliance with government regulations or utilizing adaptive learning as a tool for optimizing education. As a result, applying the credit flexible constraints of NEP 2020 of an Adaptive Genetic Algorithm that uses machine learning as a basis for the generation of educational schedules will satisfy the complexities created by government regulation and allow for improvement of algorithmic performance and thus represent a significant gap in the existing work.

3. Problem Formulation

Assigning various academic activities to limited temporal and spatial resources is systematically achieved by creating timetables within educational institutions. In particular, timetabling will consist of mapping a specific course to an appropriate classroom and time slot while guaranteeing compatibility with the scheduled time and availability of faculty, institutional rule(s) regarding scheduling and academic policy requirements. The connection between courses, infrastructure that is shared between multiple courses and faculty workload, results in an increasing number of potential combinations of timetables as a function of an institution's size causing the timetable generation problem to be considered as a computationally complex NP-Hard problem [2]. A typical medium-scale academic dataset is used for validation of performance measures. For the purpose of this validation, the dataset includes three separate multidisciplinary academic departments, all operating from a credit-based academia structure. Across all disciplines, there are 96 courses that are provided by a total of 54 faculty members. The institution provides a total of 24 classrooms for infrastructure resources and each classroom has a different number of seats. There are a total of 40 discrete time slots for every week, spread evenly across each of five days, forming a total practice schedule for each department [14].

Achieving a balance between usage (underutilized classrooms may have inadequate infrastructure; over-utilized may have excessive infrastructure). Successful schedules will not only optimize the hard constraints involved but will also create a schedule with the highest capacity score representing the overall fulfillment of soft constraints. There are also computational constraints that the university can hold to receive an executed schedule requiring this optimization problem to be solved as a multi-objective optimization problem with feasibility, allocation quality and execution time simultaneously

4. Proposed Methodology

4.2 NEP 2020 Credit-Flexible Constraint Modelling

Through the National Education Policy (NEP) 2020, a new model has been proposed to allow for modifications to course structures as a result of these changes. The new course selection procedures will allow for more interdepartmental flexibility of course scope and availability will provide additional options for students when selecting their courses. The inclusion of interdisciplinary courses provides options not only to students from multiple departments within a school. This provide students with the opportunity to select from skill-based learning will

enhance their educational experience. This new model will provide for better developed laboratory infrastructure for the instruction of skill-based courses and provide an opportunity to award credit hours according to governmental standards established by the Ministry of Education.

As part of this module, a constraint validation component was added into the schedulers that will verify constraints for executing chromosome evaluations. The constraint validation component of the modified system will include an analysis that monitors the availability of unfulfilled elective course time slots, tracking the availability of cross-disciplinary electives to ensure they will suitable on time for students' schedules. The modified system will also monitor the timing of commonly paired electives to ensure that at least one-time slot is available to students.

In addition to ensuring that all credits are allocated according to the NEP, each course must include the same number of lecture, tutorial and laboratory hours as defined by the academic structure. The scheduling engine checks to see if every course has maintained the same amount of total credit hours during a weekly period based on the approved structure (Figure 1).

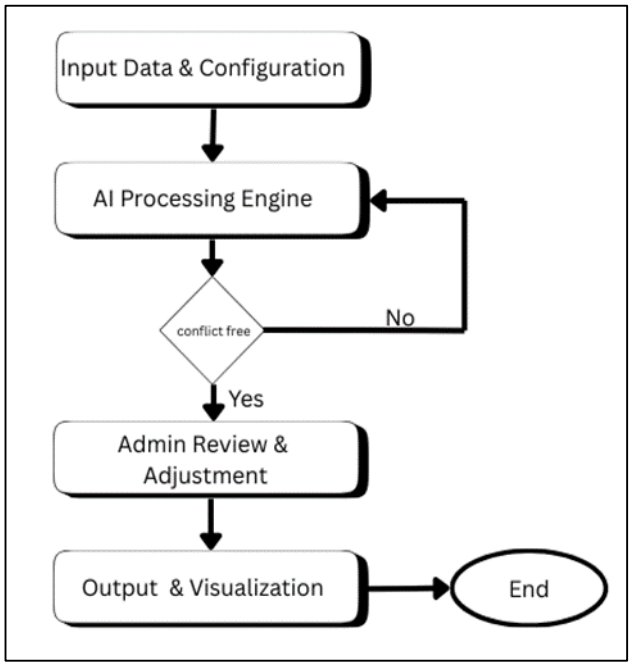

Figure 1. Flowchart for Proposed Methodology

In addition, semester-wise credit accumulation should be evenly distributed and manageable so they can work well with a multi-entry exit system. The balance of the semester can be maintained by distributing credits evenly not to overload the students and keep them all with the same academic load. The integration of these validations into an optimizing loop ensures that

all timetables produced by the system are conflict free and comply with the state. The NEP aware modelling provides a different dimension of timetable generation than has been done in the past.

4.2 Adaptive Genetic Algorithm with ML- Driven Parameter Tuning

Algorithm 1. Adaptive Genetic Algorithm

1. Initialize population /%
2. Evaluate fitness f(x) for all chromosomes
3. For each generation t=1 to G:
    a) Compute convergence indicators
    b) Adapt parameters p_c (t) and p_m (t)
    c) Perform selection
    d) Apply crossover and mutation
    e) Evaluate offspring
    f) Update population
4. Return best solution

Using an efficient search method such as a Genetic Algorithm (Algorithm 1), the timetable optimization algorithm searches a large combinatorial area of possible timetable combinations in a structured manner. Each possible timetable will be encoded as a chromosome with structured mappings of the courses to be taught, faculty to teach the courses, classrooms to conduct the courses and the time slots during which to conduct the courses. Each 'gene' on a chromosome corresponds to a valid scheduling unit and may therefore be efficiently evaluated or modified as part of the evolution operation.

4.3 Chromosome Representation

Each chromosome encodes:

$$x = \{ \text{course, faculty, room, timeslot} \} \quad (1)$$

The initial population of potential solutions in this work was created using a controlled random creation method that conforms to the fundamental constraints of creating valid solutions within a given problem domain. This is done to increase the quality of the initial generations but still keeping the diversity of the population. In traditional GA implementations the crossover and mutation probabilities are determined before the initialization of GA and remain constant

throughout the rest of the evolutionary cycle. However, a fixed parameter setting may result in premature convergence of the population to the optimal solution or exhibit excessive randomness during the continuation phase of the GA process. When an instance has different densities of constraint and objective functions associated with it. The proposed system will incorporate a supervised machine learning algorithm for the adaptive tuning of parameters within the GA.

The machine learning component of this work uses a number of features to monitor the optimization process and will be able to track the optimization process is performing based on how the optimization process behaves relative to the features (i.e. generation number, diversity of solutions or individuals in the population, variance of fitness values, and convergence rate). Population diversity is measured by the structural diversity of the population in addition to the variance of the fitness values calculated for the solutions to the problem generated at that generation. These two metrics represent the degree to which exploration and exploitation of the search space is being balanced by the GA.

The machine learning component of this system will be based on a relatively simple regression-based learning algorithm that has been trained using historical data collected from multiple previous execution of the GA using different parameter settings [8], [9]. The learning process will develop the connections that exist between the various optimization states, the corresponding effective crossover and mutation parameters for that particular optimization state. The learning algorithm will n make use of the trained model to estimate the appropriate parameter settings for each subsequent generation of the current execution of the GA. When diversity decreases significantly, mutation probability is increased to promote exploration. When convergence stabilizes, crossover probability is modified to enhance refinement of high-quality solutions.

The fitness function combines hard and soft constraints:

$$f(x) = -\alpha H(x) + \beta S(x) \tag{2}$$

where:

- $H(x) =$ hard constraint violations
- $S(x) =$ soft constraint satisfaction
- $\alpha \gg \beta$

4.4 Adaptive Parameter Tuning

A. Population Diversity

$$D_t = \frac{1}{N}\sum_{i=1}^{N} \text{dist}(x_i, \bar{x}) \tag{3}$$

B. Fitness Variance

$$\sigma_t^2 = \frac{1}{N}\sum_{i=1}^{N} \left(f(x_i) - \bar{f}\right)^2 \tag{4}$$

C. Convergence Rate

$$r_t = \frac{|f_{\text{hest}}(l) - f_{\text{hest}}(l-1)|}{f_{\text{hest}}(l-1)} \tag{5}$$

D. Adaptive Crossover Probability

$$p_c(t) = p_c^{\min} + \left(p_c^{\max} - p_c^{\min}\right) \cdot \frac{\sigma_t^2}{\sigma_t^2 + k_1} \tag{6}$$

E. Adaptive Mutation Probability

$$p_m(t) = p_m^{\min} + \left(p_m^{\max} - p_m^{\min}\right) \cdot \left(1 - \frac{D_t}{D_{\max}}\right) \tag{7}$$

F. ML-Based Parameter Prediction

$$(p_c(t), p_m(t)) = g_0(D_t, \sigma_t^2, r_t) \tag{8}$$

where g_0 is a trained regression model.

The method of evaluating score combines costs associated with hard constraints with scores (weighted) associated with soft constraints. Specifically, violations of hard constraints experience large costs so that only feasible solutions are rated higher than all other solutions. In addition, soft constraints will be rated according to their importance in providing an improved quality schedule. This approach is designed to balance secure compliance to regulatory requirements while not sacrificing the quality of the scheduled solution. Also, the inclusion of adaptive parameter tuning helps the algorithm converge, minimizes the risk of local optimum and increases the overall efficiency. Therefore, the genetic algorithm (GA) proposed within this research will converge rapidly than traditional GAs that utilize fixed parameters and produce more reliable solutions.

5. Implementation Details

The proposed approach for generating timetable automatically has been developed using a modular, three-tiered architecture which separates the various components of optimization (i.e., optimizing, managing data, and interacting with users). The three-tiered architecture provides the scalability, maintainability and efficiency needed to operate effectively in institutional settings. The primary optimization engine is implemented within Python as it has been shown to have significant support for both scientific computing and machine learning libraries. The Genetic Algorithm framework which includes modules for the initialization of population, selection of individuals for reproduction, the crossover and mutation of offspring individuals and the evaluation of fitness (i.e., a measure of how well an individual satisfies predefined constraints) has been implemented in part on the Python backend [15].

The adaptive parameter tuning capability based on machine learning has been embedded into the same environment as the other components, to facilitate simple interaction between the evolutionary operations that are occurring within the optimization engine and the predictive modifications to parameters that are being generated by machine learning techniques. The adaptive parameter tuning capability has also been implemented using a lightweight regression analysis type of model which minimizes computational overhead while avoiding excessive use of memory during run-time. All of the academic-related data needed to construct timetables (e.g., courses, instructors, classrooms, credits, constraints, and so forth) are stored within a MySQL relational database. The DBMS schema has been designed to represent the connections between the various elements associated with departments, courses, instructor assignments and physical resources in a normalized format. Constraint parameters (e.g., the number of credits required for each course to graduate, the availability of instructors, room capacity, etc.) are dynamically retrieved from the relational database by the optimization engine at run-time.

The ability of data to be accessed dynamically allows for real-time updates to the timetable generation process at an institution without the need for any changes to the algorithms that are used to generate those timetables. A front-end resource that is written in React creates an interactive and responsive method for administrators to enter academic data and set up restrictions on the schedules that create timetable, and. The back-end is supported by RESTful APIs to separate the front-end from the back-end providing secure and efficient data transfer between both parts of the system. The back-end will include an optimization engine that will communicate with a MySQL database to execute command structures and retrieve data from the database's

scheduling data tables to complete the optimization cycle for the newly created timetables. Upon completing the timetabling process, the newly created timetables will be stored in the MySQL database and be available for future reporting, creating additional timetables and recording the effectiveness of each newly developed timetable used in this system during multiple iterations (Figure.2).

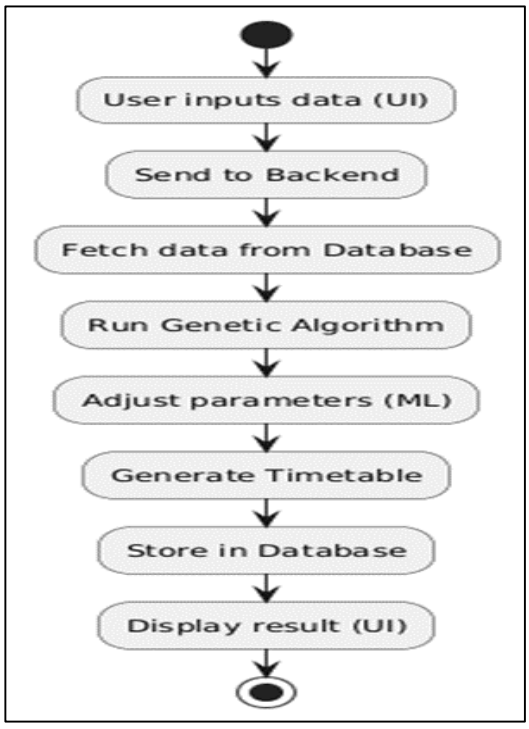

Figure 2. Workflow of the Smart Timetable Scheduling System

Figure 2 illustrating the process from user input through backend processing, genetic algorithm optimization with adaptive parameter tuning and final timetable generation and display. Flexibility at all levels for future growth is developed into the system. Each department can define its own course types, create professor assignments with its preferred methods including its own credit structure while sharing one common system resource. Additionally, because of its modular design, the system can accommodate an increase in data over time without requiring substantial modifications to the system architecture. With the addition of an Adaptive Genetic Algorithm (GA) framework to the scheduling software solution, the system should be able to continue to perform at a high level, that many new and more complex constraints are created. The result of this approach is an integrated scheduling solution for medium-sized (>10K FTE students) colleges and universities that is optimized, compliant with all regulatory agencies has incorporated smart optimization throughout the scheduling process.

6. Results

This proposed system developed a timetable generation application based on an Adaptive Genetic Algorithm (AGA). This application used the following technical aspects like Python programming language for the optimization engine, MySQL database for constraint and academic data storage and React for the user interface. A medium-sized dataset was used to evaluate the performance of the application (approximately 150 courses, 45 faculty members, 28 classrooms, and multiple academic years that comply with NEP 2020 credit flexibility). Once completed, the final generated timetables were made available through an administrative web portal.

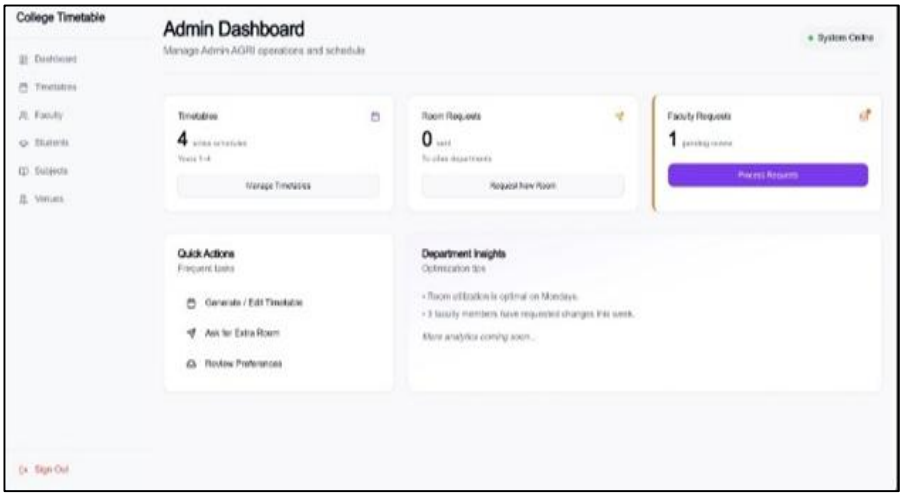

Figure 3. Admin Dashboard

According to the Admin Dashboard shown in figure 3, the administrator's dashboard is the central point for administrating all aspects of academics. It contains the newly created timetable(s) for multiple years, the real-time status of room requests and faculty-requested class schedule changes. The multiple number of active schedules for each year 1-4 shows that the optimization engine has been able to develop and keep the scheduling of multiple semesters at the same time (in real-time). The room request module provides real-time validation of room conflicts identified using constraint-handling methods to manage them properly. it provides better real-time validation of conflicts identified using constraint-handling methods for room requests. The faculty requests have considered other soft constraints (such as preferences for when they would like to teach) or changes to their own personal schedules when developing. It was not violated due to hard constraints, demonstrating the value of using soft and hard constraint recognition to satisfy both needs.

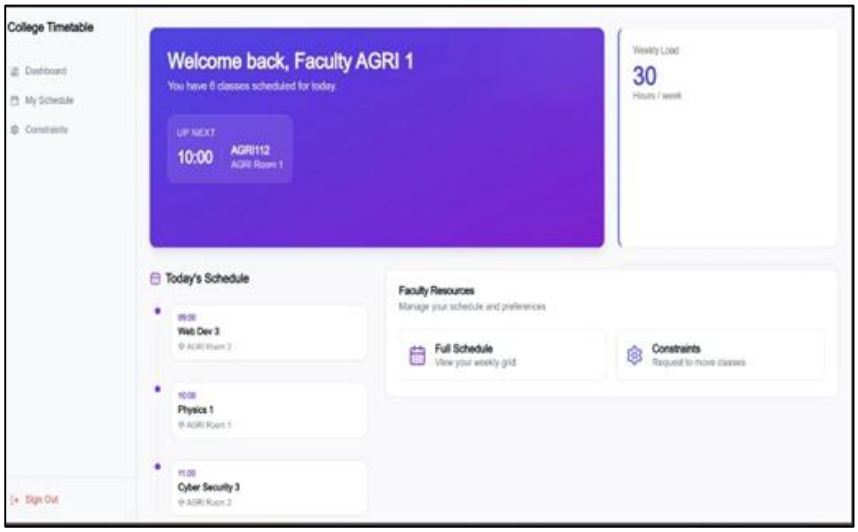

Figure 4. Faculty Dashboard

The Faculty Dashboard screen (Figure 4) shows the user's personalized daily teaching schedule. This shows the current day with all class locations and room allocations, along with the entire teaching week's workload (30 hours/week). This confirms that the generated schedule meets the workload balancing criteria for the user. The constraints option allows faculty to submit requests for revisions, confirming that the system is dynamically flexible. The real-time display of the faculty schedule confirms that properly displays the connection between the solutions generated by GA and the front-end visualization layer.

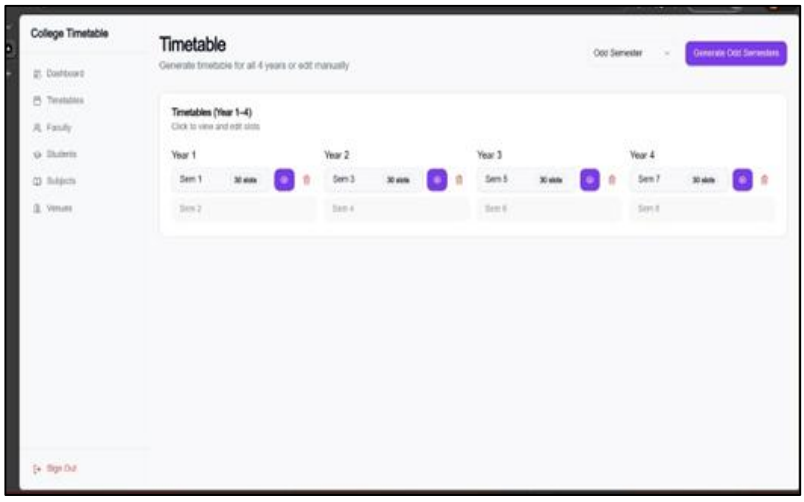

Figure 5. Timetable Generation Interface

Figure 5 (Timetable Generation Interface) represents a timetable divided over all four years of the academic program, for each of the semesters program. All of the semesters will have a total of 30 slots to confirm to the standardized model of time slots used in the optimization framework.

The “Generate Semester” capability provides for validation of whether or not the back-end engine will be able to re-generate a new schedule on-demand. The ability to view and delete the schedule of an individual semester validates that the architecture supports modular regeneration as compared to re-computing possible schedules for the system.

Performance evaluations were performed based upon the following metrics:

- Better fitness score
- Average fitness over all generations
- Execution time
- Failure rate due to constraint violations

The adaptive GA achieved stable convergence of solutions based upon data sets taken from medium scale experiments of 80-120 generations. The average timetable generation time was obtained from running on a standard set of hardware, Intel i5 with 8Gb of RAM produced a generation time of about 12-18 seconds. The final set of solutions contained no hard constraint violations which satisfies the soft constraints was better than generated fixed parameter GA implementations.

Table 1. Performance Comparison

Metric	Traditional GA	Adaptive GA (Proposed)
Best Fitness Score	0.82	0.94
Average Fitness	0.75	0.91
Execution Time (seconds)	22.5	14.2
Generations to Converge	180	95
Hard Constraint Violations	2–5	0
Soft Constraint Satisfaction	0.78	0.92

In Table 1, a comparison is made between results produced by using the standard Genetic Algorithm and Adaptive Genetic Algorithm were executed using identical parameters and the outcomes of the Adaptive Genetic Algorithm performed better than the standard Genetic Algorithm for all of these measures. Adaptive Genetic Algorithms also produce better schedules with fewer hard constraints than traditional Genetic Algorithms.

Table 2. Convergence Analysis

Generation Range	Best Fitness	Average Fitness
1 – 20	0.45	0.32
21 – 40	0.68	0.55
41 – 60	0.82	0.71
61 – 80	0.91	0.85
81 – 100	0.94	0.91

The data in this Table 2 show that the algorithm consistently converged to provide improved fitness values across the generations. In the evolution process of the execution, the performance of the algorithm stabilized at an optimum level within 80–100 generations.

Table 3. Constraint Satisfaction

Constraint Type	**Before Optimization**	**After Optimization**
Faculty Conflict	12	0
Room Conflict	9	0
Credit Violations	6	0
Core Subject Overlap	5	0
Soft Constraint Violations	18	4

This table 3 shows the optimization process worked to fix violations of constraints. All hard constraint violations are removed once the optimization has been applied, while the number of soft constraint violations has been significantly reduced, which shows a better quality and feasibility of the timetable.

Through the conduct of experiments, the evidence suggests that using ML for parameter adjustments has an overall positive effect on performance when solving complex scheduling problems. In addition, utilizing an ML-based approach makes genetic algorithms (GA) more robust by providing a mechanism for dynamically re-balancing exploration and exploitation during high-dimensional search space optimization, such as timetabling.

The probabilities of crossover and mutation that are fixed in traditional approaches to GAs are not ensure to yield the better solution for every stage of evolution. When the evolutionary process states that a considerable quantity of diversity must be present to investigate all possible

solutions. In addition, the adaptive model dynamically modifies the crossover/mutation probabilities based on the characteristics of the convergence of solutions being produced, thus avoiding sudden inactivation and continually improving upon the quality of the solutions being generated.

The explicit modelling of NEP 2020 flexible credit constraints as an in addition to enhancing the algorithm increases the practical value of the approach. Because the size of the dataset develops, the adaptive parameter tuning mechanism will adjust the evolutionary behavior of the GA and allow for expanding the viability of the model to meet the needs of larger educational institutions, while making the structural change to minimum. These two findings confirm the hybrid GA-ML approach to be theoretically valid and effectively validated for use in real-world applications in higher education.

7. Conclusion

This study proposed a genetic algorithm with adaptive mutation and crossover rates implemented through machine learning for tuning parameters to follow the national educational policy's (NEP) requirement for an automated timetable. The proposed approach resolves two of the largest challenges to modern academic scheduling: The complications involved with scheduling due to regulatory requirements, specifically increasing resistance to credit hour changes since multi-disciplinary courses are being created, the limitations and inadequacies associated with the development of static parameter evolutionary algorithms for academic scheduling. Additionally, the proposed system for all course types (core, elective, interdisciplinary and skills-based) with a structured model that follows to NEP 2020 requirements. As new data is provided, the adaptive genetic algorithm will also modify its crossover and mutation rates based on the degree of convergence, making the system more efficient in optimizing and providing a solution for the user. Experimental tests conducted using medium size databases demonstrated that the proposed system produced a more optimized solution with fewer constraint violations than traditional genetic algorithms and significantly decreased execution time. The findings of the tests indicate that adaptive parameter controls enable the stability of evolutionary searches while maintaining compliance with regulations. The system developed can also be used by institutions of higher education as they implement more flexible educational policies. Future research will continue to enhance this framework of optimization through the addition of institutional metrics, the ability to regenerate timetables in real-time based on factors that may

affect a timetable's achievement and the integration of predictive enrolment analytics to improve their ability to create a schedule.

References

[1] Burke, Edmund K., Jakub Mareček, Andrew J. Parkes, and Hana Rudová. "A SuperNnodal Formulation of Vertex Colouring with Applications in Course Timetabling." Annals of Operations Research 179, no. 1 (2010): 105-130.

[2] Abdullah, Salwani, Edmund K. Burke, and Barry Mccollum. "An investigation of variable neighbourhood search for university course timetabling." In The 2nd multidisciplinary international conference on scheduling: theory and applications (MISTA), (2005): 413-427.

[3] Qu, Rong, Edmund K. Burke, Barry McCollum, Liam TG Merlot, and Sau Y. Lee. "A survey of search methodologies and automated system development for examination timetabling." Journal of scheduling 12, no. 1 (2009): 55-89.

[4] Eiben, Agoston E., Jan K. Van Der Hauw, and Jano I. van Hemert. "Graph coloring with adaptive evolutionary algorithms." Journal of Heuristics 4, no. 1 (1998): 25-46.

[5] Carter, Michael W., and Gilbert Laporte. "Recent developments in practical course timetabling." In International conference on the practice and theory of automated timetabling, Berlin, Heidelberg: Springer Berlin Heidelberg, (1997): 3-19.

[6] Colorni, Alberto, Marco Dorigo, and Vittorio Maniezzo. "Metaheuristics for high school timetabling." Computational optimization and applications 9, no. 3 (1998): 275-298.

[7] Eiben, Ágoston E., Robert Hinterding, and Zbigniew Michalewicz. "Parameter control in evolutionary algorithms." IEEE Transactions on evolutionary computation 3, no. 2 (1999): 124-141.

[8] F. G. Lobo, C. F. Lima, and Z. Michalewicz, Parameter Setting in Evolutionary Algorithms, Springer, 2007.

[9] Jin, Yaochu. "A comprehensive survey of fitness approximation in evolutionary computation." Soft computing 9, no. 1 (2005): 3-12.

[10] Zhang, Qiang. "An optimized solution to the course scheduling problem in universities under an improved genetic algorithm." Journal of Intelligent Systems 31, no. 1 (2022): 1065-1073.

[11] Branke, Jürgen, and Hartmut Schmeck. "Designing evolutionary algorithms for dynamic optimization problems." In Advances in evolutionary computing: theory and applications, Berlin, Heidelberg: Springer Berlin Heidelberg, (2003): 239-262.

[12] Tang, Yuanjie, Rengkui Liu, Futian Wang, Quanxin Sun, and Amr A. Kandil. "Scheduling optimization of linear schedule with constraint programming." Computer-Aided Civil and Infrastructure Engineering 33, no. 2 (2018): 124-151.

[13] Al-Betar, Mohammed Azmi, Ahamad Tajudin Khader, and Munir Zaman. "University course timetabling using a hybrid harmony search metaheuristic algorithm." IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 42, no. 5 (2012): 664-681.

[14] Cevallos, Fabian, and Fang Zhao. "Minimizing transfer times in public transit network with genetic algorithm." Transportation Research Record 1971, no. 1 (2006): 74-79.

[15] Shabbir, Attia, Raja Hashim Ali, Muhammad Zeeshan Shabbir, Zain Ul Abideen, Talha Ali Khan, Ali Zeeshan Ijaz, Nisar Ali, Muhammad Imad, and Muhammad Abu Bakar. "Genetic algorithm-based feature selection for accurate breast cancer classification." In 2023 International Conference on IT and Industrial Technologies (ICIT), IEEE, (2023): 1-6.

[16] Abbas, Shaheer, Ahmad Maaz, Raja Hashim Ali, Talha Ali Khan, and Iftikhar Ahmed. "Automatic timetable generation using neural networks trained by genetic algorithms." In 2024 18th International Conference on Open Source Systems and Technologies (ICOSST), IEEE, (2024): 1-6.

[17] Solotorevsky, Gadi, Ehud Gudes, and A. Meisels. "Algorithms for Solving Distributed Constraint Satisfaction Problems (DCSPs)." In AIPS, (1996): 191-198.

[18] Yokoo, Makoto. "Asynchronous weak-commitment search for solving distributed constraint satisfaction problems." In International Conference on Principles and Practice

of Constraint Programming, Berlin, Heidelberg: Springer Berlin Heidelberg, (1995): 88-102.

[19] Neiman, Daniel E., and Victor R. Lesser. "A Cooperative Repair Method for a Distributed Scheduling System." In AIPS, (1996): 166-173.