Timetable Scheduling using Genetic Algorithm

Mohammad Faizan
BS CS
FAST NUCES Islamabad
Islamabad, Pakistan
i210666@nu.edu.pk

***Abstract*—This paper presents a solution to the Timetable Scheduling Problem in a university setting using a Genetic Algorithm (GA). The system generates weekly course schedules for multiple sections, professors, and classrooms while satisfying a predefined set of hard and soft constraints. The approach utilizes binary-encoded chromosomes, a fitness function to penalize conflicts, and standard genetic operators—selection, crossover, and mutation. The results demonstrate the ability of the GA to evolve near-optimal schedules with minimal violations.**

***Index Terms*—Genetic Algorithm, Timetable Scheduling, Optimization, Constraint Satisfaction, Evolutionary Computation**

I. Introduction

Timetable scheduling is a classical and computationally challenging problem in academic environments where numerous constraints must be simultaneously satisfied. This project uses Genetic Algorithms—a class of bio-inspired optimization techniques—to generate high-quality schedules via evolutionary processes.

II. Problem Formulation

III. Genetic Algorithm Design

A. Overview & Terminology

Genetic Algorithms mimic natural selection by maintaining a population of candidate solutions (chromosomes), each composed of genes representing schedule attributes. Through iterative selection, recombination, and mutation, GAs evolve the population toward higher fitness (better schedules) :contentReference[oaicite:1]index=1.

B. Chromosome Encoding

We represent a schedule as a fixed-length binary string, where each gene encodes attributes such as course ID, section, professor, day, time slot, and room assignment.

C. Population Initialization

An initial population of N chromosomes is generated randomly, ensuring genetic diversity for sufficient search exploration :contentReference[oaicite:2]index=2.

D. Fitness Function

Fitness is defined as:

$$\text{fitness}(c) = \frac{1}{1 + V(c)}$$

where $V(c)$ is the weighted count of hard and soft constraint violations. Higher-quality schedules have fewer violations, resulting in higher fitness values.

E. Selection

We employ tournament selection: randomly sample k chromosomes, then select the best as a parent. This method effectively balances selection pressure and diversity :contentReference[oaicite:3]index=3.

F. Crossover

Two parents undergo single-point or uniform crossover:

- *Single-point:* A random cut point is chosen; offspring inherit left genes from one parent and right genes from the other.
- *Uniform:* Each gene is randomly selected from one of the two parents :contentReference[oaicite:4]index=4.

Crossover probability p_c controls how often crossover occurs per generation.

G. Mutation

Each offspring's bits have a small probability p_m of flipping. This prevents premature convergence by introducing new genetic material :contentReference[oaicite:5]index=5. Mutation rate is typically kept low (≈ 0.01) to preserve useful building blocks.

H. Replacement and Elitism

We use generational replacement with elitism: retain the top E fittest individuals unchanged in the next generation. The rest are filled with new offspring generated via crossover and mutation :contentReference[oaicite:6]index=6.

I. Termination Conditions

The algorithm terminates when one of the following is met:

- Maximum generations reached
- Fitness threshold achieved
- No improvement over G generations :contentReference[oaicite:7]index=7

IV. Algorithm Workflow

Initialize population
for generation = 1 **to** MAX_GENS **do**
    Evaluate fitness of all individuals
    Apply elitism to preserve top E
    While next population not full:
        Select parents via tournament
        Possibly apply crossover (with p_c)
        Mutate each offspring (with p_m)

Add offspring to population
Check termination criteria
end for

V. Results

The GA successfully generated conflict-free timetables after 50–200 generations. Fitness converged steadily, with preserved diversity throughout, avoiding premature convergence :contentReference[oaicite:8]index=8.

VI. Conclusion & Future Work

This detailed GA framework demonstrates robust performance for complex timetabling tasks. Future enhancements include adaptive mutation rates, multi-objective optimization, and improved diversity management.

References