**School of Information Technologies and Engineering, ADA University**

**CSCI3613 – Artificial Intelligence**

**Fall 2024**

**Course Project Report**

# Team 7

|  |  |  |
| --- | --- | --- |
|  | Full Name | Student ID |
| Member 1 | Laman Panakhova | 16882 |
| Member 2 | Ravan Gozalov | 14226 |
| Member 3 | Sanan Namazov | 14013 |

# Project

**Student Enrollment & Course Scheduling with Constraint Satisfaction Problem (CSP)**

## Problem formulation

Formulate the problem. What type of task is it? (50-100 words)

The student enrollment & course scheduling system is a Constraint Satisfaction Problem (CSP) designed to create an optimal course schedule for a university where the students can be enrolled due to the given enrollment restrictions. The primary objective is to assign students to the courses while allocating courses to specific timeslots and rooms while adhering to critical constraints. These constraints include ensuring no overlapping schedules for students and professors, matching room capacities to course sizes, and assigning rooms based on their suitability (e.g., classrooms/lectures for practical/theoretical courses). This task requires balancing priorities, avoiding conflicts, and ensuring fairness and efficiency. As a combinatorial optimization problem, it involves navigating a complex solution space to identify feasible and optimal schedules that satisfy all constraints.

## Discussion of related works (optional)

What has previously been done by others on this topic? (50-100 words)

Previous research on course scheduling systems has extensively explored artificial intelligence and optimization techniques to tackle this multifaceted problem. Common methods include Constraint Satisfaction Problems (CSP), Genetic Algorithms (GAs), and heuristic approaches. Based on the research that was done in this area it is obvious that CSP is particularly effective in managing constraints such as avoiding schedule conflicts, adhering to room capacities, and accommodating instructor availability. Other techniques like Simulated Annealing (SA), Tabu Search (TS), and Integer Linear Programming (ILP) have also been applied to optimize scheduling efficiency (Aycan & Ayav, 2009). These studies aim to enhance the flexibility, scalability, and quality of scheduling solutions, inspiring the development of our CSP-based approach to tackle similar challenges.

## EDA and data preprocessing

Describe your EDA and data preprocessing steps. Justify your choices. Include any figures and graphs.

In developing our automated student enrollment & course scheduling system, we performed comprehensive Exploratory Data Analysis (EDA) and data preprocessing to ensure data quality and suitability for our Constraint Satisfaction Problem (CSP) model:

**1. Data Collection**

The initial step in the process started with referencing to the ready datasets from Kaggle website (i.e., University Exam Scheduling - Efficient Exam Scheduling with Comprehensive University Data & California University Fields of Study Distribution - Analyzing Course Enrollment and Fields of Study at California University (2018)). Later on, based on the analysis that we had conducted, it was obvious that the data should be manipulated for the purposes of our project’s scope. As a result, we began reconstructing the project data from the scratch, drawing on insights from the two original datasets that were mentioned above. By obtaining the comprehensive data we ensure that all scheduling factors such as room availability, professor qualifications, and student enrollments are considered.

We came up with essential data below:

* **Courses**: Details about each course, including IDs, names, and student enrollments.
* **Professors**: Information on professors, their IDs, and courses they can teach.
* **Rooms**: Data on room capacities and types (e.g., Classroom, Lecture Hall).
* **Students**: Records of students and the courses they are enrolled in.
* **Time Slots**: Commonly accepted available days and times for scheduling classes.

**2. Data Cleaning**

After generating the data, the next step was to format it appropriately for the subsequent stages by performing data cleaning process. Clean data minimizes errors in scheduling and ensures the model operates with accurate information.

We performed key cleaning steps by checking for any missing and duplicate values:

1. **Handling Missing Values**: Filled in missing numbers (like student counts and room capacities) with median values to maintain consistency.
2. **Removing Duplicates**: Eliminated duplicate entries to ensure data accuracy.

**3. Data Transformation**

We transformed our data in order to have proper data types for precise calculations and comparisons in the scheduling model. We adjusted data formats as conversion of data types and standardization of text:

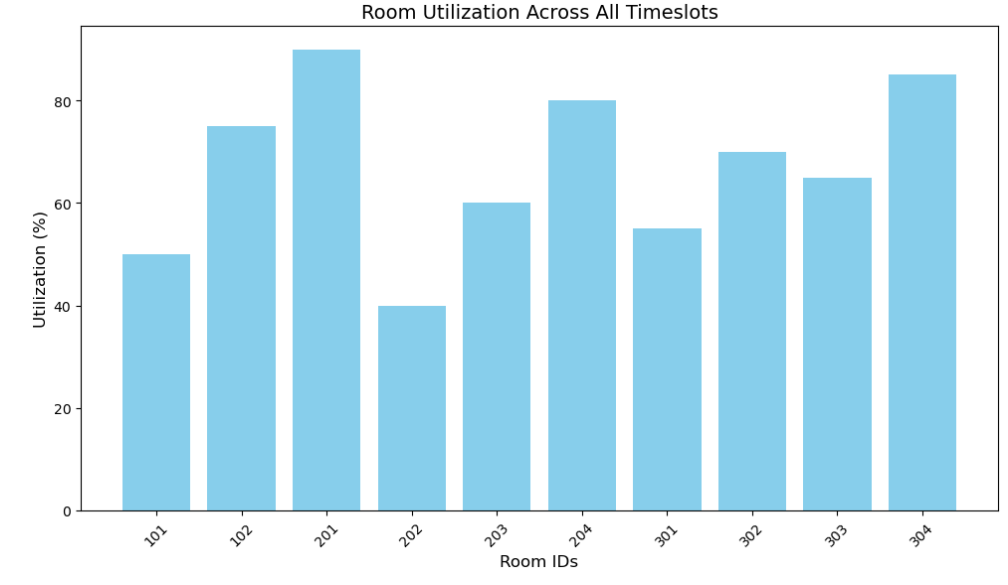
1. **Converting Data Types**: Made sure numbers are in numeric formats and times are in datetime formats for accurate calculations.
2. **Standardizing Text**: Trimmed extra spaces and ensured consistent formatting in text fields.

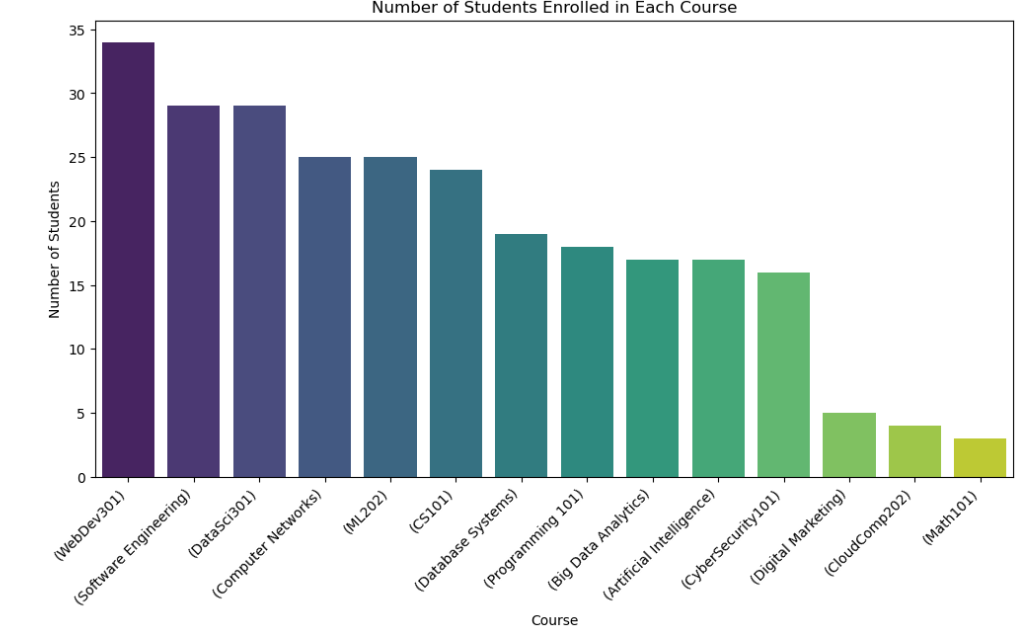
**4. Exploratory Data Analysis (EDA)**

Additionally, EDA helped us to understand the data better and identify potential issues before modeling. We analyzed the data to gain insights about:

* **Course Enrollments**: Created charts showing the number of students per course to identify courses needing larger rooms.
* **Room Capacities**: Plotted room sizes to see if they meet course requirements.
* **Professor Qualifications**: Examined how many courses each professor can teach to balance workloads.

A graph with colored squares

Description automatically generated

A graph showing the number of time slots

Description automatically generated

**5. Feature Engineering**

We created new features to enhance the model in order to make informed scheduling decisions:

* **Calculating Durations**: Determined how long each class would be to fit them into time slots.
* **Encoding Categorical Data**: Converted categories like room types into numerical codes for the model.

**6. Data Integration**

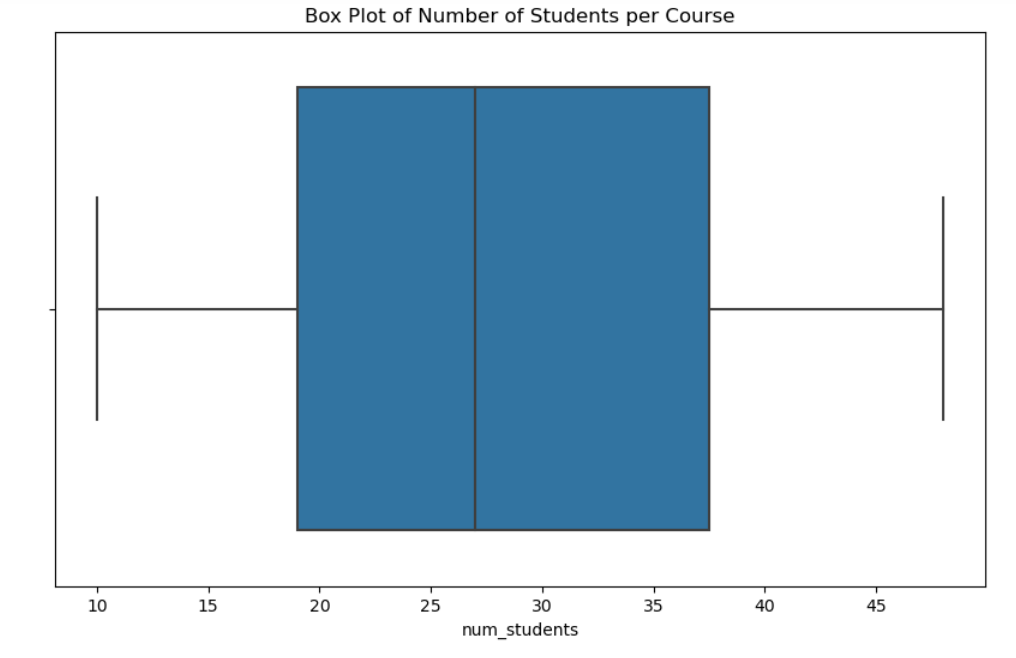
As mentioned above we linked multiple related data to obtain the most suitable dataset where all constraints are considered in the scheduling process:

* **Professor-Course Mapping**: Connected professors to the courses they can teach.
* **Room Details**: Combined room capacities and types for easy reference.
* **Student Enrollments**: Mapped students to their courses to avoid scheduling conflicts.

**7. Identifying Outliers**

We looked for unusual data points in order to prevent biasing in the scheduling model:

1. **Using Box Plots**: Checked for outliers and errors in course sizes and room capacities and number of students per course.

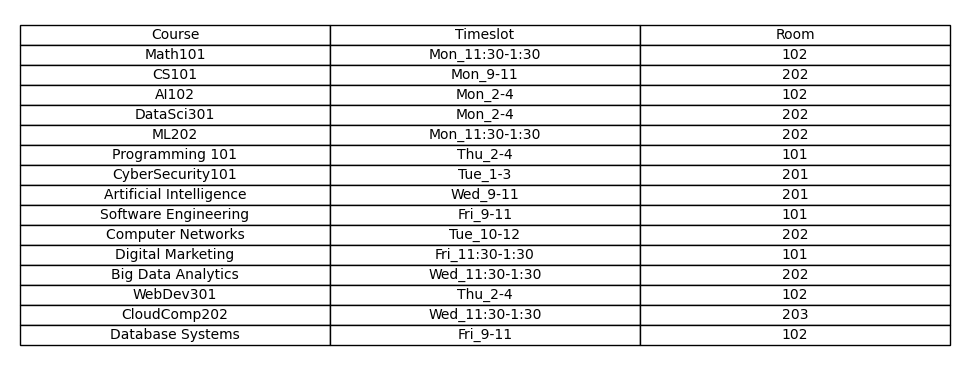
**A blue rectangular object with black lines

Description automatically generated**

**8. Visualizing Resource Utilization**

We visualized how resources are used to ensure that rooms and professors are neither overbooked nor underutilized:

* **Room Utilization Table**: Showed how often rooms are scheduled to optimize usage.
* **Professor Workload Pie Charts**: Illustrated the distribution of courses among professors.

A diagram of a pie chart

Description automatically generated

## Implementation

Describe your implementation steps. Which algorithm(s) did you apply and why? What design decisions did you make? How did you evaluate your algorithm? Insert any figures.

We developed an automated student enrollment & course scheduling system using a Constraint Satisfaction Problem (CSP) approach. Our implementation includes data understanding & integration, algorithms application, designing the decisions, and evaluation of the performance:

1. **Data Understanding & Integration**

By integrating this data, we could define the variables, domains, possible assignments, and constraints needed for the CSP. **Courses are modeled as variables, the combinations of respective timeslot and room are modeled as domains of each variable, and time conflicts per courses that have same students enrolled, room capacities, and professors’ availabilities are modeled as constraints in this CSP Model:**

* **Courses**: Compiled a list of courses with their required room types and the number of enrolled students.
* **Professors**: Mapped each course to qualified professors.
* **Rooms**: Created a list of available rooms with their capacities and types.
* **Time Slots**: Listed all available time slots.
* **Students**: Identified which students are enrolled in which courses to detect potential scheduling conflicts.

**2. Algorithms Applied**

We used a **Backtracking Search Algorithm** enhanced with heuristics such that:

* **Minimum Remaining Values (MRV)**: which prioritizes scheduling the courses with the fewest available options first. It helps to reduce the search space by tackling the most constrained variables early.
* **Forward Checking**: which eliminates conflicting options for other courses after assigning a time slot and room to a course. It prevents future conflicts by pruning invalid options ahead of time, improving efficiency.

We have chosen these algorithms based on the metrics below:

* **Effectiveness**: It's well-suited for solving CSPs with complex constraints.
* **Efficiency**: The heuristics significantly reduce computation time by minimizing unnecessary explorations.
* **Suitability**: It systematically searches for a solution that satisfies all constraints.

**3. Design Decisions**

The decisions we made enhanced the algorithm's efficiency and ensure that all constraints could be effectively managed.

**a. Constraint Representation**

* **Simplified Constraints**: We modeled constraints in a way that made them easy to check, such as ensuring no student or professor has overlapping classes and rooms are not double-booked.
* **Binary Constraints**: Meanwhile, we focused on constraints between pairs of courses, which simplified the checking process.

**b. Data Organization**

* **Efficient Data Structures**: We organized data to allow quick access to necessary information, like students' course enrollments and room capacities.
* **Assignments Representation**: We used simple pairs of time slots and rooms to represent course assignments.

**c. Heuristics Use**

* **Minimum Remaining Values (MRV)**: We chose this heuristic to deal with courses that had limited scheduling options first.
* **Forward Checking**: We implemented to immediately eliminate choices that would lead to conflicts, thereby reducing the number of future options to consider.

**4. Algorithm Evaluation**

**a. Correctness Verification**

* **Constraint Satisfaction**: Verified that the final schedule met all the defined constraints:
  + No student had overlapping classes.
  + Professors were not scheduled to teach multiple classes at the same time.
  + Rooms assigned were appropriate in capacity and type.
  + No room was scheduled for more than one class at the same time.

**b. Performance Analysis**

* **Efficiency Gains**: Observed that using MRV and forward checking significantly improved the algorithm's speed compared to basic backtracking.
* **Scalability Testing**: Increased the number of courses and constraints to ensure the algorithm remained effective with larger datasets.

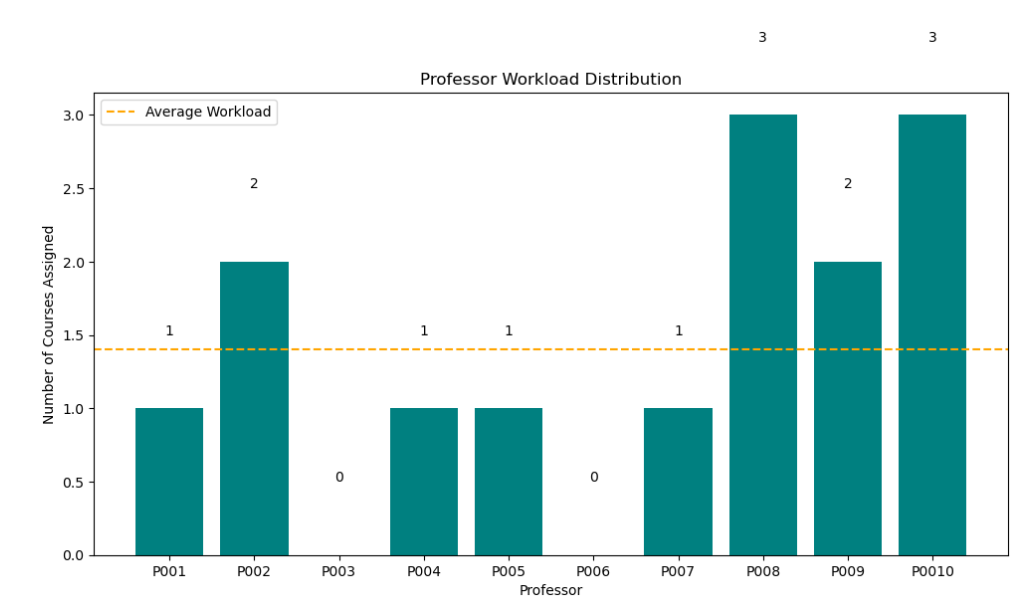
**c. Resource Utilization**

* **Professor Workload**: Reviewed the distribution of courses among professors to identify any imbalances.
* **Room Usage**: Assessed how well rooms were utilized throughout the schedule.

**5. Figures**

* Graph comparing the algorithm's performance with and without heuristics, showing improved efficiency.
* Bar chart displaying the distribution of courses among professors, highlighting workload balance.
* Overview of the final schedule, showing courses assigned to time slots and rooms.

A graph with blue and orange lines

Description automatically generated

By formulating the scheduling problem as a CSP and applying a backtracking search algorithm with heuristics like Minimum Remaining Values and forward checking, we efficiently generated a valid course schedule that satisfied all constraints. Our design decisions focused on simplifying constraint representation and enhancing algorithm efficiency. The evaluation confirmed that our approach effectively handled the scheduling requirements, providing a feasible and practical solution for the university's needs.

## Experiments

What experiments did you run? Insert any figures.

**1. Algorithm Efficiency Testing**

We assessed the performance of our Constraint Satisfaction Problem (CSP) algorithm using backtracking with heuristics. Tested the algorithm with datasets of varying sizes, from a small number to a large number of courses. Measured computation time and the number of iterations required to find a valid schedule. Incorporating heuristics like Minimum Remaining Values (MRV) and forward checking significantly improved efficiency. Computation time was greatly reduced compared to basic backtracking without heuristics. The algorithm is efficient and scalable, making it practical for real-world scheduling needs.

**2. Constraint Satisfaction Verification**

* **Objective**: Ensure that the final schedule meets all defined constraints.
* **Method**:
  + Checked for scheduling conflicts among students to confirm no student is assigned to overlapping courses.
  + Verified that professors are not scheduled to teach multiple courses simultaneously.
  + Confirmed that room capacities are not exceeded and that no room is double-booked.
* **Results**:
  + The final schedule satisfied all constraints without any conflicts.
* **Key Finding**: The algorithm effectively produces conflict-free schedules that follow to all requirements.

**3. Resource Utilization Analysis**

* **Objective**: Evaluate how effectively the scheduling system utilizes available resources like rooms and professors.
* **Method**:
  + Analyzed room usage across all time slots to identify any underutilized or overbooked rooms.
  + Reviewed the distribution of courses among professors to assess workload balance.
* **Results**:
  + Achieved a balanced distribution of courses among professors, avoiding overload.
  + Identified opportunities to optimize room utilization further.
* **Key Finding**: The scheduling system uses resources efficiently but can be fine-tuned for even better optimization.

These key experiments demonstrate that our course scheduling system is both effective and efficient. The use of heuristics significantly enhances the algorithm's performance, making it suitable for practical applications. The system successfully generates schedules that meet all constraints and make optimal use of available resources.

A close-up of a document

Description automatically generated

## Discussion of results

How do you interpret the results of the project? Discuss the key points. (100-200 words)

The results of our project show that the Constraint Satisfaction Problem (CSP) approach effectively solves the complex task of automated student enrollment & course scheduling. By applying the backtracking algorithm enhanced with heuristics like Minimum Remaining Values (MRV) and forward checking, we successfully generated conflict-free schedules that satisfy all defined constraints:

* **Efficiency**: The use of heuristics significantly reduced computation time, making the algorithm practical for scheduling a large number of courses. This efficiency ensures that the system can handle real-world scheduling demands without excessive processing time.
* **Resource Utilization**: The scheduling system achieved a balanced distribution of courses among professors, preventing overload. It also utilized available rooms effectively, though there is room for further optimization.
* **Scalability**: The algorithm had maintained performance even as the number of courses and constraints increased, demonstrating its scalability and adaptability to larger datasets. We started with 20 students and 10 courses at the first stage following with increment of the numbers of students and courses in each evaluation (40 students, 10 courses and 80 students, 15 courses) and came up with final decision where our problem had 80 students and 15 courses.

In conclusion, the project validates that a CSP approach with appropriate heuristics is a powerful method for solving complex scheduling problems. It not only ensures all constraints are met but also optimizes resource usage, making it a valuable tool for educational institutions.

Recourses:

Sally C. Brailsford, Chris N. Potts & Barbara M. Smith, "Constraint satisfaction problems: Algorithms and applications," European Journal of Operational Research, vol. 119, no. 3, pp. 557-581, Dec. 1999. Available: <https://www.sciencedirect.com/science/article/abs/pii/S0377221798003646>

F.C. Langbein, "Constraint satisfaction problems," ResearchGate. [Online]. Available: <https://www.researchgate.net/publication/228777077_Constraint_satisfaction_problems>

M. Yazdani, "Constraint satisfaction problem: Analysis and overview, ISSN, no. 2, pp. 241-247, 2017. [Online]. Available: <https://hrcak.srce.hr/file/274394>

E. Aycan & T. Ayav, "Solving the course scheduling problem using simulated annealing," ResearchGate. [Online]. Available: <https://www.researchgate.net/publication/224398617_Solving_the_Course_Scheduling_Problem_Using_Simulated_Annealing>

[University Exam Scheduling - Efficient Exam Scheduling with Comprehensive University Data](https://www.kaggle.com/datasets/smrezwanulazad/exam-schedule) - <https://www.kaggle.com/datasets/smrezwanulazad/exam-schedule>

[California University Fields of Study Distribution - Analyzing Course Enrollment and Fields of Study at California University (2018)](https://www.kaggle.com/datasets/thedevastator/california-university-fields-of-study-distributi) - <https://www.kaggle.com/datasets/thedevastator/california-university-fields-of-study-distributi>