University Timetable Scheduling Using Genetic Algorithm Approach Case Study: Rajarata University OF Sri Lanka Premasiril

{.D.M.\* \*Lecturer, Department of Information and Communication Technology, Faculty of Technology, RUSL, Mihintale, Sri Lanka.}

* **Genetic Algorithm Methodology**

**1. Problem Representation:**

* **Population Structure:** Each individual in the population represents a complete timetable.
* **Chromosomes:** Each chromosome corresponds to a specific lecture hall's schedule.
* **Genes:** Each gene represents a time slot (e.g., Monday 8:00–9:00 AM) and contains information about the course, lecturer, and student group assigned.

**2. Constraints Handling:**

* **Hard Constraints (must be strictly satisfied):**
  + No overlapping courses for the same student group or lecturer.
  + Lecture halls and laboratories cannot host more than one session simultaneously.
  + Room capacities must accommodate the number of students.
  + Courses taught by visiting lecturers must be scheduled according to their availability.
* **Soft Constraints (preferable to satisfy):**
  + Lecturer preferences for time slots and lecture halls.
  + Limiting lecturers to a maximum of five lecture hours per day.
  + Minimizing student movement between distant lecture halls.
  + Distributing student workload evenly throughout the week.

**3. Genetic Operations:**

* **Initial Population:** Randomly generated timetables.
* **Fitness Evaluation:** Timetables are evaluated based on the number of satisfied constraints, with higher fitness scores assigned to those meeting more constraints.
* **Selection:** Timetables with higher fitness scores are selected as parents for the next generation.
* **Crossover:** Two-point crossover is applied to parent timetables to produce offspring, combining features from both parents.
* **Mutation:** With a low probability (2%), random changes are introduced to offspring timetables to maintain genetic diversity and explore new solutions.
* **Case Study Implementation**

**Context:** The Faculty of Technology at Rajarata University faced challenges in manual timetable scheduling due to limited resources, such as lecture halls and laboratories, and the need to coordinate with other faculties.

**Implementation Details:**

* **Programming Language:** Python
* **Population Size:** 80 timetables per generation
* **Crossover Probability:** 80%
* **Mutation Probability:** 2%

**Results:**

* The GA successfully generated feasible timetables that met all hard constraints.
* Approximately 60% of soft constraints were satisfied, indicating a significant improvement over manual scheduling methods.
* The system produced various timetable outputs, including schedules for departments, lecturers, lecture halls, and laboratories.

A Genetic Algorithm Based University Timetabling System Edmund Burke, David Elliman and Rupert Weare Department of Computer Science, University of Nottingham, University Park, Nottingham, NG7 2RD. e-mail: [ekb@cs.nott.ac.uk](mailto:ekb@cs.nott.ac.uk)

* **Genetic Algorithm Methodology**

**1. Problem Representation:**

* **Chromosome Structure:** Each chromosome represents a complete exam timetable, encoding assignments of exams to specific periods and rooms
* **Gene Encoding:** Each gene corresponds to an exam and encodes its scheduled period and assigned room, ensuring that the fundamental constraints are maintained.

**2. Constraints Handling:**

* **Hard Constraints (must be strictly satisfied):**
  + No student or invigilator is scheduled for more than one exam at the same time.
  + Room capacities must accommodate the number of students for each exam.
* **Soft Constraints (preferable to satisfy):**
  + Minimizing the number of students having exams in consecutive periods (second-order conflicts).
  + Efficient utilization of room capacities to reduce the number of unused seats.
  + Shortening the overall examination period to allow more time for marking.

**3. Genetic Operations:**

* **Initial Population:** Generated using a variation of graph coloring algorithms to ensure feasibility.
* **Fitness Evaluation:** Timetables are evaluated based on a composite fitness function that penalizes longer examination periods, second-order conflicts, and inefficient room usage.
* **Selection:** Timetables with higher fitness scores are more likely to be selected as parents for the next generation.
* **Crossover:** Combines early exams from one parent timetable with later exams from another, ensuring that offspring timetables remain feasible.
* **Mutation:** Randomly changes the period and room assignments of exams, maintaining feasibility while introducing diversity into the population.

**Case Study Implementation**

**Test Scenario:**

* **Exams:** 100 exams to be scheduled.
* **Rooms:** Four rooms with capacities of 40, 80, and two rooms with 160 seats.
* **Conflict Probability (p):** Varied between 0.2 and 0.5 to simulate different levels of exam overlap among students.

**Evaluation Function:**

The fitness function used in the study is as follows:

Where:

* **ADJ** is a constant representing the penalty for two exams that conflict being in adjacent periods.
* **Spare Seats** are only counted when there is at least one exam in the room.

**Results:**

* The GA consistently improved upon the initial randomly generated timetables across all tested scenarios.
* In cases with higher conflict probabilities (p = 0.4 and 0.5), the GA outperformed heuristic methods, producing timetables with fewer second-order conflicts and more efficient room usage.
* The system demonstrated flexibility in balancing the trade-offs between timetable length, student convenience, and resource utilization.

**User Interface and Interaction**

The system features an interactive, spreadsheet-like graphical user interface (GUI) that allows timetablers to:

* Visualize and edit timetables dynamically.
* Perform database-like queries to retrieve schedules for specific students, staff, or rooms.
* Manually adjust exam placements by dragging and dropping exam icons within the timetable grid.
* Access detailed information about exams, rooms, and scheduling constraints through intuitive interactions.

GENETIC ALGORITHM FOR UNIVERSITY COURSE TIMETABLING PROBLEM A Thesis presented in partial fulfillment of requirements for the degree of Masters of Science in the Department of Computer and Information Science The University of Mississippi by Achini Kumari Herath May2017

**Genetic Algorithm Methodology**

**1. Problem Representation:**

* **Chromosome Structure:** Each chromosome represents a complete timetable, encoding assignments of courses to specific time slots and rooms.
* **Gene Encoding:** Each gene corresponds to a course and contains information about its scheduled time slot and assigned room, ensuring that the fundamental constraints are maintained.

**2. Constraints Handling:**

* **Hard Constraints (must be strictly satisfied):**
  + No student or instructor is scheduled for more than one course at the same time.
  + Room capacities must accommodate the number of students for each course.
  + Courses must be scheduled within the available time slots and rooms.
* **Soft Constraints (preferable to satisfy):**
  + Minimizing the number of students having courses in consecutive periods.
  + Efficient utilization of room capacities to reduce the number of unused seats.
  + Distributing courses evenly throughout the week to balance workloads.

**3. Genetic Operations:**

* **Initial Population:** Generated using heuristic methods to ensure feasibility.
* **Fitness Evaluation:** Timetables are evaluated based on a composite fitness function that penalizes violations of hard constraints and rewards the satisfaction of soft constraints.
* **Selection:** Timetables with higher fitness scores are more likely to be selected as parents for the next generation.
* **Crossover:** Combines parts of two parent timetables to produce offspring, ensuring that offspring timetables remain feasible.
* **Mutation:** Randomly changes the time slot or room assignments of courses, maintaining feasibility while introducing diversity into the population.

**Case Study Implementation**

**Test Scenario:**

* **Courses:** A set of courses to be scheduled.
* **Rooms:** Multiple rooms with varying capacities.
* **Time Slots:** A predefined number of time slots available for scheduling.

**Results:**

* The GA successfully generated feasible timetables that met all hard constraints.
* The system demonstrated flexibility in balancing the trade-offs between timetable compactness, student convenience, and resource utilization.

**Future Enhancements**

The study suggests several avenues for future development:

* **Hybrid Approaches:** Combining GAs with other optimization techniques, such as local search or simulated annealing, to enhance solution quality.
* **Dynamic Constraint Management:** Incorporating real-time adjustments to accommodate last-minute changes in staff availability, room assignments, or student enrollments.
* **Scalability:** Extending the system to handle larger datasets and more complex scheduling scenarios.

UNIVERSITY COURSE TIMETABLING USING GENETIC ALGORITHMS Ozan Akı1 1 Trakya University

**Genetic Algorithm Methodology**

**1. Problem Representation:**

* **Chromosome Structure:** Each chromosome represents a complete course timetable, encoding assignments of courses to specific time slots and rooms.
* **Gene Encoding:** Each gene corresponds to a course and contains information about its scheduled time slot and assigned room, ensuring that the fundamental constraints are maintained.

**2. Constraints Handling:**

* **Hard Constraints (must be strictly satisfied):**
  + No student or instructor is scheduled for more than one course at the same time.
  + Room capacities must accommodate the number of students for each course.
  + Courses must be scheduled within the available time slots and rooms.
* **Soft Constraints (preferable to satisfy):**
  + Minimizing the number of students having courses in consecutive periods.
  + Efficient utilization of room capacities to reduce the number of unused seats.
  + Distributing courses evenly throughout the week to balance workloads.

**3. Genetic Operations:**

* **Initial Population:** Generated using heuristic methods to ensure feasibility.
* **Fitness Evaluation:** Timetables are evaluated based on a composite fitness function that penalizes violations of hard constraints and rewards the satisfaction of soft constraints.
* **Selection:** Timetables with higher fitness scores are more likely to be selected as parents for the next generation.
* **Crossover:** Combines parts of two parent timetables to produce offspring, ensuring that offspring timetables remain feasible.
* **Mutation:** Randomly changes the time slot or room assignments of courses, maintaining feasibility while introducing diversity into the population.

**Case Study Implementation**

**Test Scenario:**

* **Courses:** A set of courses to be scheduled.
* **Rooms:** Multiple rooms with varying capacities.
* **Time Slots:** A predefined number of time slots available for scheduling.

**Results:**

* The GA successfully generated feasible timetables that met all hard constraints.
* The system demonstrated flexibility in balancing the trade-offs between timetable compactness, student convenience, and resource utilization.

**Future Enhancements**

The study suggests several avenues for future development:

* **Hybrid Approaches:** Combining GAs with other optimization techniques, such as local search or simulated annealing, to enhance solution quality.
* **Dynamic Constraint Management:** Incorporating real-time adjustments to accommodate last-minute changes in staff availability, room assignments, or student enrollments.
* **Scalability:** Extending the system to handle larger datasets and more complex scheduling scenarios.

Particle Swarm Optimisation Variants and Its Hybridisation Ratios

for Generating Cost‑Effective Educational Course Timetables

**Methodology**

**1. Problem Context**

The study addresses the University Course Timetabling Problem (UCTP), an NP-hard problem that involves assigning courses to timeslots and rooms while satisfying various constraints. The objective is to minimize the total operating costs associated with course scheduling.

**2. Hybrid Particle Swarm Optimisation-Based Timetabling (HPSOT) Tool**

The researchers developed the HPSOT tool, integrating two PSO variants:

* **Standard PSO (SPSO)**
* **Maurice Clerc PSO (MCPSO)**

To enhance solution quality, they incorporated two local search operators:

* **Insertion Operator (IO):** Moves a course to a different timeslot.
* **Exchange Operator (EO):** Swaps timeslots between two courses.

Five hybridization ratios of IO to EO were tested: 100% IO, 75% IO : 25% EO, 50% IO : 50% EO, 25% IO : 75% EO, and 100% EO.

**3. Constraints Considered**

* **Hard Constraints (must be strictly satisfied):**
  + No overlapping courses for students or lecturers.
  + Room capacity and availability.
  + Lecturer and student availability.
  + Specific room requirements for courses.
  + Consecutive scheduling for multi-period courses.
* **Soft Constraints (preferable to satisfy):**
  + Assigning courses to preferred rooms to avoid additional costs.
  + Scheduling courses according to lecturer preferences to reduce hiring costs.
  + Utilizing classrooms in consecutive periods to minimize setup times.

The objective function aimed to minimize total operating costs by considering violations of soft constraints, with specific weightings assigned to each.

**4. Algorithm Workflow**

1. **Initialization:**
   * Input data (courses, lecturers, rooms, etc.) are loaded.
   * An initial population of candidate timetables is generated using a heuristic called Largest Unpermitted Period Degree (LUPD).
2. **PSO Evolution:**
   * Particles (candidate timetables) are updated using SPSO or MCPSO equations.
   * A random key technique is applied to determine the sequence of course assignments.
3. **Local Search Hybridization:**
   * After PSO updates, IO and EO are applied according to the specified hybridization ratio to refine solutions.
4. **Repair Mechanism:**
   * Infeasible timetables violating hard constraints are repaired by reassigning conflicting courses to feasible timeslots.
5. **Evaluation and Selection:**
   * Timetables are evaluated based on the objective function.
   * The best solutions are selected for the next iteration.
6. **Termination:**
   * The process repeats until a stopping criterion (e.g., maximum iterations) is met.

**Results**

**1. Experimental Setup**

* **Data:** Eleven real-world timetabling instances from Thai universities.
* **Metrics Evaluated:**
  + Minimum, maximum, and mean total operating costs.
  + Standard deviation of costs.
  + Computational time.
  + Convergence speed.

**2. Key Findings**

* **Performance Improvement:** Hybrid PSO variants outperformed their conventional counterparts (SPSO and MCPSO) across all instances in terms of lower operating costs and faster convergence.
* **Optimal Hybridization Ratio:** The 75% IO : 25% EO hybridization ratio yielded the best results for most instances, balancing exploration and exploitation effectively.
* **Statistical Significance:** ANOVA tests confirmed that the improvements achieved by hybrid methods were statistically significant with a 95% confidence interval.
* **Computational Efficiency:** Hybrid methods reduced computational time by up to 60% compared to conventional PSO variants, attributed to more efficient search processes.
* **Convergence Behavior:** Hybrid approaches demonstrated faster convergence to high-quality solutions, particularly in larger problem instances.

A simulated annealing algorithm for the faculty-level university course

timetabling problem

**Problem Overview**

This study tackles a multifaceted course timetabling problem characterized by:

* **Double Major and Minor Program Constraints**: Ensuring that students enrolled in multiple programs do not face scheduling conflicts.
* **Shared Classrooms**: Managing the allocation of classrooms shared among various faculties.
* **Instructor Availability**: Accommodating the time preferences and availabilities of instructors.

**Methodology**

The authors propose a two-pronged approach:

1. **Goal Programming Model**: A mathematical model aiming to minimize deviations from seven specific scheduling goals, including:
   * Avoiding classes during lunch breaks.
   * Preventing consecutive classes that span lunch periods.
   * Ensuring non-overlapping compulsory courses for consecutive student groups.
   * Limiting daily course hours for students.
   * Eliminating scheduling conflicts for double major and minor program students.
2. **Simulated Annealing (SA) Algorithm**: Due to the computational complexity of the goal programming model for large instances, a simulated annealing algorithm is developed. Key features include:
   * **Solution Representation**: A matrix format where rows represent classrooms and columns represent time periods, facilitating the tracking of course assignments.
   * **Initial Solution Generation**: Prioritizing courses with larger student enrollments and assigning them to classrooms with appropriate capacities.
   * **Neighborhood Structures**: Employing both simple move (relocating a course to a different time slot) and swap (exchanging time slots between two courses) strategies to explore the solution space.
   * **Cooling Schedule**: Implementing a cooling rate of 0.05 with a final temperature set to zero, allowing the algorithm to converge effectively

**Results**

The proposed methods were evaluated through:

* **Sample Problem**: Demonstrated that the SA algorithm could achieve optimal solutions more quickly than the goal programming model solved via GAMS/Cplex.
* **Randomly Generated Test Problems**: For larger instances, the goal programming model failed to find feasible solutions within the time limit, whereas the SA algorithm successfully generated high-quality timetables in reasonable computational times.
* **Real-Life Case Study**: Applied to an engineering faculty with 107 courses, 53 instructors, and 31 classrooms, the SA algorithm produced a timetable with an 83% improvement over the existing schedule, completing the task in approximately 149 seconds.