Article

Not peer-reviewed version

# Optimizing the Scheduling of Teaching Activities in a Faculty

Francis Patrick Diallo and Cătălin Tudose *

Posted Date: 10 September 2024

doi: 10.20944/preprints202409.0732.v1

Keywords: automatic timetable generation; faculty scheduling; metaheuristics algorithms; resource utilization; scheduling conflicts; scalability; Timefold

*Article*

# Optimizing the Scheduling of Teaching Activities in a Faculty

**Francis Patrick Diallo [1,2] and Cătălin Tudose [1,2,*]**

[1] Faculty of Automatic Control and Computers, National University of Science and Technology POLITEHNICA Bucharest, 060042 Bucharest, Romania; francis.diallo@stud.acs.upb.ro
[2] Luxoft Romania, 020335 Bucharest, Romania
* Correspondence: catalin.tudose@gmail.com

**Abstract:** To maximize resource usage, minimize disputes, and improve academic experience, professors must schedule teaching activities efficiently. This study provides an optimized automated schedule creation technique. The system generates conflict-free, efficient schedules using evolutionary algorithms and multi-objective optimization. Resource usage, scheduling problems, and faculty/student satisfaction are the goals of the research. The system optimizes scheduling based on room limitations, instructor availability, and student preferences. The project uses system design, model creation, algorithm implementation, and performance analysis to solve the difficult timetable-generating problem. This research should save administrators time, improve academic operations, and improve staff and student academic experiences. Scalability and flexibility allow the system to be used in multiple faculties and incorporate new limits and requirements. This paper presents a complete approach to faculty scheduling, including insights and recommendations for future study and application in educational institutions.

**Keywords:** automatic timetable generation; faculty scheduling; metaheuristics algorithms; resource utilization; scheduling conflicts; scalability; Timefold

## 1. Introduction

During an academic year at a university, a significant amount of time and human resources are dedicated to the manual search for a solution that adheres to all established regulations and satisfies the expectations of all involved parties. The scheduling of courses and exams in higher education institutions is a complex issue that raises significant challenges for universities worldwide. Universities must ensure optimal utilization of their finite resources by both their students and staff. Currently, all academic institutions need to reduce operational expenses as an ongoing component of everyday reality. It seems unreasonable for institutions to persist with manually developing timetables when they can potentially be automated and optimized.

The goal of this research is to develop and execute an automated system capable of producing schedules for academic activities within a specific faculty. The process of generating a timetable entails the allocation of courses, faculty members, and rooms to designated time slots in a manner that is both efficient and devoid of conflicts. The objective of the research is to enhance the efficiency of the faculty by optimizing the scheduling of teaching activities through automation. This will result in improved resource utilization, reduced conflicts, and streamlined processes.

### *1.1. Context*

The efficient scheduling of teaching activities in a faculty plays a crucial role in ensuring a well-organized and effective educational environment, particularly in faculties with many courses/seminars/laboratories, faculty members, and students. The problem of optimizing didactic activities is inspired by the well-known Timetable problem [1]. The manual generation of timetables is time-consuming, error-prone, and often results in suboptimal schedules. This project seeks to

 doi:10.20944/preprints202409.0732.v1



address these challenges by leveraging advanced algorithms and technologies to automate the timetable generation process.

### *1.2. Motivation*

The intricate process of organizing educational activities is a multifaceted undertaking that pertains to the wider scope of scheduling concerns, which possess extensive implications across diverse fields. The process of creating schedules, such as those for universities, schools, healthcare facilities, transportation, sports events, and entertainment, poses distinct difficulties concerning the allocation of resources, limitations, and goals [2, 3]. From a computational standpoint, these issues are categorized as nondeterministic polynomial time complete (NP-complete), which implies that there are no known efficient techniques for identifying optimal solutions [4]. Hence, heuristic methodologies are frequently utilized to ascertain efficient resolutions. The objective of this study is to create an automated system that enhances the scheduling of educational tasks within a university setting while taking into account the unique difficulties and limitations that are inherent to this field. This approach aims to overcome the drawbacks of manual timetable creation.

Conventional approaches necessitate substantial time and effort from administrators and faculty members, making them labor-intensive. Additionally, the resultant schedules may not optimally allocate resources, may give rise to scheduling clashes, and may not conform to the preferences and limitations of the interested parties. The implementation of an automated timetable generation system can address the aforementioned concerns by facilitating optimal resource allocation, minimizing conflicts, and enhancing satisfaction levels among both students and faculty.

### *1.3. Objectives*

The main aim is to create and execute a resilient system that automates the process of generating timetables for the faculty. This will be achieved through the development of an Automated Timetable Generation System. The system will take into account multiple constraints, preferences, and optimization criteria to generate timetables of superior quality.

The project endeavors to enhance the utilization of resources, including time slots, faculty members, and rooms. The optimization of these resources is a key objective. Through the strategic allocation of courses and activities, the system will optimize the utilization of existing resources, mitigating potential conflicts and enhancing the overall capacity of the academic staff.

The system will strive to reduce scheduling conflicts, including but not limited to instances of classes overlapping, faculty members being assigned to multiple classes concurrently, and students experiencing timetable clashes. The system will produce timetables that are free of conflicts and offer a well-structured and coordinated schedule for all parties involved, taking into account multiple constraints and interdependencies.

At last, the objective of the project is to elevate the contentment of both faculty and students by devising schedules that cater to their preferences and needs. The generation of timetables that cater to the needs of all stakeholders, including faculty availability, student course preferences, and other constraints, can lead to a more favorable and efficient academic environment.

### *1.4. Expected Impact*

The anticipated benefits of implementing an automated timetable generation system within the faculty are manifold and significant:

- The implementation of automated timetable generation will result in noteworthy reductions in the temporal and laborious demands placed on both administrative personnel and faculty members. The process of manually creating timetables can consume a significant amount of resources. By automating these tasks, administrators can allocate their time to other strategic activities.

 doi:10.20944/preprints202409.0732.v1



- The implementation of the system is expected to optimize the efficiency of academic operations in the faculty. The optimization of timetables can lead to increased efficiency in resource utilization, reduced scheduling conflicts, and streamlined administrative processes.
- The implementation of an automated timetable generation system will enhance the academic experience by promoting a better work-life balance for faculty members. This will be achieved by aligning their schedules with their preferences and workload constraints. By implementing this approach, individuals can optimize their time management, effectively schedule research or administrative duties, and prevent scheduling conflicts. Consequently, the reduction of stress levels and the increase in job satisfaction among faculty members will result in heightened productivity and an enhanced quality of teaching.
- The automated timetable generation system offers a notable benefit in terms of enhancing the efficient utilization of resources. The implementation of a system that efficiently allocates courses, faculty members, and rooms can effectively mitigate inefficiencies and minimize occurrences of underutilization or overbooking. The optimization of available resources without the requirement of supplementary investments will result in cost savings for the faculty. The system will facilitate the allocation of classrooms and facilities in a manner that reduces the likelihood of conflicts and promotes a cohesive and structured learning environment.
- The development of an automated system for generating timetables is intended to be flexible and versatile, capable of accommodating the evolving requirements of the academic staff. As the academic staff expands or novel courses are implemented, the system can readily integrate supplementary limitations and inclinations. The ability of the system to produce optimized timetables despite an increase in the size and complexity of the faculty is attributed to its scalability.

Furthermore, the system will possess the capability to accommodate diverse scheduling scenarios and academic prerequisites. The system is designed to facilitate the implementation of diverse timetabling policies, including semester-based or modular-based scheduling, and can effectively manage a wide range of academic activities, such as lectures, seminars, laboratory sessions, and exams. The system's adaptability enables it to accommodate the varied requirements of various faculties or departments within the institution.

## 2. Related Work

Over time, educational institutions around the globe have worked extensively on the deployment of ever-more-effective algorithms to alleviate the administrative portion that occurs annually with technological advancement, and numerous scientific articles have been created based on these technical achievements.

For instance, Denzil D'souza et. all [5] used the Genetic Algorithm meta-heuristic algorithm, which is based on Darwin's theory of evolution, and parameters including Course code, number of hours for each course, teacher code, class code, room code, day, and time; to generate a timetable in a matrix form. In this study, the authors did not emphasize limitations or the implementation of a functioning platform, but they did illustrate the computing performance of this approach.

Research conducted at the University of Technology Malaysia [6] highlighted the constraints that can reduce the performance of the optimization algorithm to generate a stable schedule, as well as proposed an algorithm that is based on the pre-processing of the matrix to be processed and the use of the Neighborhood Search Algorithm, which is based on a mathematical function.

D. Wungguli and N.Nurwan [7] created a course scheduling optimization model using Integer Linear Programming [8] to reduce lecturers' and students' dissatisfaction. This purpose is achieved

 doi:10.20944/preprints202409.0732.v1



by the model's objective function, which seeks to minimize the amount of student and lecturer rejection of the schedule based on the formulation of the equation's time option.

X.Feng, Y.Lee, and I.Moon [9] developed an algorithm based on a combination of their own mathematical model and the Hybrid Genetic Algorithm after comparing the performance and parameters of other optimization algorithms.

A.Schaerf [10] provided a comprehensive overview of the different types of timetable optimization problems, as well as a summary of the most frequent methods used to solve them. One of the methods is Direct heuristics, which involves filling the full schedule with a single course (or one set of courses) at a given moment if there are no scheduling conflicts. The approaches employed in this problem include reduction to graph coloring, network flow problem, and simulated annealing. "Simulated annealing is a probabilistic method" [11] used to identify the global minimum or maximum of a cost function which might include several local minimums or maximums. "Tabu search is a local search method" [12] that is widely recognized as a powerful tool for tackling challenging optimization issues. On the other hand, the rule-based approach consists of five sorts of rules: allocation rules, restriction rules, local change rules, and contextual rules.

Krzysztof Socha, Michael Sampels, and Max Manfrin [13] studied several variants of the Ant Colony Optimization technique that was applied to a set of predefined restrictions.

An additional approach provided by A. Abuhamdah and M. Ayob [14] is the use of a computational scheduling system. MPCA-ARDA is the hybridization of the Multi-Neighborhood Particle Collision Algorithm [15] with the Adaptive Randomized Descent Algorithm acceptance criteria [16]. It was developed to address problems related to organizing courses at universities. This method was studied and compared to the algorithms in their simplest form, and it was shown to be more effective.

Shrunkhala et. all [17] used Genetic Algorithm to create a platform that involves a dashboard that students can use to navigate the educational institute's platform, Job allocation. Ong Boon Chun [18] created a platform using PHP, Laravel, and SQL to generate a schedule for students and professors automatically.

Hoda Atef Yekta and Robert Day [19] focused mainly on applying a combination of algorithms, such as matching algorithms, second-price concepts, and improvement, to promote truth-telling among agents with limited logic. They compared these algorithms based on metrics of equity, efficacy, and incentive integration to evaluate their effectiveness.

Rayna Burgess [20] examined the utilization of OptaPlanner [21], an open-source constraint solver and predecessor to Timefold Solver [22], to create an automated course scheduling system for the Computer Science Department at California State University. The study focuses on the complexity of manual scheduling by utilizing advanced algorithms to optimize course assignments, ensure balanced faculty workloads, and meet student requirements. The results indicate significant improvements in the effectiveness and efficacy of scheduling, highlighting the potential of OptaPlanner in handling difficult scheduling problems in academic settings with a smaller dataset (70 courses, 50 instructors, 20 rooms, and 60 timeslots).

Fiechter [23] tackles the complex problem of generating university timetables that comply with various constraints, such as room boundaries, teacher availability, and course prerequisites. By applying the algorithms of the Optaplaner API, Fiechter successfully generated timetables that not only satisfied all hard restrictions but also optimized multiple soft constraints, resulting in increased satisfaction among both teachers and students. The issue instance examined by Fiechter consisted of a small dataset covering three semesters. It involved scheduling 14 courses and 70 lectures within a school week that consisted of 40 periods (with 8 time slots each day) and had 8 accessible rooms. The resulting schedules were generated within approximately 7 seconds and were found to be comparable to the existing timetables currently being utilized for the semester. This exemplifies the efficacy and pragmatic utility of Fiechter's technique in real-life contexts.

However, none of the aforementioned studies have explored or addressed the complexities involved in scenarios with a significantly larger scale, such as those involving thousands of courses, teachers, student groups, rooms, and time slots. The existing research primarily focuses on smaller

 doi:10.20944/preprints202409.0732.v1



datasets, with relatively limited numbers of variables, which might not capture the full spectrum of challenges faced by large educational institutions. In contrast, the present work undertakes the challenge of optimizing timetables for extensive datasets, addressing a broader range of constraints, and ensuring the scalability of the solution. This effort marks a significant advancement in the application of algorithmic approaches to timetable generation, catering to the needs of institutions with substantial student and faculty populations.

## 3. Model of the Automatic Timetable Generation System

### *3.1. System Architecture*

A web-based platform is proposed for the development of an Automatic Timetable Generation system for a faculty. The frontend of the system will be constructed using the Angular framework, while the backend will be developed using Java with Spring Boot. The aforementioned architectural approach guarantees a design that is both scalable and modular, effectively segregating the presentation layer from the business logic and data processing layers as shown in the figure below (Figure 1). The proposed system is designed to adopt a client-server architecture, wherein the Angular-based frontend will establish communication with the Java-based backend via APIs.

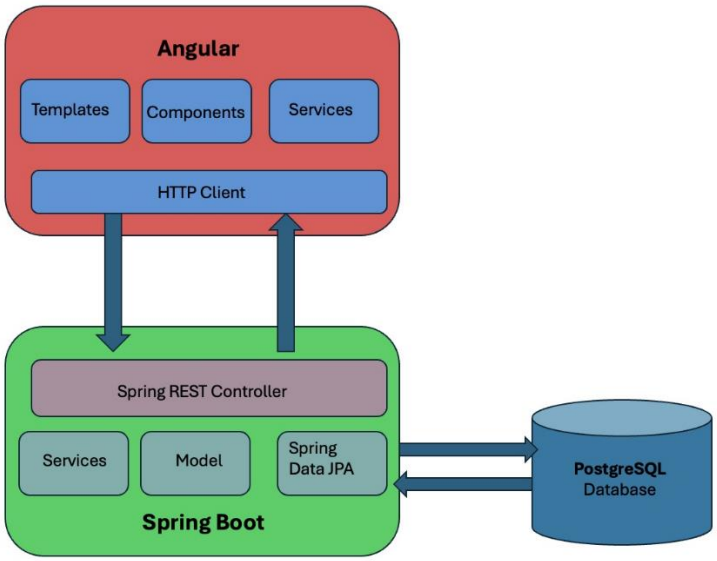


**Figure 1.** Application Architecture.

### *3.2. Frontend Architecture*

The frontend of the web platform will be developed using Angular [24], an effective framework for building single-page applications. Angular has a comprehensive range of features and tools that streamline the creation of dynamic and adaptable web apps. The system follows a component-based architectural design, which encourages the structured development of various features and functionalities in a modular manner.

The primary attributes and capabilities of Angular that can assist in the development of web-based applications include the following:

- Component-Based Architecture: Angular's architecture is based on a component-based approach, which focuses on creating reusable components. Every component contains its own HTML, CSS [25], and TypeScript [26], which helps maintain a clear distinction between different aspects of the application and facilitates its management and scalability.
- Data Binding: Angular offers robust data binding features, covering both one-way and two-way data binding. This enables smooth synchronization between the model and the view, guaranteeing that the user interface is consistently updated with the underlying data.

 doi:10.20944/preprints202409.0732.v1



- Dependency Injection: Angular's inherent dependency injection architecture improves modularity and testability. Components can be readily injected with services and other dependencies, resulting in code that is more manageable and facilitates testing.
- Routing: Angular's router module facilitates the development of a multi-page user interface within a single-page application. Developers can use it to specify routes and transition between several views while preserving the application's state and ensuring a seamless user experience.
- Reactive Programming: Angular leverages RxJS to enable reactive programming, allowing developers to efficiently manage asynchronous data streams and events.
- Forms and Validation: Angular offers powerful form-handling features, including template-driven and reactive forms, enabling effective form management and validation. The presence of built-in validators and the option to develop custom validators assist in guaranteeing the integrity of data and validating user input.
- Testing: Angular provides extensive support for unit testing and end-to-end testing using tools such as Jasmine [27] and Protractor [28], guaranteeing the reliability and bug-free nature of applications.

The proposed application will consist of several modules, including user authentication, course selection, scheduling preferences, and timetable display. These modules will interact with the backend application programming interfaces (APIs) to retrieve data and update the application's status, guaranteeing a user interface that is dynamic and responsive.

### *3.3. Backend Architecture*

The system's backend will be implemented using the Java programming language [29, 30] and the Spring Boot framework [31]. These technologies are known for their dependable and efficient infrastructure in developing server-side applications. Spring Boot is a robust framework that simplifies the development of software applications that are set up for production. It achieves this by offering convention over configuration, pre-built functionalities, and a seamless interface with a range of tools and libraries.

Some features and capabilities of Spring Boot are the following:

- Auto-Configuration: Spring Boot's auto-configuration feature streamlines the application setup process by automatically setting Spring and third-party libraries according to the project's dependencies. This reduces the need for redundant code and configuration.
- Embedded Servers: Spring Boot incorporates embedded servers such as Tomcat, and Jetty enabling programmers to execute their applications as independent executables without requiring a separate application server.
- Microservices Support: Spring Boot is highly compatible with the construction of microservices. It offers several capabilities such as Spring Cloud integration, service discovery, configuration management, and circuit breakers. These characteristics facilitate the creation of distributed systems that are both scalable and resilient.
- RESTful APIs: Spring Boot accelerates the process of creating RESTful APIs [32] with Spring MVC [33]. Annotations like *@RestController* and *@RequestMapping* streamline the process of creating API endpoints that may be accessed by the frontend.
- Security: Spring Security [34] smoothly integrates with Spring Boot, providing extensive security functionalities including authentication, authorization, and protection against prevalent vulnerabilities such as CSRF and XSS.
- Data Access: Spring Data JPA [35] streamlines database interactions by offering a repository abstraction on top of JPA (Java Persistence API). Developers can use it to execute CRUD activities and intricate queries with minimum boilerplate code [36].
- Testing: Spring Boot provides comprehensive testing capabilities, including JUnit for unit testing and Spring's testing framework for integration testing [37]. These features guarantee the dependability and accuracy of the application.
- Spring IoC: Spring IoC, also known as Inversion of Control, is a crucial component of the Spring Framework [38]. It plays a vital role in overseeing the life cycle and configuration of objects within an application and it is achieved by employing dependency injection. This mechanism enables objects to specify their dependencies via constructor arguments, factory methods, or

 doi:10.20944/preprints202409.0732.v1



properties. The Spring container is then responsible for creating and providing these dependencies.

The suggested backend architecture is designed to follow the principles of a layered architecture [39], consisting of separate layers including the presentation layer, service layer, and data access layer.

- The Presentation Layer is responsible for managing API endpoints and operating as the intermediary between the backend and the frontend. The main objective of this layer is to receive incoming HTTP requests, execute the required actions on them, and provide appropriate responses.
- The Service Layer is responsible for managing the business logic and operational procedures involved in the creation and optimization of timetables. It functions as a mediator between the display layer and the data access layer, guaranteeing the consistent application of business rules.
- The Data Access Layer is responsible for handling interactions with the database and ensuring the long-term storage of relevant data. Spring Data JPA [35] is utilized to abstract the fundamental database activities, simplifying the process of switching databases if necessary.

### *3.4. Timefold Solver Optimization Algorithm*

The Timefold Solver [22] serves as the central component of the timetable generation method, utilizing its advanced capabilities to efficiently distribute resources and time slots. Timefold Solver is an artificial intelligence planning solver implemented in the Java programming language. Timefold enhances the efficiency of planning problems, such as the vehicle routing problem (VRP) [40], maintenance scheduling [41], job shop scheduling [42], and school timetabling [1]. It develops logistics strategies that significantly minimize expenses, enhance the quality of service, and reduce the environmental impact, frequently by up to 25%, for intricate, practical scheduling activities.

Timefold is the subsequent development of OptaPlanner [21]. It is a type of mathematical optimization that is used in the fields of Operations Research and Artificial Intelligence. It allows for the inclusion of constraints written in code. The Timefold Solver utilizes Plain Old Java Objects (POJOs) to implement the input and output data, which consist of the planning challenge and the optimal solution. The Timefold Solver provides three types of optimization algorithms: Construction Heuristics, Exhaustive Search, and Metaheuristics. The Timefold Solver integrates optimization algorithms, such as metaheuristics and heuristics, with a score calculation engine (Figure 2). This combination is highly effective for the following reasons:

- A score computation engine is a powerful instrument that efficiently calculates the rating of a solution to a scheduling problem. It streamlines and enables the incorporation of additional hard or soft constraints in a scalable manner. The program does progressive score calculation, commonly referred to as deltas, without needing additional coding. Nevertheless, it is usually inappropriate to find new solutions.
- An optimization algorithm excels at discovering enhanced solutions for a planning challenge, without the need to exhaustively explore every possible option. However, it cannot efficiently calculate the score of a solution and does not provide any support in doing so.

 doi:10.20944/preprints202409.0732.v1



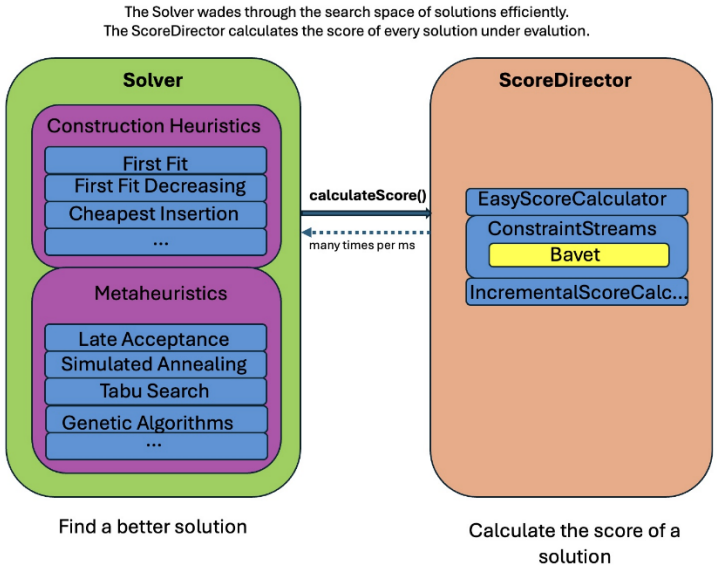


**Figure 2.** Timefold Solver Architecture Overview.

Generally, the scheduling problem has at least two distinct levels of restrictions that are important in ensuring that the timetable satisfies the requirements and preferences of students, teachers, and the entire organization.

- A hard constraint, whether negative or not, must not be violated. For instance, a teacher is unable to instruct two distinct lessons simultaneously.
- A negative soft constraint should be respected wherever possible. For instance, Teacher A has a dislike of teaching classes on Friday afternoons.
- Certain problems also have positive limitations.
- If feasible, it is desirable to fulfill a positive soft restriction (or incentive). For instance, Teacher X specifically prefers teaching lessons on Monday mornings.

The N queens problem [43] and similar classical problems are defined by the existence of exclusively hard constraints. Some problems demonstrate various tiers of restrictions, including hard, medium, and soft constraints. The determination of the score computation (fitness function) for a scheduling problem is based on these constraints. Every solution to a planning challenge can be assessed by assigning a score using an object-oriented programming language.

The Timefold Solver incorporates multiple optimization techniques to effectively navigate the vast number of potential solutions. The performance of optimization algorithms varies depending on the scenario, but it is not possible to predict their performance in advance. Timefold's versatility enables seamless switching between algorithms, depending on the particular use case, by modifying the configuration in XML or code.

A construction heuristic efficiently generates a high-quality starting solution within a limited period. Although its answer may not always be feasible, it can quickly locate it, allowing metaheuristics to complete the task.

Construction heuristics have an automatic termination, therefore there is generally no requirement to set up a specific termination for the construction heuristic step. Several construction heuristics algorithms are supported such as:

- The First Fit algorithm sequentially iterates over all planning entities, initializing one at a time, according to their default sequence. The process allocates the planning entity to the most optimal planning value, considering the previously initiated planning entities. The termination occurs when all planned entities are fully initialized. Once a planning entity has been designated, it remains unchanged.
- The First Fit Decreasing algorithm is a variant of the First Fit algorithm, but it prioritizes assigning the more challenging planning entities first, as they are less inclined to be

 doi:10.20944/preprints202409.0732.v1



accommodated in the remaining space. Therefore, it arranges the planning entities in descending order of difficulty.

- The Weakest Fit algorithm is a variant of the First Fit algorithm, but it prioritizes using weaker planning values initially. This is because the stronger planning values have a higher probability of being able to support subsequent planning entities. Therefore, it arranges the planning values in ascending order of their strength.
- The Strongest Fit algorithm is similar to the First Fit algorithm, the Strongest Fit algorithm prioritizes the adoption of planning variables that have a higher likelihood of having a decreased soft cost. Therefore, it arranges the planning values in descending order of strength.

A local search begins with an initial solution and iteratively improves upon that solution, gradually achieving increasingly superior solutions. It uses a solitary search path of solutions, rather than a search tree. During every iteration of this sequence, it evaluates a set of moves on the current solution and executes the most appropriate move to advance toward the next solution. The process persists through many iterations until it is halted, typically due to reaching its time limit.

Local search functions similarly to a human planner by employing a singular search path and restructuring data to discover an optimal and obtainable solution. Thus, it is quite straightforward to implement. Timefold incorporates various Local Search algorithms, including Hill climbing [44], Tabu Search [44], Late acceptance [46], Great Deluge [47], and Simulated Annealing [48].

### *3.5. Constraints*

Developing an efficient schedule for a faculty requires careful consideration of several constraints [49], as they are essential in ensuring that the timetable satisfies the requirements and preferences of students, teachers, and the institution as a whole.

*Types of Constraints*

*Hard Constraints*

During the timetable-generating process, hard constraints are rigid regulations that must be followed precisely. These constraints are important for preserving the essential structure and feasibility of the timetable. The hard constraints consist of:

- Courses must be organized according to the specified time slots provided by the faculty, ensuring the timing is suitable.
- The room capacity should be sufficient to accommodate the number of students enrolled in each subject.
- A teacher is unable to simultaneously teach two distinct lessons in separate classrooms.
- It is not possible for a room to accommodate two distinct lessons simultaneously.
- The scheduling of lessons should not have any overlapping.
- A lesson with a duration of $x$ cannot be scheduled for a time slot with a different duration than $x$.
- A student group cannot attend 2 different lessons at the same time.

*Medium Constraints*

Medium constraints are positioned between hard and soft constraints in terms of their level of flexibility and relevance. While medium constraints offer greater flexibility compared to hard constraints, it is essential to adhere to them to generate a timetable that can be both useful and efficient. The medium constraints include the following:

- It is imperative to provide a fair allocation of teaching hours among faculty members to avoid stressing certain teachers while underutilizing others and also take into consideration the availability of faculty members who are assigned to teach them. This approach is important for sustaining faculty satisfaction and minimizing burnout.
- Courses and seminars should be organized together in the same time slot and room, either for a whole series of students (in the case of courses) or for a specific group of students (in the case of seminars).
- It is advisable to spread out the courses evenly across the week to prevent the concentration of too many lessons on a single day. This helps to minimize exhaustion for both students and professors and also ensures efficient utilization of classroom space.

 doi:10.20944/preprints202409.0732.v1



- The maximum amount of hours of courses per day for a student group should not exceed 10 hours.
- It is important to ensure that students' schedules are evenly distributed throughout the week to facilitate efficient learning and mitigate the risk of exhaustion. This entails the avoidance of scheduling back-to-back classes and maintaining sufficient intervals between sessions.
- A teacher's workload should not exceed 12 hours of courses each day.
- Courses should be arranged based on the availability of teachers during specific time slots for teaching.

*Soft Constraints*

Soft restrictions are preferences or recommendations that are desired to meet but can be eased if needed. These limits enhance the overall quality and satisfaction of the timetable but are not essential to its fundamental practicality. The soft limitations encompass the following:

- Teachers have a preference for instructing in a singular place to minimize the necessity of moving between several classrooms.
- Students exhibit a preference for having courses located in the same building to minimize the duration and effort required for transitioning between classes.
- It is important to avoid intervals greater than 4 hours between two consecutive courses, as these prolonged breaks might cause disruptions to students' schedules.
- It is advisable to arrange laboratory courses at the same time and in the same location for student groups to ensure consistency.
- Teachers prioritize maintaining a continuous and efficient teaching schedule by avoiding gaps between their classes.

### *3.6. Database Considerations*

The storage of pertinent information, including course details, student inclinations, faculty accessibility, and timetable arrangements, mandates the utilization of a database. The recommended database solution for the proposed system architecture, which consists of an Angular-based frontend and a Java-based backend powered by Spring Boot, is a Relational Database Management System (RDBMS). An option that is suitable and will be used in the implementation of this scenario is PostgreSQL [50].

PostgreSQL is a resilient and feature-rich open-source relational database management system (RDBMS). The system exhibits robust data integrity and concurrency control mechanisms, rendering it a fitting solution for managing concurrent requests originating from both the Angular frontend and Java backend.

Key features and advantages include:

- **Resilience:** PostgreSQL is acknowledged for its robustness and ability to endure challenges. This resilience ensures data consistency and reliability, crucial for managing complex timetabling data.
- **Sophisticated Functionalities:** The system's capability to accommodate sophisticated functionalities aligns seamlessly with the intricate nature of university timetabling. From managing course details to handling concurrent requests, PostgreSQL stands as a versatile and capable solution.
- **Data Integrity:** PostgreSQL places a premium on data integrity, ensuring that information remains accurate and consistent. This aspect is pivotal when dealing with diverse datasets and intricate relationships within the timetabling context.
- **Concurrency Control:** The system's prowess in managing concurrent requests is particularly relevant in the context of the timetabling system, where multiple users interact with the database simultaneously. PostgreSQL's concurrency control mechanisms contribute to a seamless and reliable user experience.

### *3.7. Integration and Communication*

The integration and communication aspects are an essential element. To facilitate seamless communication between the frontend and backend, the system will employ RESTful APIs. The

 doi:10.20944/preprints202409.0732.v1



backend system will provide a series of clearly defined endpoints that the Angular frontend can utilize to retrieve and submit data. The exchange of information between the frontend and backend will rely on the HTTP protocol, thereby guaranteeing the ability of various system components to work together and function properly. A workflow of the application is presented in Figure 3.

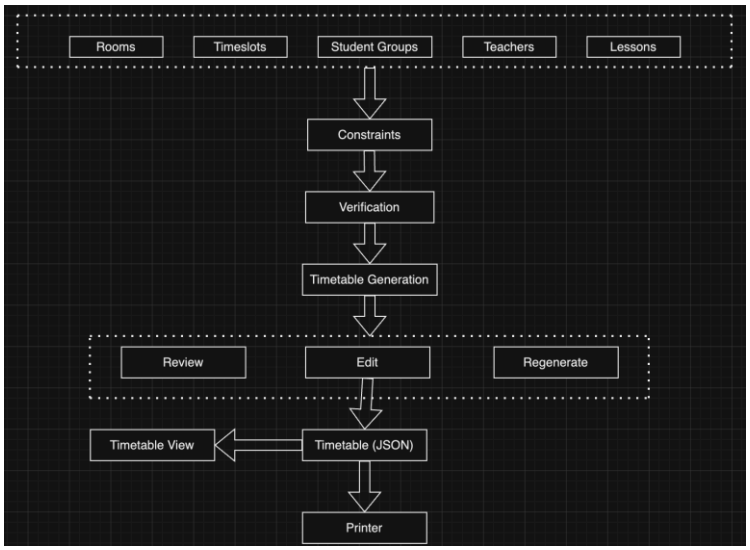


**Figure 3.** Proposed Workflow Architecture.

### 3.8. *Enhancing System Integrity*

The topic of concern in this section is the protection of privacy and data security.

Preserving data security and privacy is essential in any academic setting. The system must incorporate suitable measures to safeguard confidential information, including but not limited to student records and faculty data. This entails the deployment of robust authentication and authorization protocols, encryption of data both in transit and at rest, and strict adherence to established data privacy standards.

In the ongoing pursuit of fortifying the "Automatic Timetable Generator for Faculty" project, the paramount focus extends beyond the foundational aspects of data security and privacy. This section delves into the meticulous considerations and implementation strategies to ensure not only the safeguarding of sensitive information but also the reliability, security, and availability of the entire system.

1. Data Security and Privacy

A key component of this academic-focused system's operation and design is the preservation of data confidentiality and privacy. The system's purview encompasses safeguarding confidential information, ranging from student records to faculty data. To fortify the integrity of the data, the following measures are diligently employed:

- **Robust Authentication and Authorization Protocols:** Implementing a robust set of authentication and authorization protocols ensures that access to sensitive information is restricted to authorized individuals. This includes the meticulous use of role-based access controls, where users are granted specific permissions based on their roles within the academic hierarchy.
- **Hashing of Data:** Sensitive information undergoes hashing procedures, a cryptographic technique that converts data into irreversible, fixed-length strings of characters. Hashing ensures that the original data cannot be reconstructed from the hash values, providing an additional layer of protection against unauthorized access and potential data breaches [51].
- **Adherence to Data Privacy Standards:** Striving for excellence, the system adheres steadfastly to established data privacy standards. This commitment ensures that the collection, storage, and utilization of personal data strictly comply with legal and ethical guidelines, fostering a secure and trustworthy environment for all stakeholders.

 doi:10.20944/preprints202409.0732.v1



2. Reliability: The Timefold Solver Library

Reliability is a paramount aspect of the system, especially in the context of timetable generation. To achieve unparalleled reliability, the project leverages the robust capabilities of the Timefold Solver library. Key attributes include:

- **Algorithmic Precision:** The Timefold Solver library employs cutting-edge algorithms, rigorously tested and validated with diverse datasets and scenarios. This ensures the generation of accurate and conflict-free timetables, contributing significantly to the reliability of the overall system.
- **Error Mitigation:** With a focus on minimizing conflicts and rectifying errors, the Timefold Solver library incorporates extensive error detection and handling mechanisms. This ensures that unexpected issues are identified promptly, preventing disruptions in the timetable generation process.

3. Security: Role Guards and JWT Authentication

Security is paramount in safeguarding the system against unauthorized access and potential breaches. The following security measures are implemented to fortify the system:

- **Role Guards:** Role guards [52] play a pivotal role in ensuring that only authorized users with specific roles or permissions can access designated parts of the application. This measure enhances data and system security, preventing unauthorized individuals from tampering with critical information.
- **JWT Authentication and Authorization:** The system employs JSON Web Token (JWT) [53] authentication, a secure and efficient method for verifying the identity of users. This, coupled with robust authorization mechanisms, ensures that only authenticated and authorized users can interact with the system, minimizing the risk of unauthorized access.

## 4. Research Methodology

To address the research objectives and develop an effective solution for automatic timetable generation in a faculty or college, a structured research methodology will be followed. This chapter outlines the methodology that will guide the research process, including the steps involved in data collection, system design, implementation, and evaluation.

### *4.1. Problem Analysis and Requirements Gathering*

The research will begin with a thorough analysis of the existing timetabling process and the specific challenges faced by the faculty or college. This will involve collecting relevant information from faculty administrators, staff members, and students. The objective is to identify the key requirements, constraints, and preferences that need to be considered in the automated timetable generation system.

### *4.2. Literature Review*

An extensive literature analysis will be performed to evaluate existing approaches, algorithms, and techniques used in automatic timetable generation. This will involve studying research papers, conference proceedings, and relevant publications in the field of timetabling and optimization. The literature review will provide a solid foundation for understanding the state-of-the-art methods and identifying potential gaps in the existing solutions, as presented in the Related Work chapter.

### *4.3. System Design*

Based on the requirements gathered and the insights gained from the literature review, the research will focus on designing the system architecture, algorithms, and data models for automatic timetable generation. This will involve identifying the appropriate algorithms and techniques that can effectively address the specific challenges of the faculty or college. The system design will aim to optimize resource allocation, minimize conflicts, and satisfy the various constraints and preferences identified in the problem analysis.

 doi:10.20944/preprints202409.0732.v1



*4.4. System Implementation*

Once the system design is finalized, the research will proceed with the implementation phase. The web platform will be developed using Angular for the frontend and Java with Spring Boot for the backend, as mentioned earlier. The algorithms and techniques selected during the system design phase will be implemented, and the necessary APIs and database connections will be established. This phase will also involve testing and debugging to ensure the correctness and functionality of the system.

*4.5. Data Collection and Integration*

To generate effective timetables, relevant data will be collected and integrated into the system. This may include course information, faculty availability, student/teacher preferences, and any additional constraints specific to the faculty or college. Data collection can involve surveys, interviews, or data extraction from existing systems. The collected data will be processed and integrated into the system to provide accurate inputs for the timetable generation process.

*4.6. Evaluation and Performance Analysis*

Once the system is implemented and the data is integrated, an evaluation process will be conducted to assess the performance and effectiveness of the automated timetable generation system. This will involve generating timetables based on different scenarios and real-world data and comparing the results against predefined metrics and objectives. The evaluation will focus on factors such as timetable quality, resource utilization, and user satisfaction. Performance analysis will also be conducted to measure the system's responsiveness, scalability, and efficiency.

*4.7. Iterative Refinement*

Based on the evaluation results, feedback from stakeholders, and insights gained during the implementation and evaluation phases, the system will undergo iterative refinements. This may involve fine-tuning the algorithms, enhancing the user interface, and addressing any identified limitations or shortcomings. The iterative refinement process will continue until a satisfactory level of performance and user acceptance is achieved.

*4.8. Ethical Considerations*

Throughout the research process, ethical considerations will be given utmost importance. The privacy and security of student and faculty data will be ensured, and informed consent will be obtained for data collection. Any potential biases or unintended consequences of the automated timetable generation system will be carefully analyzed and mitigated. The research will adhere to ethical guidelines and regulations to maintain the integrity and welfare of the participants.

## 5. The Development of Theoretical/Practical Contributions

The development of an "Automatic Timetable Generator for Faculty" represents a pivotal advancement in the realm of academic management systems. This section delves into the intricate details of the application's implementation, elucidating the theoretical underpinnings and practical functionalities that characterize this innovative tool.

*5.1. Comprehensive Approach to Automated Timetable Generation*

The process of developing theoretical and practical contributions for automated schedule production commenced with a comprehensive review of the present timetabling method at University Politehnica Bucharest. This entailed in-depth consultations with the faculty member in charge of manual timetable creation, along with thorough literature analysis and system design.

*Problem Analysis*

 doi:10.20944/preprints202409.0732.v1



During the initial consultations with the faculty member, important features of the timetabling procedure were identified.

- **Student Hierarchy:** Students are categorized into series, which are then separated into groups, and these groupings can be further divided into semigroups. Classes are conducted for complete series, seminars for groups, and laboratory sessions for either groups or subgroups.
- **Room and Time Constraints:** Lessons must be assigned to rooms that have suitable capacity to handle the student groups. In addition, students are limited to a maximum of 10 hours of coursework every day.
- **Scheduling Complexity:** The interplay of these constraints leads to a high level of complexity in scheduling, requiring the allocation of many lessons into available rooms and timeslots without any clashes. The timetable generation problem includes 153 rooms with different capacities and located in different buildings, 195 timeslots (from Monday to Friday), 18 constraints of hard, medium, or soft levels, 166 teachers, 4187 lessons, and 338 students' groups (including Bachelor and Master students). This is the highest complexity, compared with other papers, research articles, and implementations.

*Literature Overview*

An extensive literature assessment was undertaken to investigate the strategies, algorithms, and techniques employed in the automated generation of timetables. In the areas of scheduling and optimization, research papers and related publications were reviewed. Although there have been several studies on evolutionary algorithms, constraint satisfaction, and heuristic approaches, there is a clear lack of research addressing the specific difficulties of scheduling university timetables with a high volume of classes, as observed at University Politehnica Bucharest.

*Data Collection and Integration*

Data related to lessons, student groups, rooms, teachers, timeslots, and limits was gathered from the online timetable accessible on the Politehnica University of Bucharest website. The data underwent processing and integration into the system to ensure precise inputs for the development of the timetable:

- **Lessons:** Detailed information about courses, seminars, and laboratory sessions.
- **Student Groups:** Organization of series, groups, and semigroups.
- **Rooms:** Name, capacity, and location details.
- **Teachers:** Name and availability.
- **Timeslots:** Defined periods for scheduling lessons.
- **Constraints:** Rules for maximum hours per day, room capacity, lesson type allocation, and the basic constraints that exist in the university timetable college.

*Ethical Considerations*

Given that the data utilized for timetable construction was publicly available online, ethical considerations were not a major consideration in this research. Therefore, matters concerning privacy and informed consent were not applicable.

*Database Model for Timetable Generation*

An effective and organized database model is essential for the efficient creation of timetables. The database should be structured to accommodate diverse entities, including student groups, teachers, classes, rooms, limitations, time slots, and users (both administrators and regular users).

The primary entities in the database model consist of *StudentGroup*, *Teacher*, *Lesson*, *Room*, *ConstraintModel*, *Timeslot*, and *User*. Every entity possesses distinct attributes and establishes connections with other entities, which are essential for the effective operation of the timetable generation system.

(1) StudentGroup

Attributes:

- id (Primary Key): Unique identifier for each student group.
- year: year of the student group
- name: Name of the series of student group (e.g., AA1, AB3).
- studentGroup (group): Group to which the students belong.
- semiGroup: Semigroup to which the students belong (Semigroup 1, Semigroup 2, or Semigroup 0 for the Master students).

 doi:10.20944/preprints202409.0732.v1



- numberOfStudents: Number of students in the group.

Relationships:

- A student group can be linked to multiple lessons.
- A student group is associated with specific constraints.

(2) Teacher

Attributes:

- id (Primary Key): Unique identifier for each teacher.
- name: Full name of the teacher.
- timeslots: Time slots during which the teacher is available to teach.

Relationships:

- A teacher can be linked to multiple lessons.
- A teacher is associated with specific constraints.

(3) Lesson

Attributes:

- id (Primary Key): Unique identifier for each lesson.
- subject: Lesson's name
- teacherId (Foreign Key): Identifier of the teacher conducting the lesson.
- studentGroupId (Foreign Key): Identifier of the student group attending the lesson.
- lessonType: Type of lesson (e.g., Course, Seminar, Laboratory).
- year: The year for which the lesson is held.
- duration: Duration of the lesson.
- timeslotId (Foreign Key): Identifier of the timeslot where the lesson is assigned.
- roomId (Foreign Key): Identifier of the room where the lesson is held.

Relationships:

- A lesson is linked to a specific teacher and student group.
- A lesson must be scheduled in an appropriate room and time slot.

(4) Room

Attributes:

- id (Primary Key): Unique identifier for each room.
- name: Name or number of the room.
- capacity: Maximum number of students the room can accommodate.
- building: Physical location of the room.

Relationships:

- A room can host multiple lessons.
- A room is associated with specific constraints.

(5) ConstraintModel

Attributes:

- id (Primary Key): Unique identifier for each constraint.
- description: Description of the constraint.
- weight: Type of constraint (e.g., Hard, Medium, Soft).

Relationships:

- Constraints are linked to student groups, teachers, rooms, and lessons.
- Constraints must be considered during timetable generation to ensure compliance.

(6) Timeslot

Attributes:

- id (Primary Key): Unique identifier for each time slot.
- day: Day of the week (e.g., Monday, Tuesday).
- startTime: Start time of the time slot.
- endTime: End time of the time slot.

Relationships:

- A time slot can be assigned to multiple lessons.
- Time slots must align with teacher availability.

(7) User

Attributes:

- id (Primary Key): Unique identifier for each user.

 doi:10.20944/preprints202409.0732.v1



- email: Contact email of the user.
- password: Hashed password for the user account.
- role: Role of the user (e.g., Admin, Normal User).

Relationships:

- Users can manage and access timetable data according to their role.
- Admin users have additional permissions for configuring constraints and settings or for generating the timetable.

To better understand the relationships between these entities, an Entity-Relationship (ER) diagram can be constructed as can be seen in Figure 4.

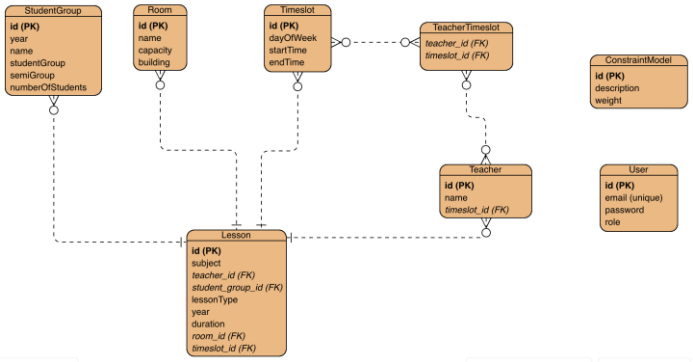


**Figure 4.** ER Diagram.

*5.2. Functionalities of the Application*

In the pursuit of optimizing academic scheduling, an effective Automatic Timetable Generator for Faculty should encapsulate multifaceted functionalities:

- **Course and Activity Management:** The application should allow administrators to define and manage courses and various academic activities such as courses, seminars, laboratories, and exams. It should provide options to specify activity durations, resource requirements, and any constraints associated with each activity.
- **Faculty and Staff Management:** The application should support the management of faculty members and staff by allowing administrators to input their availability, preferences, and workload constraints. This information will be used to generate optimized timetables that consider individual preferences and constraints.
- **Room and Resource Management:** The application should facilitate the management of rooms and resources available within the faculty. It should allow administrators to define room capacities, availability, and any specific requirements for activities. This information will be used to allocate appropriate rooms and resources during the timetable generation process.
- **Timetable Generation:** The core functionality of the application is to generate automated timetables based on the input data provided. It should employ optimization algorithms, such as genetic algorithms or other suitable techniques, to create conflict-free and optimized timetables while considering various constraints and preferences.
- **Conflict Resolution:** The application should have mechanisms to identify and resolve conflicts that may arise during the timetable generation process. Conflicts could include overlapping activities, resource constraints, faculty availability conflicts, or student preferences that cannot be accommodated. The system should provide options for resolving these conflicts based on predefined rules or user-defined priorities.
- **Timetable Visualization and Editing:** The generated timetables should be presented in a user-friendly and intuitive manner. The application should provide visualization tools to display timetables, allowing users to easily view and navigate through the schedules. Additionally, there should be options for manual editing or adjustments to the generated timetables, if necessary.

*5.3. Use Case UML Diagram*

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 10 September 2024 doi:10.20944/preprints202409.0732.v1



The Use Case Diagram in UML stands as a visual representation of the functional requirements and interactions within the system, offering a comprehensive view from a user's perspective. It identifies the different actors (users, systems, external entities) and the various use cases (functionalities or actions) they perform within the system. It is noteworthy that there are three distinct categories of users within the system: administrators, teachers, and students. The administrator is responsible for constructing the schedule by accessing the designated platform, inputting pertinent data to facilitate schedule generation, making necessary modifications, and reviewing the resultant timetable. Conversely, students and teachers are limited to viewing the schedule without the ability to make changes. The use case diagram can be found in Figure 5.

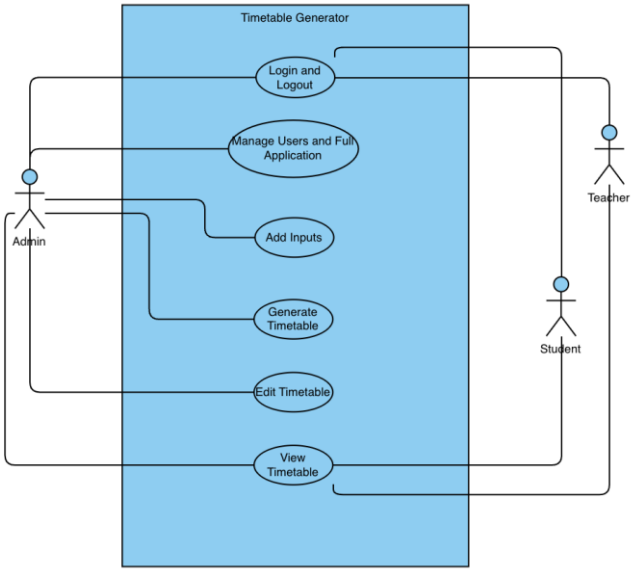


**Figure 5.** Use Case Diagram.

*5.4. Implementation Details*

*User Authentication and Authorization*

The key component of the application is based on the meticulous implementation of user authorization and authentication, with a focus on ensuring data security and controlling access. Utilizing the BCrypt hashing algorithm for password storage is a good choice for protecting user credentials. The adaptive hashing technique used by BCrypt, along with the addition of salting, effectively reduces the probability of password-related vulnerabilities. Every user in the database has a password that has been hashed and salted, which provides a safe basis for user authentication, as demonstrated in Figure 6.

| | id [PK] integer | email character varying (255) | password character varying (255) | role character varying (20) |
|---|---|---|---|---|
| 1 | 1 | admin@gmail.com | $2a$12$ySaIIfWLimnqVTf/.dIWRuwpVhsjBO1nV3fNTKolfpgLhidD6Hp... | ADMIN |
| 2 | 2 | user@gmail.com | $2a$12$ySaIIfWLimnqVTf/.dIWRuwpVhsjBO1nV3fNTKolfpgLhidD6Hp... | USER |

**Figure 6.** Users' database with hashed password.

Upon initiating the login process, the Spring Security module orchestrates a sophisticated authentication mechanism. A comprehensive security filter chain evaluates user credentials against the stored data in the database. Successful authentication triggers a seamless redirection to the application dashboard, where the user's assigned role—either ADMIN or USER—determines the extent of their access privileges. The authentication process extends to the generation of a JWT token (Figure 7), providing a time-limited, secure session for users.

 doi:10.20944/preprints202409.0732.v1



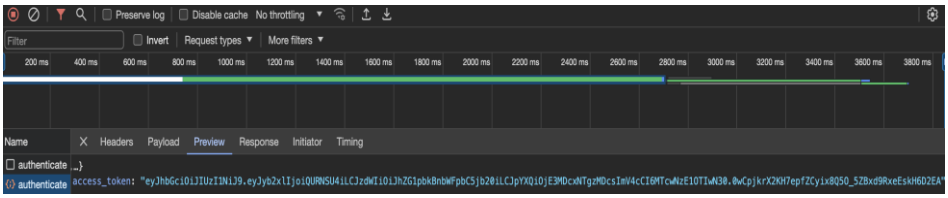


**Figure 7.** Successful authentication with JWT token response.

*CRUD Operations with Spring JPA*

The dynamic nature of the application necessitates the seamless handling of Create, Read, Update, and Delete (CRUD) operations. Leveraging the capabilities of Spring JPA, the system adeptly manages essential data entities required for timetable generation. These entities encompass teachers, students, classrooms, courses, and restrictions, among others. The implementation ensures a streamlined and extensible approach to handling the core data components that fuel the timetable generation process.

Angular guards play a pivotal role in maintaining a secure and role-based access environment. These guards thoroughly examine user permissions and roles, ensuring that only authorized users are granted the ability to access the specific pages. Unauthorized attempts to access restricted pages result in automatic redirection to the login page, thereby enhancing the overall security posture of the application.

*Timefold for Timetable Generation*

The main feature of the application is centered around the advanced integration of Timefold for generating timetables. Timefold, a powerful constraint satisfaction solver, simplifies the difficult task of generating university timetables. The approach utilizes optimization algorithms, effectively integrating Java annotations and configurations with Spring Boot.

Key Annotations and Configurations:

- *@PlanningEntity*: Marks classes as entities manipulated by Timefold to identify an optimal solution. Entities include activities, teachers, and classrooms.
- *@PlanningVariable*: Identifies variables subject to change during optimization, such as time slots and rooms for each activity.
- *@PlanningSolution*: Designates the class representing the entire problem to solve, with the timetable serving as the planning solution.
- *@SolverFactory*: Configures and initializes the solver, allowing for custom settings such as optimization algorithms and termination conditions.
- *@PlanningScore*: Identifies the attribute representing the score or fitness of the solution, guiding Timefold in refining the timetable.
- *@ConstraintConfigurationProvider*: Holds the configured constraint weights and other constraint parameters.
- *@PlanningEntityCollectionProperty*: Timefold Solver can change the annotated object during solving and the *ConstraintProvider* (where the constraints are implemented) can select from those too.
- *@ProblemFactCollectionProperty*: The *ConstraintProvider* can select from those annotated object instances to assign to a *@PlanningEntity* object.

The integration with Timefold represents a pinnacle achievement in the application, providing a powerful and adaptive solution for efficient university timetable generation. Figure 8 represents the utilized domain model for the timetable generation with Timefold annotations.

 doi:10.20944/preprints202409.0732.v1



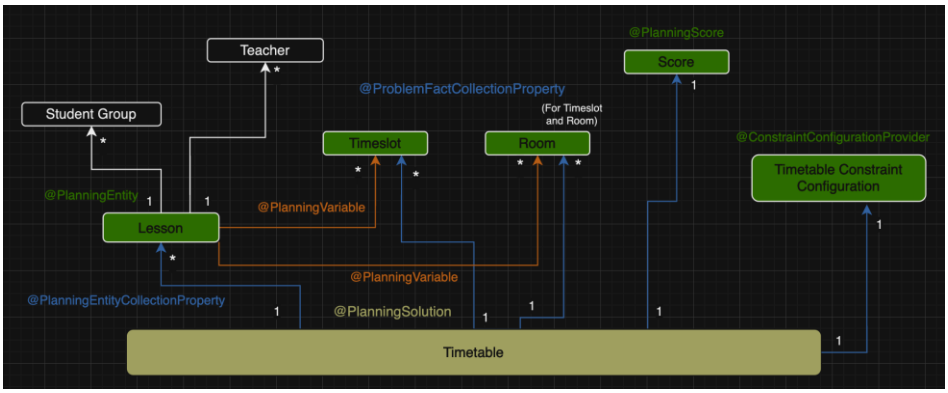


**Figure 8.** Timefold Domain Model.

In the development of the automatic timetable generation system, the following Timefold Solver configuration was used to ensure efficient and optimized scheduling.

(1) Timefold Configuration

- *InitializingScoreTrend*: This parameter defines the trend of the score during the initialization phase, while the previously initialized variables remain unchanged. Some optimization algorithms, including Construction Heuristics, can achieve better execution times when provided with such information.
- *MoveThreadCount*: This parameter is set to *AUTO*, which allows Timefold to automatically determine the optimal number of threads for performing move evaluations in parallel. This enhances the solver's performance by leveraging multi-core processors, thereby reducing the computational time required for generating optimized timetables

(2) Algorithms Used

- Construction Heuristic Phase: Strongest Fit Decreasing
  In the Construction Heuristic phase, a generic form of the Strongest Fit Decreasing (SFD) algorithm was used. This algorithm works as follows:
  - Sorting: It sorts the lessons based on their 'strength' or importance, such as their duration and it also sorts the timeslots and rooms by their importance, such as interval between start time and end time in the case of timeslot, and capacity in the case of room.
  - Assignment: It then assigns the most 'difficult' lessons (strongest fit) to the most suitable time slots and rooms first, ensuring that the hardest parts of the problem are tackled initially.

This approach helps in creating a feasible initial solution by focusing on the most challenging aspects of the timetable, thereby laying a strong foundation for further optimization.

To refine the initial solution and avoid getting stuck in local optima, two advanced local search algorithms were employed:

- Hill Climbing Late Acceptance:
  - Hill Climbing: This method iteratively improves the solution by making small changes (moves) that lead to better scores.
  - Late Acceptance: To avoid the algorithm from being trapped in local optima, Late Acceptance allows the acceptance of worse solutions if they are better than a solution encountered several steps earlier. This adds a level of flexibility and helps the solver escape local minima.
- Tabu Search:
  - Memory Structure: Tabu Search uses a memory structure to keep track of recently visited solutions and marks them as 'tabu' (forbidden) for a certain number of iterations. This prevents the solver from cycling back to previously visited solutions and encourages the exploration of new areas in the solution space.
  - Aspiration Criteria: Despite being marked as 'tabu', certain moves can be accepted if they meet specific criteria (aspiration criteria), such as leading to a significantly better solution.

 doi:10.20944/preprints202409.0732.v1



The combination of these algorithms was selected to balance the need for finding a feasible initial solution and performing thorough optimization:

- Strongest Fit Decreasing: Ensures a robust starting point by addressing the most challenging constraints first.
- Hill Climbing Late Acceptance: Provides a mechanism to explore the solution space effectively, allowing for temporary acceptance of suboptimal solutions to escape local optima.
- Tabu Search: Further enhances the search process by maintaining diversity in the solutions explored and preventing cycles.

These algorithms work together to ensure that the timetable generation process is both efficient and effective for real-world scenarios, producing high-quality solutions that meet the diverse constraints and preferences of the stakeholders.

(3) Timefold Constraints

Timefold restrictions are implemented by creating a network composed of nodes. The nodes representing the sources are the *forEach()* nodes, while the nodes representing the sinks are the penalties. The solver sequentially traverses all problem facts and entities across this graph according to a predetermined order. The limitations are broken down into a network consisting of a single node, which leads to penalties in the conclusion. This methodology is similar to the RETE algorithm, a commonly employed method in rule-based systems for pattern matching.

The RETE algorithm, created by Charles L. Forgy [54], is a very efficient algorithm for pattern matching that is used to design production rule systems. Its purpose is to efficiently compare a vast number of rules with an extensive collection of facts or data. The fundamental concept behind the RETE algorithm is to minimize the number of comparisons needed by sharing common conditions among many rules and storing partial matches to prevent repetitive evaluations. Below is an elaborate description of the functioning of the RETE algorithm:

- Network Construction: The RETE algorithm builds a network of nodes, consisting of alpha nodes and beta nodes. Alpha nodes evaluate basic conditions on particular facts. The entering facts are filtered depending on attribute values, such as verifying the availability of an instructor at a specific time. Beta nodes execute join operations on several facts, merging partial matches obtained from alpha nodes. They are accountable for managing intricate situations that involve several factors, such as ensuring that no teacher is scheduled for two classes simultaneously.
- Fact Propagation: Facts are distributed throughout the network, originating from the alpha nodes. Every alpha node assesses its condition based on the incoming facts and transmits the corresponding facts to the beta nodes.
- Partial Match Storage: Beta nodes store partial matches in memory, which are intermediate results of the join operations. This storage allows the RETE algorithm to avoid re-evaluating the same conditions repeatedly, significantly improving efficiency.
- Rule Activation: The rule is activated when all the conditions of the rule are satisfied by a collection of facts, and this activation occurs at the final node in the network. The RETE algorithm effectively oversees the working memory, guaranteeing that only pertinent modifications initiate rule assessments.
- Efficiency: The efficiency of the RETE algorithm comes from its ability to share condition evaluations among different rules and store intermediate results. This reduces the number of comparisons needed, especially in systems with a large number of rules and facts.

Upon executing the algorithm and applying the constraints, the frontend component returns a JSON-formatted response. This response is then utilized to display the data in the university timetable.

*Security Measures: Front-end Interceptors and Backend Filters*

Augmenting the security architecture, both frontend interceptors in Angular and a robust security filter in the backend using Spring Security are implemented.

Angular interceptors play a crucial role in securing communication between the frontend and backend. These interceptors facilitate the inclusion of the user's token in HTTP requests, ensuring secure and authenticated interactions. This mechanism safeguards against potential threats and unauthorized access, contributing to a resilient and secure application.

 doi:10.20944/preprints202409.0732.v1



A security filter in the backend, integrated with Spring Security's *AuthenticationProvider*, analyzes each incoming request. This filter serves as the last line of defense, rigorously verifying the validity of tokens and user credentials for every request. Unauthorized attempts trigger appropriate responses, reinforcing the integrity of the system.

The combination of these security measures fortifies the application, creating a robust shield against potential security threats. The implementation ensures that data integrity, confidentiality, and availability are uncompromised, establishing a secure and dependable platform for university timetable generation.

## 6. Performance Analysis Measure

The evaluation of the effectiveness and efficiency of the automatic timetable generation system in a faculty or college is contingent upon the critical aspect of performance analysis. The present chapter centers on the utilization of performance analysis measures to evaluate diverse facets of the system's performance. Through the analysis of these metrics, valuable insights can be obtained to optimize the system and improve its overall efficiency.

### *6.1. Timetable Quality Metrics*

The assessment of timetable quality metrics pertains to the degree to which the generated timetables conform to the stipulated requirements and constraints of the faculty or college. The subsequent metrics may be taken into account:

- The metric of conflict resolution evaluates the frequency of conflicts or discrepancies in the schedule, such as the occurrence of lectures or exams that overlap. A reduced frequency of conflicts is indicative of a superior-quality timetable. Timefold Solver provides a score calculation functionality that indicates the quality of the solution within a specified time.
- Resource utilization refers to the measurement of the efficiency with which the available resources, including but not limited to classrooms and faculty, are being utilized. The process assesses the distribution of resources to reduce instances of either underutilization or overutilization.
- The metric of "balanced workload" prioritizes the equitable distribution of workload among faculty members. The algorithm takes into account various parameters, including the number of instructional hours, intervals between courses, and successive instructional hours assigned to individual faculty members.

### *6.2. Computational Performance Metrics*

The automatic timetable generation system is evaluated based on its efficiency and scalability through the use of computational performance metrics. The subsequent metrics may be employed:

- The metric of execution time pertains to the duration required by the system to produce a schedule. A reduced execution time is a desirable outcome, as it signifies a more expedited timetable generation process.
- Scalability refers to the system's ability to maintain optimal performance levels as the scope and complexity of the problem being addressed increase. The evaluation assesses the system's capacity to process extensive datasets and produces schedules within acceptable temporal constraints.
- The evaluation of memory usage pertains to the memory demands of the system while generating the timetable. This aids in the detection of any suboptimal memory utilization or possible performance limitations.

### *6.3. User Satisfaction Surveys*

The utilization of user satisfaction surveys is a valuable approach to gaining insights into the usability and user experience of the automatic timetable generation system. One potential approach to collecting feedback on the system's usability, functionality, and overall satisfaction is to administer surveys to faculty members, students, and administrators.

 doi:10.20944/preprints202409.0732.v1



### *6.4. Comparison with Manual Timetabling*

A comparative analysis between the automated system and the manual timetabling process can be conducted to assess the efficiency of the former. This may encompass various indicators, such as gains in efficiency, decreases in interpersonal frictions, and enhancements in the allocation of assets. The act of juxtaposing the efficiency of the mechanized system against the manual procedure serves to illustrate the merits and gains of the suggested resolution.

### *6.5. Real-World Testing and Validation*

The process of real-world evaluation and verification entails the execution of the automatic timetable generation system within an authentic academic setting, such as Politehnica University of Bucharest. This facilitates the assessment of the system's efficacy in practical scenarios and facilitates the acquisition of constructive input from users. The performance of the system can be evaluated in high-demand scenarios by conducting tests during peak scheduling periods.

## 7. Results

The university timetable generation application was executed using two different Apple laptops, the MacBook Pro M1 (2021) and the MacBook Air M1 (2020), each with specific hardware and software configurations to evaluate the algorithm's performance and effectiveness.

The MacBook Pro M1 (2021) utilized for this study is equipped with a 10-core CPU, a 16-core GPU, 3,2 GHz, 16GB unified memory, and a 1TB SSD. The operating system used was macOS Monterey. The frontend was developed using Angular 16 and TypeScript 4.9.4, while the backend was implemented with Java 21, Spring Boot 3.2.3, and PostgreSQL 14.10 for the database. The solver employed was Timefold Solver 1.9.0, though the current version at the time of writing was 1.10.0.

Similarly, the MacBook Air M1 (2020) used in the experiments featured an 8-core CPU, a 7-core GPU, 3,2 GHz, 8GB of unified memory, and a 512GB SSD, running on macOS Monterey. The software stack for the frontend and backend was identical to that of the MacBook Pro, using Angular 16, TypeScript 4.9.4, Java 21, Spring Boot 3.2.3, and PostgreSQL 14.10 with Timefold Solver 1.9.0.

The timetable generation algorithm was tested twice on the MacBook Pro M1, each with different durations to assess performance and effectiveness. In the first run, which lasted approximately 8 hours, the best score obtained was -262init/0hard/-2113medium/7647soft (Figure 9). This result indicates that there were no violations of hard constraints, 2113 violations of medium constraints, and a score of 7647 for soft constraints, with 262 rooms or timeslots unassigned to any lesson. The algorithm ran for 28,800,043 milliseconds, achieving a score calculation speed of 8401 scores per second, utilizing 4 threads for processing.

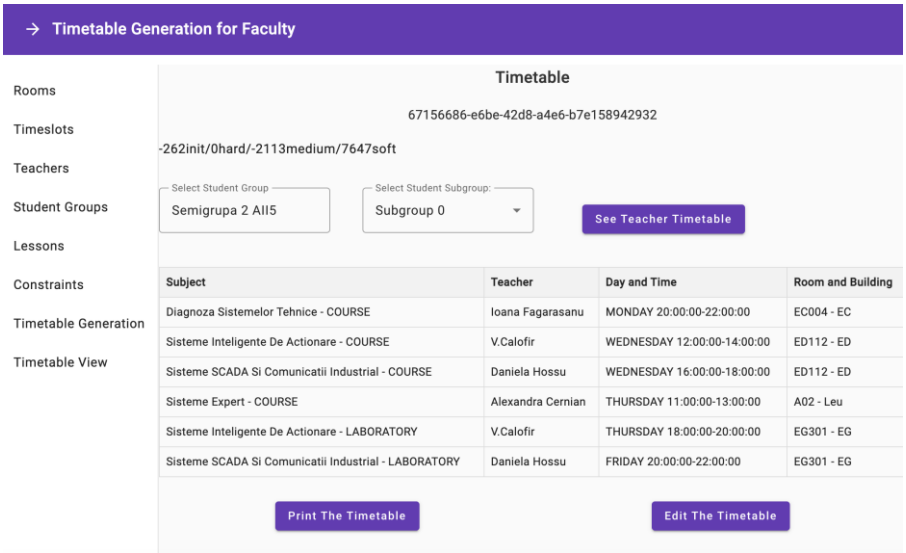


**Figure 9.** Results of the first run of the algorithm.

 doi:10.20944/preprints202409.0732.v1



In the second run, extending to 12 hours, the Construction Heuristic phase successfully initialized the best feasible solution after 8 hours and 37 minutes with a score of 0hard/-2187medium/7889soft. For the remaining hours, the Local Search phases made incremental improvements, culminating in a final score of 0hard/-2183medium/8121soft (Figure 10). As with the first run, there were no violations of hard constraints and all the rooms and timeslots were assigned to lessons, but the number of medium constraint violations decreased to 2183. The score for soft constraints improved to 8121. The algorithm ran for 43,200,005 milliseconds, with an overall score calculation speed of 6342 scores per second, and averaged 1991 scores per second during the local search phases. This run also used 4 threads for processing.

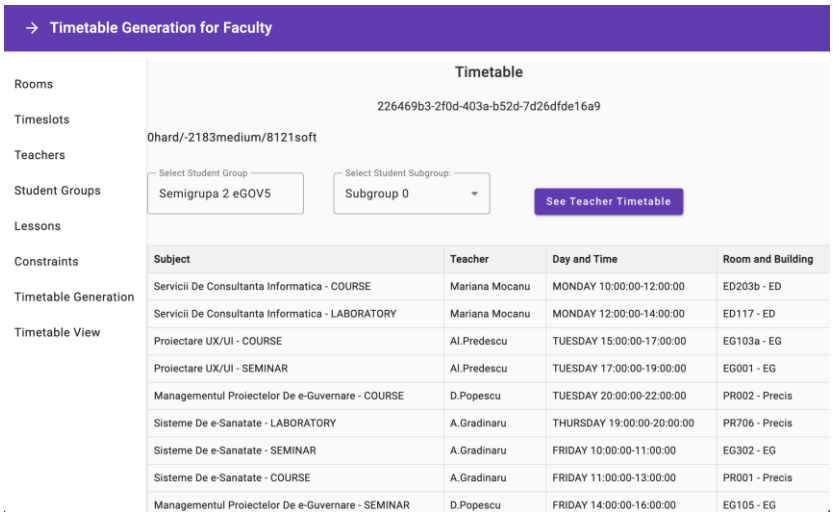


**Figure 10.** Results of the second run of the algorithm.

The timetable generation algorithm was also tested on a MacBook Air M1 (2020) to evaluate performance differences due to hardware variations with 4 threads for processing.

After running the algorithm for 5 hours, the best score obtained was -3122init/0hard/-462medium/3417soft. Extending the runtime to 8 hours resulted in a score of -1938init/0hard/-798medium/5531soft. Further extending the runtime to 11 hours improved the initial unassigned count to -1102init, with improvements of the soft constraint score, but a lower medium score (-1102init/0hard/-1246medium/6571soft) compared to the 8-hour run. In each case, the score calculation speed was lower in comparison to MacBook Pro M1.

Additionally, an alternative configuration was tested on the MacBook Air M1, where timeslots were assigned to courses first, followed by room assignments during the construction heuristic phase. After 8 hours, this configuration produced a score of -10hard/-5602medium/3361soft. This configuration led to 10 hard constraint violations, 5602 medium constraint violations, and a soft constraint score of 3361.

Despite using 4 threads, the algorithm required considerable time to generate the timetable on the MacBook Pro M1 (Table 1). Running the algorithm for 12 hours helped the solver assign lessons to classrooms and timeslots without violating any hard constraints, significantly enhancing the soft constraint score from 7629 to 8121. However, the extended runtime also increased medium constraint violations from 2102 to 2183, indicating that while the solution became more optimal regarding soft constraints, it struggled to minimize medium constraint violations effectively over the extended duration.

The low medium constraint score can be related to the limited access to information from course lecturers, excluding those who teach seminars or labs. The assumption that these lecturers also teach the course, seminar, and lab was adopted. Despite imposing the constraint that a professor could not teach more than 12 hours per day, the system recorded instances where a professor was allocated 13 or 14 hours of lecture, resulting in scores of -1 and -2 accordingly. This also applies to student groups that are limited to 10 hours of instruction per day. Furthermore, the implementation of the constraints considering teachers' preferences lacks precision. Instead of assigning estimated periods during

 doi:10.20944/preprints202409.0732.v1



which each teacher can teach, we assigned specific time slots. This leads to instances where a teacher is allocated to a time slot that he is unavailable for, although being capable of teaching during that time without any difficulty.

The performance on the MacBook Air M1 revealed that shorter runtimes resulted in higher unassigned counts and lower soft constraint scores. The algorithm possibly needs at least 16 hours to achieve a feasible solution with all timeslots and rooms assigned, but even then, there is no guarantee that the medium constraints violation will decrease over time. This indicates that while the algorithm can assign more lessons over time, it reaches a point where additional runtime does not significantly improve medium or soft constraint scores. Using a configuration that assigned timeslots first resulted in hard and medium constraint violations, indicating a less effective approach for meeting all constraints. The higher medium constraint violations suggest that the initial assignment of timeslots without considering room assignments leads to suboptimal solutions.

A Spring Boot REST API simulation on a MacBook Air M1 demonstrated that generating a feasible solution with instantiated objects in Java required only 6 hours. This result suggests that, while the full-stack solution involving Angular and Spring Boot provides comprehensive functionality, the backend algorithm alone can operate more efficiently when separated from the additional overhead introduced by the web application context. The simulation began with an initial score of -8320init/0hard/0medium/0soft. After approximately 15 minutes, the construction heuristic phase improved the score to 0hard/-1787medium/7848soft. The first local search phase, completed after 2 hours, further refined the score to 0hard/-1781medium/8248soft. The second local search phase concluded after 3 hours, achieving a final score of 0hard/-1778medium/8265soft. The entire process confirmed that the backend algorithm can solve the problem efficiently when isolated from the additional overhead introduced by the web application context.

**Table 1.** Comparative execution results on different machines.

| Machine | Hours | Score | Score Calculation Speed |
|---|---|---|---|
| M1 Air (I) | 5 | -3122init/0hard/-462medium/3417soft | 8703/sec |
| M1 Air (II) | 8 | -1938init/0hard/-798medium/5531soft | 6666/sec |
| M1 Air (III) | 11 | -1102init/0hard/-1246medium/6571soft | 5478/sec |
| M1 Air (alt. config) | 8 | -10hard/-5602medium/3361soft | 664/sec |
| M1 Pro (I) | 8 | -262init/0hard/-2113medium/7647soft | 8401/sec |
| M1 Pro (II) | 12 | 0hard/-2183medium/8121soft | 6342/sec |
| Simulation (M1 Air) | 6 | 0hard/-1778medium/8265soft | 73135/sec |

The entire process of creating a highly complex university timetable from scratch in the current state of the application would take about a week. Compared to generating a timetable from scratch in a manual way, it can be said that automation brings a big advantage to educational institutions. And in the case of schools or colleges even more so, as there is much less data, so chances are that generating a school or college timetable will only take a few days.

The extended run times on both the MacBook Pro M1 and MacBook Air M1 highlight the limitations of current hardware in generating optimal timetables efficiently. More powerful hardware is necessary for real-world applications to achieve quicker and more optimal results. The choice of configuration significantly impacts the quality of the generated timetable. Assigning timeslots first led to more violations, suggesting a need for integrated assignment approaches. The consistent use of four threads aligns with Timefold's recommendations as increasing the thread count beyond this could introduce additional overhead, potentially slowing down the process due to the complexities of thread management. However further optimization in thread management might be needed to improve performance. Unit tests written in JUnit5 validated the constraints, ensuring their correct application and enforcement during the timetable generation process.

The results demonstrate that while the algorithm can generate feasible and optimized timetables, performance and time required are significantly influenced by hardware capabilities and algorithm

 doi:10.20944/preprints202409.0732.v1



configurations. For real-world applications, optimizing the environment and leveraging more powerful infrastructure would be essential for handling real-world cases efficiently. Furthermore, while extended runtimes improved the solution quality regarding soft constraints, they also introduced more medium constraint violations, suggesting a need for better-written constraints and for fine-tuning the algorithm's balancing of different constraint types more efficiently.

## 8. Conclusions and Future Work

The present research delves into the subject of automated timetable creation in an academic institution and examines the background, rationale, aims, and anticipated ramifications of the investigation. The objective of the study is to enhance the scheduling of educational tasks through the creation of an automated framework that produces effective and conflict-free schedules while satisfying the preferences and limitations of both faculty and students.

The present study concerns the optimization of didactic activities, which draws inspiration from the widely recognized timetable problem. The optimization of timetabling poses a complex challenge that pertains to the category of scheduling problems. They manifest in diverse fields, including education, healthcare, transportation, sports, and entertainment. The aforementioned issues are recognized as NP-complete, necessitating the utilization of heuristic methodologies to attain efficient resolutions.

The implementation of an automated system for generating timetables offers a multitude of advantages to an academic institution. The implementation of automation technology leads to a reduction in manual labor, a decrease in the occurrence of errors, and an optimization of resource allocation, ultimately resulting in heightened levels of efficiency and productivity. The study's focus on tackling the obstacles posed by a range of distinct requirements, limitations, and preferences makes a valuable contribution to the domains of computer science, operations research, and artificial intelligence.

To accomplish the aims of this study, a proposed model has been put forth to generate an automatic timetable system. The implemented model integrates a web-based platform that has been constructed utilizing Angular to facilitate the frontend, Java in conjunction with Spring Boot to support the backend and Timefold Solver API for the timetable generation algorithm. The study has taken into account the architectural design, methodologies, algorithms, and appropriateness of a database.

In addition, a research methodology has been delineated to provide direction for the study. This methodology encompasses problem analysis, literature review, system design, implementation, data collection and integration, performance evaluation, and iterative refinement. The utilization of a methodology guarantees a methodical and thorough strategy to effectively tackle the research objectives and challenges.

Regarding performance analysis measures, the evaluation metrics have been deliberated upon to appraise the excellence of the produced timetables, computational efficiency, user contentment, comparison with manual timetabling, empirical experimentation, and data analysis methodologies. These metrics offer valuable perspectives on the efficacy, productivity, and user-friendliness of the system, facilitating enhancements and refinements.

The results obtained from the experiments demonstrate the effectiveness and limitations of the algorithm on different hardware configurations. On the MacBook Pro M1 (2021), the algorithm achieved the best score of -284init/0hard/-2102medium/7629soft after eight hours and 0hard/-2183medium/8121soft after twelve hours, indicating improvements in soft constraint scores but increased medium constraint violations over extended runtime. On the MacBook Air M1 (2020), shorter runtimes (5, 8, or 11 hours) resulted in higher unassigned counts and lower soft constraint scores, necessitating at least 16 hours to achieve a feasible solution. An alternative configuration that first assigned timeslots led to significant hard and medium constraint violations, suggesting a less effective approach.

While existing research has made significant strides in the field of automated timetable generation, it has largely focused on smaller datasets and less complex scenarios. The present study

 doi:10.20944/preprints202409.0732.v1



distinguishes itself by addressing the complexities involved in large-scale scheduling, where hundreds of courses, instructors, rooms, and time slots must be coordinated. This work extends the capabilities of current approaches, demonstrating that even with a much larger and more intricate set of constraints, it is possible to generate effective and efficient timetables. This advancement is critical for institutions with substantial student and faculty populations, where the scheduling process is considerably more challenging.

Several enhancements are planned to further improve the timetable generation system ensuring it meets the diverse needs of educational institutions effectively:

- Interactive Elements: Users will be able to interact with the timetable, making manual adjustments or edits based on specific requirements. This will provide greater flexibility and customization for meeting unique needs.
- Reporting and Communication: The application will enable the generation of reports and summaries of the generated timetables. These reports can be used by administrators, faculty members, and students for various purposes such as academic planning, resource allocation, and schedule communication. Additionally, the application will support communication features to notify stakeholders about timetable updates or changes.
- Load Balancing [55]: Implementing load balancing will enhance the system's performance, particularly when handling large datasets or multiple simultaneous users.
- Automated Data Retrieval and Export: An automated way to retrieve the rooms, timeslots, student groups, teachers, and lessons (via Excel, for example) will be added. This will streamline data entry processes and ensure accuracy. Furthermore, a feature to export the generated timetable will be included.
- Issue Resolution and Entity Updates: Current issues related to the implemented constraints and updating entities using Spring JPA will be addressed to enhance overall system performance and accuracy.
- Algorithm Fine-Tuning: Fine-tuning the algorithm and adding a filtered move selection for faster construction of the initial solution after the Construction Heuristics phase will be undertaken to improve efficiency.
- Benchmarking: A benchmark feature will be added, allowing users to test various configurations and evaluate their impact on performance.
- Centralized Timetable Generation: The application will be extended to enable centralized timetable generation for schools, high schools, and other colleges, each with potentially different constraints. This will expand the system's applicability and utility across various educational institutions.

In conclusion, the study on the implementation of automated approaches for creating timetables in a higher education institution demonstrates promise as an effective solution for the complex task of planning educational activities. By using an automated system and following a systematic study technique, it is feasible to improve resource allocation, minimize conflicts, and enhance overall efficiency. The proposed model, along with the performance analysis measures, provides a strong foundation for future research, implementation, and improvement of the automated scheduling system. Through continuous investigation and iterative improvements, it is feasible to achieve the desired outcome of improving the scheduling process and enhancing the educational experience for both teachers and students in academic institutions.

**Author Contributions:** Conceptualization, P.F.D. and C.T.; formal analysis, P.F.D. and C.T.; investigation, P.F.D. and C.T.; methodology, P.F.D. and C.T.; software, P.F.D.; supervision, C.T.; validation, P.F.D. and C.T.; writing—original draft, P.F.D.; writing—review and editing, C.T. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Informed Consent Statement:** Not Applicable

**Data Availability Statement:** Data available in publicly accessible repository. The data presented in this study are openly available on GitHub at https://github.com/PatrickDiallo23/Automatic-Timetable-Generation-System.

**Conflicts of Interest:** The authors declare no conflicts of interest.

 doi:10.20944/preprints202409.0732.v1



## References


1. Knauer, B.A. Solution of a Timetable Problem, *Computers & Operations Research*, **1974**, *1*, *3–4*, 363-375
2. Petrovic, S.; Burke, E. University timetabling. In Handbook of scheduling: Algorithms, Models, and Performance Analysis, *CRC Press, Boca Raton, FL*, **2004**, 1-23.
3. Pinedo M.L. Planning and scheduling in manufacturing and services. In Springer Series in Operations Research, *Springer, New York*, 2005, 3-8.
4. Fang H. L. *Genetic Algorithms in Timetabling Scheduling*. Ph.D. Thesis, University of Edinburgh, Edinburgh, UK, 1994.
5. D'souza, D.; D'sa, O.; Pillai, P.; Shaikh, P. Multi-Constraint Satisfaction and Solution Optimization Using Genetic Algorithm for Solving Timetable Generation Problem *Proceedings of the International Conference on Recent Advances in Computational Techniques (IC-RACT)* **2020**
6. Teoh, C.-K.; Abdullah M. Y. C.; Haron H. Effect of Pre-Processors on Solution Quality of University Course Timetabling Problem, *2015 IEEE Student Conference on Research and Development (SCOReD)*, Kuala Lumpur, Malaysia, **2015**, 472-477.
7. Wungguli, D.; Nurwan, N. Application of Integer Linear Programming Model in Automatic Lectures Scheduling Optimization, *BAREKENG: J. Math. & App.*, **2020**, 14, 3, 413-424.
8. Schrijver, A. *Theory of Linear and Integer Programming*, Wiley, 1998.
9. Feng, X.; Lee, Y.; Moon, I. An Integer Program and a Hybrid Genetic Algorithm for the University Timetabling Problem, *Optimization Methods and Software*, **2016**, *32*, *3*, 625–649.
10. Schaerf, A. A Survey of Automated Timetabling, *Artificial Intelligence Review*, **1999**, *13*, 87–127.
11. Bertsimas, D.; Tsitsiklis, J. Simulated Annealing, *Statistical Science*, **1993**, *8*, *1*, 10-15.
12. Pirim, H.; Eksioglu, B.; Bayraktar, E. Tabu Search: A Comparative Study, 2008. 10.5772/5637.
13. Socha, K.; Sampels, M.; Manfrin, M. Ant Algorithms for the University Course Timetabling Problem with Regard to the State-of-the-Art. In: *Cagnoni, S., et al. Applications of Evolutionary Computing*. EvoWorkshops 2003. Lecture Notes in Computer Science, vol 2611. Springer, Berlin, Heidelberg.
14. Abuhamdah A.; Ayob M. MPCA-ARDA for Solving Course Timetabling Problems, *2011 3rd Conference on Data Mining and Optimization (DMO)*, Putrajaya, Malaysia, **2011**, 171-177.
15. Abuhamdah A.; Ayob M. Multi-neighbourhood Particle Collision Algorithm for Solving Course Timetabling Problems, *Proceeding in 2009 2nd Conference On Data Mining and Optimization*, Selangor, Malaysia, IEEE, **2009**, 21-27.
16. Abuhamdah A.; Ayob M. Adaptive Randomized Descent Algorithm for Solving Course Timetabling Problems, *International Journal of the Physical Sciences*, **2010**, *5*,*16*, 2516-2522.
17. Wankhede, S. Automatic College Timetable Generation, *International Journal of Scientific Research in Computer Science*, Engineering and Information Technology, 2019.
18. Ong, B. C. *Web-based Scheduling Software for a University*, Dissertation, Universiti Tunku Abdul Rahman, Malaysia, 2020.
19. Atef Yekta, H.; Day, R. Optimization-Based Mechanisms for the Course Allocation Problem. *INFORMS Journal on Computing*, **2020**, *32*, *3*, 641-660.
20. Burgess, R. *Automation for CSUN Computer Science Department Course Scheduling* Doctoral dissertation, California State University, Northridge, 2017.
21. OptaPlanner Official Website - Available online: https://www.optaplanner.org/.
22. Timefold Official Website - Available online: https://timefold.ai/.
23. Fiechter, A. *University Timetable Scheduling* Bachelor Project Report, Università della Svizzera Italiana, Faculty of Informatics, 2018.
24. Grant, A. The Basics of AngularJS, *Beginning AngularJS*, Apress, Berkeley, CA, **2014**, 35-45.
25. Meloni, J.; Kyrnin, J. *HTML, CSS, and JavaScript All in One*, Sams Publishing, 2018.
26. Freeman A. *Essential TypeScript 5*, Third Edition, Manning, 2023.
27. Jasmine Documentation - Available online: https://jasmine.github.io/.
28. Protractor Documentation - Available online: https://www.protractortest.org/.
29. Arnold, K.; Gosling, J.; Holmes, D. *The Java Programming Language*, 4th ed.; Addison-Wesley Professional: Glenview, IL, USA, 2005.
30. Sierra, K.; Bates, B.; Gee, T. *Head First Java: A Brain-Friendly Guide*, 3rd ed.; O'Reilly Media: Sebastopol, CA, USA, 2022.
31. Spring Boot Documentation - Available online: https://docs.spring.io/spring-boot/index.html.
32. Fielding, R.T. *Architectural Styles and the Design of Network-based Software Architectures*, PhD Thesis, 2000.
33. Chain, R *Mastering Spring MVC: From Novice to Expert* Independently published, 2023.
34. Spilcă, L., *Spring Security in Action*; Second Edition; Manning: New York, NY, USA, 2024.
35. Tudose, C. *Java Persistence with Spring Data and Hibernate*; Manning: New York, NY, USA, 2023.
36. Bonteanu, A.M.; Tudose, C. Performance Analysis and Improvement for CRUD Operations in Relational Databases from Java Programs Using JPA, Hibernate, Spring Data JPA, *Applied Sciences – Basel*, **2024**, *14*, *7*, 2743.

 doi:10.20944/preprints202409.0732.v1




37. Tudose, C. *JUnit in Action*; Manning: New York, NY, USA, 2020.
38. Spilcă, L., *Spring Start Here: Learn What You Need and Learn It Well;* Manning: New York, NY, USA, 2021.
39. Erder, M.; Pureur, P., Woods, E. *Continuous Architecture in Practice: Software Architecture in the Age of Agility and DevOps*; Addison-Wesley Professional, 2021.
40. Christofides, N. Vehicle Routing Problem, Revue francaise d'automatique, informatique, recherche operationnelle, **1976**, *10*, *2*, 55-70.
41. Zurn, H.H. Generator Maintenance Scheduling via Successive Approximations Dynamic-Programming *IEEE Transactions on Power Apparatus And Systems*, Quintana, VH, **1975**, *94*, *2*, 665-671.
42. Mandel, R. A. Contribution to Solution of Job-Shop Scheduling Problems *Ekonomicko-Matematicky Obzor*, **1975**, *11*, *1*, 22-27.
43. Bruen A; Dixon R . N-Queens Problem, *Discrete Mathematics*, **1975**, *12*, *4*, 393-395.
44. Ohashi, T.; Aghbari, Z.; Makinouchi, A. Hill-Climbing Algorithm for Efficient Color-Based Image Segmentation, *IASTED International Conference on Signal Processing, Pattern Recognition, and Applications*, **2003**, 17-22.
45. Gendreau, M.; Potvin, J.Y. Tabu Search. Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques, **2005**, 165-186.
46. Burke, E.K.; Bykov, Y. The Late Acceptance Hill-Climbing Heuristic, *European Journal of Operational Research*, **2017**, *258*, *1*, 70-78.
47. McMullan, P. An Extended Implementation of the Great Deluge Algorithm for Course Timetabling, *Computational Science–ICCS 2007: 7th International Conference, Beijing, China*, **2007**, *I*, *7*, 538-545.
48. Aarts, E.; Korst, J.; Michiels, W. Simulated Annealing, *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, **2005**, 187-210.
49. Kalshetti, U.; Nahar, D.; Deshpande, K. Gawas, S.; Sudeep, S. Dynamic Timetable Generation Using Constraint Satisfaction Algorithm. *Proceedings of the Second International Conference on Computer and Communication Technologies* **2015**, *1*, 761-771.
50. Ferrari, L.; Pirozzi, E., *Learn PostgreSQL: Use, manage and build secure and scalable databases with PostgreSQL 16*, 2nd Edition; Packt Publishing; 2023.
51. Shukla, S.; George, J.P.; Tiwari, K.; Kureethara, J.V. Data Security *Data Ethics and Challenges*, 2022, 41-59. Singapore: Springer Singapore.
52. Park, J.S.; Sandhu, R.; Ahn, G.J. Role-Based Access Control on the Web *ACM Transactions on Information and System Security (TISSEC)*, **2001**, *4*, *1*, 37-71.
53. Jones, M.; Bradley, J.; Sakimura, N. JSON Web Token (JWT), RFC 7519, **2015**.
54. Forgy, C.L. Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem. *Readings in Artificial Intelligence and Databases*, **1989**, 547-559, Morgan Kaufmann.
55. Ghomi, E.J.; Rahmani, A.M.; Qader, N.N. Load-Balancing Algorithms in Cloud Computing: A Survey. *Journal of Network and Computer Applications*, **2017**, *88*, 50-71.