Project Specification: Genetic Algorithm for University Course Timetabling (UniTime-Based)

Dr. Minhua Huang
COSC4330.001; Spring 2026
School of Engineering & Computer Sciences
A & M University at Corpus Christi

Points: 20
Out: Feb 13, Due: Apr 27.

1 Objectives

The primary goal of this project is to design and implement a **Genetic Algorithm (GA)** to solve the University Classroom Scheduling Problem using a simplified UniTime-style formulation. Students must utilize the provided Purdue Benchmark datasets (`classes_demand.csv` and `rooms_pool.csv`) to achieve the following objectives:

- **Feasibility (Hard Constraints):** Eliminate all room overlaps (two classes assigned to the same room at overlapping time slots) and ensure that class enrollments do not exceed assigned room capacities.
- **Spatial Optimization (Soft Constraints):** Minimize the total weighted Euclidean distance traveled by students between consecutive classes scheduled on the same day using building coordinate information (X, Y).
- **Resource Efficiency:** Improve room utilization by assigning classes to rooms with capacities that closely match class enrollments.

2 Chromosome Representation

A candidate schedule is encoded as a chromosome

$$\phi = [\phi(1), \phi(2), \ldots, \phi(N)],$$

where:

- N is the total number of classes listed in `classes_demand.csv`.
- $\phi(i)$ denotes the **room ID** assigned to class i.
- Each class is assigned to exactly one room, and that room is used consistently for all scheduled meetings during the week.

3 Mathematical Model

Each candidate schedule ϕ is evaluated using a fitness function that strictly prioritizes feasibility while allowing optimization of soft objectives:

$$\mathrm{f}(\phi) = \frac{1}{1 + \mathrm{w_h} \cdot \mathrm{H}(\phi) + \mathrm{w_s} \cdot \mathrm{S}(\phi)} \tag{1}$$

where $w_h \gg w_s$ ensures that feasible solutions are always preferred over infeasible ones.

Hard Constraint Penalty

The hard constraint penalty function $H(\phi)$ is defined as:

$$\mathrm{H}(\phi) = \sum_{\mathrm{k \in Rooms}} \sum_{\mathrm{i,j \in C_k}} \mathbf{1}\left[\mathrm{Overlap(i,j)}\right] + \sum_{\mathrm{i \in Classes}} \mathbf{1}\left[\mathrm{Enrollment_i} > \mathrm{Capacity}_{\phi(\mathrm{i})}\right] \tag{2}$$

where:

- C_k is the set of classes assigned to room k.
- $\mathbf{1}[\cdot]$ is an indicator function equal to 1 if the condition is true and 0 otherwise.

Any nonzero value of $H(\phi)$ indicates an infeasible schedule.

Soft Constraint Penalty

The soft constraint penalty $S(\phi)$ approximates total student travel distance across one week:

$$\mathrm{S}(\phi) = \sum_{\mathrm{d} \in \{\text{Mon, Tue, Wed, Thu, Fri, Sat, Sun}\}} \sum_{\mathrm{t}=1}^{\mathrm{T_d}-1} \mathrm{Enrollment_t} \cdot \sqrt{\left(\mathrm{X}_{\phi(\mathrm{t}+1)} - \mathrm{X}_{\phi(\mathrm{t})}\right)^2 + \left(\mathrm{Y}_{\phi(\mathrm{t}+1)} - \mathrm{Y}_{\phi(\mathrm{t})}\right)^2} \tag{3}$$

where:

- d indexes the days of the week.
- T_d is the number of classes scheduled on day d.
- Classes on each day are sorted by `StartingSlot`.
- $(X_{\phi(t)}, Y_{\phi(t)})$ are room coordinates from `rooms_pool.csv`.
- Enrollment$_t$ weights the distance by the number of affected students.

Weight Selection Strategy

To strictly enforce feasibility, the hard constraint weight is chosen as

$$w_h \in [10^3, 10^5], \quad w_s = 1.$$

4 Genetic Algorithm

Algorithm 1 Genetic Algorithm for University Course Timetabling

Require: Population size P_s, tournament size k, crossover rate p_c, mutation rate p_m, maximum generations G
Ensure: Best schedule ϕ^{best}

Initialize population P with random room assignments
Evaluate fitness $f(\phi)$ for all $\phi \in P$
for $g = 1$ to G **do**
    $P_{\text{new}} \leftarrow \emptyset$
    while $|P_{\text{new}}| < P_s$ **do**
        Select parents using Tournament Selection
        if Random$(0, 1) < p_c$ **then**
            Apply One-Point Crossover
        end if
        Apply Random Resetting Mutation
        Add offspring to P_{new}
    end while
    $P \leftarrow P_{\text{new}}$
    Evaluate fitness for all chromosomes in P
end for
$\phi^{best} \leftarrow$ chromosome with highest fitness
return ϕ^{best}

5 Project Components

5.1 Design, Implementation, and Evaluation

- **Design:** Define a chromosome that assigns each class to a single room for the entire week. Ensure the fitness function strictly penalizes hard constraint violations while encouraging improvements in soft constraints.
- **Implementation:** Implement the GA using **C or C++**. Students must explicitly code tournament selection, one-point crossover, and random resetting mutation using arrays, structs, and control flow constructs.
- **Evaluation:** Compare GA performance against a **Random Assignment Baseline**. Evaluate feasibility using hard constraint violations and solution quality using fitness improvement across generations.

5.2 Written Report

A comprehensive technical report must be submitted. The report should clearly document both the modeling decisions and the algorithmic implementation, and must include:

- An abstract and introduction describing the University Course Timetabling Problem, its NP-hard nature, and the motivation for using a Genetic Algorithm.
- A detailed explanation of the chromosome representation, fitness function formulation, and constraint modeling, including clear justification of the chosen penalty weights (w_h and w_s).
- Plots illustrating the **fitness progression over generations** (learning curve) for at least 100 generations, clearly indicating convergence behavior.
- A summary table comparing the **Random Assignment Baseline** and the **final GA-optimized schedule**, including the number of hard constraint violations and the final fitness value.

5.3 Oral Presentation

Students will present their work in a 10-minute oral presentation. The presentation must include:

- An overview of how the Genetic Algorithm scales to the full dataset of 2,418 classes and 221 rooms, including chromosome size and population design.
- A demonstration of the implemented solution, either through code walkthroughs or visualizations of the final weekly schedule (e.g., room usage or daily assignments).
- A discussion of Genetic Algorithm behavior, including examples of premature convergence, local optima, and the role of mutation in maintaining population diversity.

6 Bonus Credits

Up to **10% bonus credit** may be awarded for extensions beyond the base requirements, including:

- **Advanced Genetic Operators:** Implementation of elitism or adaptive mutation rates, with a brief explanation of their impact on convergence.
- **Heuristic Initialization:** Designing an informed initial population (e.g., assigning larger classes to larger rooms first) to reduce early hard constraint violations.
- **Visualization Tools:** Developing visual aids such as room-occupancy charts or campus building heatmaps based on room (X, Y) coordinates to illustrate scheduling quality.

7 Final Project Grading Rubric

Criterion	**Weight**
Implementation (Code Correctness & Efficiency in C/C++)	40%
Constraint Satisfaction (Hard Constraints Eliminated)	10%
Written Documentation (Technical Clarity & Quality)	25%
Oral Presentation & Q&A	20%
Analysis of Results (Baseline vs. GA Comparison)	5%

Hard Constraint Evaluation: Hard constraint satisfaction is evaluated quantitatively using the hard constraint penalty

$$H(\phi) = \sum_{k \in \text{Rooms}} \sum_{i,j \in C_k} \mathbf{1}\big[\text{Overlap}(i,j)\big] + \sum_{i \in \text{Classes}} \mathbf{1}\big[\text{Enrollment}_i > \text{Capacity}_{\phi(i)}\big].$$

- Full credit (10%) is awarded if the final schedule produced by the GA satisfies $H(\phi) = 0$.
- Partial credit is awarded if $H(\phi) > 0$ but demonstrates a clear reduction compared to the random baseline.
- No credit is awarded if hard constraint violations remain unchanged or increase relative to the baseline.

8 Technical Notes

- **Cartesian Coordinate System:** Each room is associated with fixed building coordinates (X, Y) provided in `rooms_pool.csv`. Distances between rooms are computed using the Euclidean metric:

$$d = \sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2}.$$

  These distances are used exclusively in the evaluation of the soft constraint penalty.
- **Time Slot Overlap Logic:** Two classes i and j are considered to overlap if they share at least one common meeting day and their assigned time intervals intersect. Formally, an overlap occurs if:

$$\max(\text{Start}_i, \text{Start}_j) < \min(\text{End}_i, \text{End}_j),$$

  where $\text{End}_i = \text{Start}_i + \text{Length}_i$, and the condition is evaluated only for days where both classes are scheduled, as indicated by the `Days` field.
- **Random Seed Control:** To ensure reproducibility of experimental results, a fixed random seed must be used when initializing the population and applying genetic operators (e.g., `srand(42)` in C/C++). This is required for baseline comparison and grading consistency.
- **Data Integrity:** The original input files (`classes_demand.csv` and `rooms_pool.csv`) must not be modified. All parsing, validation, and type conversion (e.g., integer fields, day encoding) must be handled programmatically within the C/C++ implementation.