

**Course Offering**

by

**Mohammad Salhab**

**Abed Al Rida Nehme**

**Wafik Ibrahim**

**Ibrahim Dahboul**

**Ali Abou Khashfe**

**Mohammad Ghoush**

**Project**

Submitted in Partial Fulfillment of the Requirements for the Degree of Bachelor

in Computer Engineering

Department of Electrical and Computer Engineering

Faculty of Engineering

3rd Year

Contents

[List of Figures 4](#_Toc191980492)

[List of Tables 5](#_Toc191980493)

[Project Description 6](#_Toc191980494)

[Project Overview 6](#_Toc191980495)

[Objective: 6](#_Toc191980496)

[Background: 6](#_Toc191980497)

[Literature Review 8](#_Toc191980498)

[Top Level View 8](#_Toc191980499)

[Historical review 8](#_Toc191980500)

[Optimization Techniques Used 9](#_Toc191980501)

[Research Gaps and Opportunities 9](#_Toc191980502)

[Conclusion 10](#_Toc191980503)

[Applications 10](#_Toc191980504)

[Alternative Designs 10](#_Toc191980505)

[Project Planning 10](#_Toc191980506)

[Constraints: 10](#_Toc191980507)

[Project Issues 11](#_Toc191980508)

[Team Members Tasks 11](#_Toc191980509)

[Ethical Issues 11](#_Toc191980510)

[Software Model Process 11](#_Toc191980511)

[Feasibility Study 11](#_Toc191980512)

[Tools/Technology 11](#_Toc191980513)

[Standards 11](#_Toc191980514)

[Milestones 11](#_Toc191980515)

[Requirements 12](#_Toc191980516)

[Use Cases 12](#_Toc191980517)

[Functional Requirements 12](#_Toc191980518)

[Data Requirements 12](#_Toc191980519)

[Non-Functional Requirements 12](#_Toc191980520)

[Design 13](#_Toc191980521)

[Class Diagrams 13](#_Toc191980522)

[Dynamic Model 13](#_Toc191980523)

[Subsystem Decomposition 13](#_Toc191980524)

[Hardware / software mapping 13](#_Toc191980525)

[User Interface 13](#_Toc191980526)

[Test Plans 14](#_Toc191980527)

[Implementation 15](#_Toc191980528)

[Results Evaluation 16](#_Toc191980529)

[Conclusion 17](#_Toc191980530)

[Summary 17](#_Toc191980531)

[Novelty 17](#_Toc191980532)

[Integrity and Values 17](#_Toc191980533)

[Future Work 17](#_Toc191980534)

[References / Bibliography 18](#_Toc191980535)

[Appendix 20](#_Toc191980536)

# List of Figures

# List of Tables

# Project Description

## Project Overview

The project focuses on a local hosted platform that generates the Course Registration Numbers (CRNs) and timetables for academic institutions, such as universities and schools. The process is done by applying Genetic Algorithm (GA) on the course’s data that is provided by the tutors through a restricted yet smooth and user-friendly web interface that is supported by a database holding all the necessary info. Ensuring sufficient allocation of resources such as classrooms, Dr’s name and time slot.

## Objective:

1. Automate the process of CRNs generation and timetables using Genetic Algorithm.
2. Reduce exponentially the manual effort and time spent by tutors in creating timetables, where all it is expected from them is to provide the needed data.
3. Optimize resource allocation and minimize schedule conflicts.
4. Provide a flexible and more customizable system for academic institutions making it more adaptable for both tutors’ and students’ specific needs and exceptional cases.
5. Ensuring the scalability of the system on various institutions with different sizes and complexities.
6. User-friendly web interface where the usability features are concise and clear for all potential users regardless of their background.

## Background:

University time tabling is a process that undergoes a lot of constraints and limitations and needs optimizing resources efficiently. The growing demand for more flexible, user-friendly, and conflict-free scheduling systems has made this a subject of ongoing research. The process itself is not only limited to logistical problems and faculty availability but also related to the needs of the students and the instructors. All these factors make the university course time-tabling a challenging optimization problem.

The field of academic administration, specifically **timetable scheduling**, is a time-consuming and complex process that requires coordination between the academic administrators to allocate classes and labs for multiple courses. This process involves multiple variables and considerations such as classroom availability and capacity, course/lecture type, faculty availability and student preferences. Therefore, manually creating timetables is not just labor-intensive and complex but also prone to conflicts.

The problem is clear, but since we needed a starting point, we had to ask these questions:

* “How can we engineer a time tabling system that balances the needs of the instructor and the student, ensuring an efficient and ***conflict-free schedule***?”
* “What are the trends int the time tabling field and what options does universities rely on when it comes to***modern course offering system***?”
* “How can we optimize the time tabling to improve efficiency and accuracy of university *course* ***timetabling systems***?”

These questions serve as the foundation for investigating various optimization methods, such as meta-heuristics and machine learning techniques, and their impact on the real-world solution for the scheduling problem. So, we had to set boundaries for the studies that will be shaped by these factors:

* Time frame: we wanted to spend about **3-4 hours** revising the research since we didn’t have much time and we were already similar with the project idea and the choices we wanted to make, and since optimization is an old trend so we included the studies from the last **10-15 years.**
* Geographical Scope: We decided not to limit our focus on a certain region, and we made it **global**.
* Types of Studies: we mainly focused on the studies that included time tabling and the optimization techniques to make time tabling as efficient as possible. So, we focused on case studies of **existing timetable systems**. And we mostly included research about real life applications.

To handle such issues, an advanced algorithm like **Genetic Algorithm (GAs)** can be a powerful tool to automate and optimize such process promising to provide the global optimal timetable tailored for students’ and tutors’ preferences.

**Genetic Algorithms** are an evolutionary algorithm inspired by the natural selection and genetics principles in the real world of constant evolution. They work by continuously evolving a collection of potential solutions to a problem, using operations/functions like selection, crossover, and mutation. GAs explore numerous solutions to find optimal ones while considering the multiple given constraints which makes it well-suited for such a problem. Their ability to handle such non-linear, dynamic and multi-constraint problem makes them ideal for generating tailored conflict-free timetables.

In the case of **course offering**, GAs will optimize the allocation of resources (e.g. classrooms, labs, time slots, tutors’ availability) all while considering the constraints like course prerequisites, tutor availability, and student preferences. By implementing such algorithm, academic institutions could generate effortlessly and automatically timetables significantly reducing the manual effort, enhancing the experience and provide a highly adaptable and scalable solution, allowing institutions to customize this algorithm to their specific need and constraints.

## 

## Literature Review

### Top Level View

Universities usually set their timetables based on various constraints and limitations, these limitations can vary between departments. For example, a department may have many instructors and professors and have a very nested study plan that will of course increase the complexity of the process.

The core challenge in timetable lies in balancing the needs and preferences of **students**, **instructors**, and **universities,** with taking into consideration that each one of them might have its own priority. Instructors may have limited availability, while students often have specific course preferences or constraints based on their individual schedules.

We noticed that the timetable problem can be highlighted by the following key constraints:

* Instructor availability.
* Room preferences.
* Room allocation.
* Course pre-requisites.
* Time-slot restrictions.

### Historical review

University timetable has evolved from **manual scheduling** and basic **heuristics** to more advanced **optimization techniques.** Now we mostly use genetic algorithms and the meta-heuristics for handling complex constraints efficiently.

**Current Trends:**

AI and **machine learning** are increasingly used for real-time, dynamic scheduling, while **constraint programming** and **meta-heuristics** help generate conflict-free timetables. Also, we noticed that most designs are focused on the user interface and making it easier and user friendly.

**Relevance to Our Project:**

Our project aims to create an optimized, conflict-free timetable system by using advanced algorithms to meet the needs of both students and instructors while adapting to real-time data.

**Key Themes of Literature**

Optimization Algorithms for University Timetabling

* Overview: Common optimization techniques include **genetic algorithms**, **simulated annealing**, and **meta-heuristics**, which help generate conflict-free timetables and efficiently allocate resources.
* Relevance: These techniques can optimize your course offering system by resolving conflicts, ensuring efficient room allocation, and handling multiple constraints.
* **Studies to Reference**:
  + *"A Review of Optimization Algorithms for University Timetable Scheduling"*
  + *"Optimization Techniques in University Timetabling Problem"*
  + *"Meta-heuristic approaches for the University Course Timetabling Problem"*

Challenges in University Timetabling

* **Common Challenges**: we have constraints concerning the student, the Doctor, the lab instructor, university requirements, study plan.
* **Relevance**: Our system can address these by using optimization algorithms to resolve conflicts and ensure efficient use of resources.
* **Studies to Reference**:
  + *"School timetabling for quality student and teacher schedules"*
  + *"Optimization Techniques in University Timetabling Problem"*

User-Centered Design in Timetabling Systems

* **Balancing Needs**: Effective systems balance the needs of both students and instructors by considering their availability and preferences.
* **Relevance**: Your system can incorporate user input to allow customization, improving user satisfaction (like letting the dr decide what hours he prefers)
* **Studies to Reference**:
  + *"School timetabling for quality student and teacher schedules"*
  + *"A Survey of University Course Timetabling Problem"*

### Optimization Techniques Used

1. **Heuristic Methods**:
   * *Simple, rule-based approaches like* ***greedy algorithms*** *are fast but may not guarantee optimal solutions.*
2. **Meta-heuristics**:
   * *Techniques such as* ***genetic algorithms****,* ***simulated annealing****, and* ***tabu search*** *are more effective for large-scale problems, generating high-quality solutions by navigating complex constraint spaces.*
3. **Constraint Programming**:
   * *Models the timetabling problem with a focus on constraints (e.g., room, time, availability) and utilizes search techniques for precise solutions.(we previously did this in the basic version we created as a project for the programing two course)*
4. **AI and Machine Learning**:
   * *AI methods like* ***reinforcement learning*** *and* ***machine learning*** *help optimize timetables dynamically by learning from historical data and adapting in real-time.*

### Research Gaps and Opportunities

* ***Limited Real-Time Adaptation****: Most systems still struggle with real-time scheduling adjustments. There’s an opportunity to develop* ***AI-driven systems*** *that adapt dynamically to changes in instructor or student availability. (we can include the information we learnt from the Introduction to Ai to make an Ai module that is able to learn how to construct the time table like a human) (****he can’t replace the efforts of both Dr. Imane and Dr. Hiba****)*
* ***Integration of User Preferences****: While user preferences are often considered, integrating* ***real-time feedback*** *from both students and instructors into the scheduling process remains underexplored. (for example, we have courses that is always set by other departments for general engineering requirements)*
* ***Scalability and Efficiency****: Many current systems face scalability issues when applied to larger universities. There’s a gap in creating* ***highly scalable*** *optimization methods that can handle large datasets efficiently.*
* ***Hybrid Approaches****: Combining multiple optimization techniques, such as* ***AI*** *with* ***meta-heuristics****, could offer better solutions, but more research is needed on* ***hybrid models****.*

### Conclusion

University course timetabling remains a complex problem that involves balancing multiple constraints such as instructor availability, room allocation, and student preferences. Optimization techniques like **meta-heuristics**, **genetic programming**, and **AI** are crucial in developing efficient timetables. Despite multiple researches and papers, challenges such as real-time adaptation and scalability persist. However, these challenges also present opportunities for future research, particularly in areas like **AI-driven scheduling** and **hybrid optimization models**.

This review highlights the importance of integrating advanced optimization methods to improve timetabling systems, ultimately making them more flexible, scalable, and adaptive to real-world needs

## Applications

This GAs-based project is highly reusable and can be used in various scenarios. In educational institutions like BAU, it streamlines and automates the production of university timetables and CRNs with maximum resource usage by efficient allocation. It eliminates significant manual effort and can accommodate changes easily, for instance, adding new courses, making room for irregular students, or adjusting tutors' availability.

Other than education, the system can be applied in a variety of other fields. For example, it can schedule employee training sessions, hospital staff shifts, or conference events. In short, it's a all-purpose solution for any complex, non-linear problem with many variables and intricate constraints.

## Alternative Designs

1. Ant Colony Optimization (ACO) – ACO mimics the ant behavior while searching for food. where each ant will explore paths to find the solution. Over time the ants will find the shortest paths reinforcing good solutions while weaker paths will fade over time
2. Cuckoo Search Algorithm (CSA) inspired by cuckoo birds, where each solution will represent an egg in the nest the new egg (solution) will replace the worst ones, ensuring continuous improvement.
3. Swarm Intelligence (SI) each solution will represent a particle. A group of solutions moves around a search space each solution adjusts its position based on its best-found solution
4. Tabu Search mimics human problem solving by remembering past choices and avoiding repeating bad one.

# Project Planning

## Constraints:

* Implementation Environment of the Current System
* Partner or Collaborative Applications
* Off-the-shelf Software
* Anticipated Workplace Environment
* Schedule Constraints
* Budget Constraints

## Project Issues

Issues that have been raised and do not yet have a conclusion.

Migration to the New Product

Risks

## Team Members Tasks

Manager

Designer

Developper

## Ethical Issues

## Software Model Process

## 

## Feasibility Study

## 

## Tools/Technology

## Standards

## 

## Milestones

# Requirements

## Use Cases

This section begins to describe in more specific and precise detail exactly what steps the system takes in the course of its performance. Use cases serve not only to more specifically define the system (and its boundaries), but also to identify functional requirements, to identify initial objects / classes, and to organize the work.

## Functional Requirements

## Data Requirements

## Non-Functional Requirements

Performance Requirements

Dependability Requirements

Maintainability and Supportability Requirements

Security Requirements

Usability and Humanity Requirements

Look and Feel Requirements

Operational and Environmental Requirements

Cultural and Political Requirements

Legal Requirements

# Design

## Class Diagrams

## Dynamic Model

## Subsystem Decomposition

## Hardware / software mapping

## User Interface

# Test Plans

Features to be tested / not to be tested

Pass/Fail Criteria

Approach

Suspension and resumption

Testing materials (hardware / software requirements)

Test cases

Testing schedule

# Implementation

Output

# Results Evaluation

# Conclusion

## Summary

## Novelty

## Integrity and Values

## Future Work

# References / Bibliography

1. *Alghamdi, H., Alsubait, T., Alhakami, H., & Baz, A. (2020). A review of optimization algorithms for university timetable scheduling. Engineering, Technology & Applied Science Research, 10, 6410–6417.* [*A Review of Optimization Algorithms for University Timetable Scheduling | Engineering, Technology & Applied Science Research*](https://www.etasr.com/index.php/ETASR/article/view/3832)
2. *Bashab, A., Ibrahim, A. O., Hashem, I. A. T., Aggarwal, K., Mukhlif, F., Ghaleb, F. A., & Abdelmaboud, A. (2023). Optimization techniques in university timetabling problem: Constraints, methodologies, benchmarks, and open issues. Computers, Materials & Continua, 74, 6461–6484.* [*CMC | Optimization Techniques in University Timetabling Problem: Constraints, Methodologies, Benchmarks, and Open Issues*](https://www.techscience.com/cmc/v74n3/50939)
3. *Birbas, T., Daskalaki, S., & Housos, E. (2009). School timetabling for quality student and teacher schedules. Journal of Scheduling, 12, 177–197.* [*School timetabling for quality student and teacher schedules | Journal of Scheduling*](https://link.springer.com/article/10.1007/s10951-008-0088-2)
4. *Chen, P., Sze, S. N., Goh, S. L., & Kendall, G. (2021). A survey of university course timetabling problem: Perspectives, trends and opportunities. IEEE Access, 9, 82031–82053.* [*A Survey of University Course Timetabling Problem: Perspectives, Trends and Opportunities | IEEE Journals & Magazine | IEEE Xplore*](https://ieeexplore.ieee.org/document/9499056)
5. *Abdipoor, S., Yaakob, R., Goh, S. L., & Abdullah, S. (2023). Meta-heuristic approaches for the university course timetabling problem. Intelligent Systems with Applications, 19, 200253.* [*Meta-heuristic approaches for the University Course Timetabling Problem - ScienceDirect*](https://www.sciencedirect.com/science/article/pii/S2667305323000789?via%3Dihub)

: Cite all ideas, concepts, text, data that are not your own. If you make a statement, back it up with your own data or a reference. All references cited in the text must be listed. There are two main ways to cite a reference within a text:

Citing the reference by author’s name: the author’s name must be placed at the end of the sentence that is taken from that reference along with the year of publication, then in the reference section the author’s name is to be arranged in alphabetical order.

Citing the reference by numbers: you should start numbering from 1 and continue according to order of appearance in text. Numbers should be placed the end of the sentence that is taken from that reference, then in the reference section you start your reference list from number 1.

You are recommended to use the APA writing style, which cites the reference by the author’s name, in your references’ citations.

The first line of each entry in your reference list should be on the left margin. Subsequent lines should be indented five spaces from the margin. All references should be double-spaced. Capitalize only the first word of a title or subtitle of a work. Italicize titles of books and journals. Note that the italicizing in these entries often continues

beneath commas and periods. Each entry is separated from the next by a double space (thus the entire reference list is double spaced, with no extra returns added).

Authors' names are inverted (last name first); give last name and initials for all authors of a particular work. Your reference list should be alphabetized by authors' last names. If you have more than one work by a particular author, order them by publication date, oldest to newest (thus a 1991 article would appear before a 1996 article). When an author appears as a sole author and as the first author of a group, list the one-author entries first. If no author is given for a particular source, alphabetize by the title of the piece and use a shortened version of the title for parenthetical citations. Use "&" instead of “and” on the reference page and only within parentheses when citing multiple authors of a single work in your text. At the end of the project list all references cited in the text in alphabetical order.

For an article in a journal:

***Author, A. A., Author, B. B., & Author, C. C. (Year of Publication). Title of article. Title of periodical, Volume Number, pages.***

Example 1: Harlow, H. F. (1983). Fundamentals for preparing psychology journal articles. Journal of Comparative and Physiological Psychology, 55, 893-896.

Example 2: Kernis, M. H., Cornell, D. P., Sun, C. R., Berry, A., & Harlow, T. (1993). There's more to self-esteem than whether it is high or low: The importance of stability of self-esteem. Journal of Personality and Social Psychology, 65, 1190-1204.

For a chapter in a book:

***Author, A. A., & Author, B. B. (Year of Publication). Title of chapter. In A. Editor &***

B. Editor (Eds.), Title of book (pages of chapter). Location: Publisher. When you list the pages of the chapter or essay in parentheses after the book title, use "pp." before the numbers: (pp. 1-21).

Example: O'Neil, J. M., & Egan, J. (1992). Men's and women's gender role journeys: Metaphor for healing, transition, and transformation. In B. R. Wainrib (Ed.), Gender issues across the life cycle (pp. 107-123). New York: Springer.

For a web page:

***Author, A. A., & Author, B. B. (Date of Publication or Revision). Title of full work [online]. Retrieved month, day, year, from source Web site: URL.***

Example: Chou, L., McClintock, R., Moretti, F. & Nix, D. H. (1993.) Technology and education: New wine in new bottles: Choosing pasts and imagining educational futures. Retrieved August 24, 2000, from Columbia University Institute for Learning Technologies Web site: <http://www.ilt.columbia.edu/publications/papers/newwine1.html>

For an online journal:

***Author, A. A., & Author, B. B. (Date of Publication). Title of article. Title of periodical, xx, xxx-xxx. Retrieved month, day, year, from URL.***

Example: Frederickson, B. L. (2000, March 7). Cultivating positive emotions to optimize health and well-being. Prevention &Treatment, 3 Article 001a. Retrieved November 20, 2000, from <http://journals.apa.org/prevention/volume3/pre0030001a.html>

# Appendix

Glossary

Naming Conventions and Definitions

Code and links

User Manual