**Mainstream Programme – Computer Science Dept.**

**EAs Term Group–Project (COVER SHEET)**

**Discussions Scheduled for Week 14** *(Thursday, May 14th, 2025)***.**

**30**

* Print this cover sheet and attach it to a printed copy of the documentation.
* Please write all your names in Arabic & ensure your students’ IDs are correct.
* Handwritten Signatures for the attendance of all team members should be filled in before the discussion.
* Please attend the discussion on time *(announced separately)*.

**ProjectName:** 2 [Exam Timetabling Optimization using Genetic Algorithms and/or Ant Colony Optimization **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Team Information *(typed, not handwritten, except for the attendance signature)*:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **ID**  **[Ordered by ID]** | **Full Name**  **[In Arabic]** | **Attendance**  **[Handwritten Signature]** | **Final Grade** |
| **1** | 20220225 | سيف ممدوح ياسين |  |  |
| **2** | 20220307 | عمار هشام عبدالملك |  |  |
| **3** | 20220365 | مازن نبيل عبدالغني |  |  |
| **4** | 20220434 | محمد وحيد خيري |  |  |
| **5** | 20220514 | مهند جمال محمد |  |  |

| **Category** | **Description** | **Points** | **Grade** |
| --- | --- | --- | --- |
| Problem Understanding | Clear constraints identification, student overlap and scheduling dynamics | 4 |  |
| Algorithm Design | Proper GA or ACO structure, fitness function, conflict minimization | 7 |  |
| Implementation | Working system to generate conflict-free timetables | 6 |  |
| Results & Analysis | Effects of pheromone/crossover/mutation strategies across experiments | 8 |  |
| Presentation & Report | UI output, analysis documentation, literature review | 5 |  |
| **Total** |  | **30** |  |

**Final Report: Exam Scheduling Using Genetic Algorithm (GA)**

#### 1. **Introduction and Overview**

The exam scheduling problem is a combinatorial optimization task that aims to allocate courses to time slots and rooms in such a way that all constraints (e.g., conflicts, room capacity, instructor availability) are met while minimizing scheduling conflicts. This problem is complex and difficult to solve using traditional methods, particularly as the scale of the problem increases. In this project, we utilize the **Genetic Algorithm (GA)** to generate optimal exam schedules while considering multiple constraints.

The objective is to create an efficient exam timetable that minimizes student conflicts (i.e., students who have overlapping exams), room scheduling conflicts, and invigilator availability issues. The GA is an ideal choice for this task due to its ability to explore large search spaces and find near-optimal solutions.

#### 2. **Proposed Solution**

We propose a solution based on **Genetic Algorithm (GA)**, which is a population-based heuristic search technique inspired by the process of natural selection. In the context of exam scheduling, the GA aims to evolve better solutions through selection, crossover, and mutation operations. The problem involves assigning courses to time slots, rooms, and invigilators while satisfying the constraints of the problem.

**Key Constraints**:

**Hard Constraints:**

**1: An exam will be scheduled for each course**

**3: A student can not give more than one exam at a time**

**4: All exams must be held between 9 AM and 5 PM**

**5: A teacher can not invigilate two exams at the same time**

**6: A teacher can not invigilate two exams in a row**

**7: No exam scheduled in the same room at the same time**

**Soft Constraints:**

**1: An exam will be scheduled for each course**

**2: A student shall not give more than one exam consecutively**

#### 3. **Applied Genetic Algorithm (GA) Approach**

The Genetic Algorithm (GA) for exam scheduling works as follows:

1. **Initial Population**: The population consists of a set of potential schedules (chromosomes). Each chromosome represents a timetable where each gene corresponds to a specific exam and its associated room, time, and invigilator.

A screen shot of a computer screen

AI-generated content may be incorrect.

**A screen shot of a computer program

AI-generated content may be incorrect.**

1. **Fitness Function**: The fitness of a chromosome is evaluated based on how well it satisfies the constraints. The fitness function is designed to penalize solutions that violate constraints such as room conflicts, student conflicts, and invigilator availability.
   * **Penalty for Conflicts**: A chromosome that has scheduling conflicts will incur a penalty, reducing its fitness score.

A screen shot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

A graph with a line going up

AI-generated content may be incorrect.

1. **Selection**: The selection process chooses the best chromosomes based on their fitness values. The roulette wheel selection or tournament selection method can be used to probabilistically select candidates for the next generation.

A screen shot of a computer program

AI-generated content may be incorrect.

1. **Crossover (Recombination)**: Two parent chromosomes are combined using a crossover operator to create new offspring. The crossover operator exchanges portions of the two parent schedules to produce new timetables.

A screen shot of a computer program

AI-generated content may be incorrect.

1. **Mutation**: After crossover, some chromosomes undergo mutation, where random changes are made to the timetable (e.g., swapping the time slots or rooms of two exams) to introduce diversity and avoid local optima.

A screen shot of a computer code

AI-generated content may be incorrect.

#### 4. **Experiments and Solution**

We implemented the solution using **Python** and employed the **DEAP** (Distributed Evolutionary Algorithms in Python) library to facilitate the implementation of the genetic algorithm. The program starts by reading input data, including exam details, room availability, and student registrations.

The process of generating the timetable begins with creating an initial population of random solutions. Each solution is evaluated based on how well it satisfies the constraints, and the best solutions are selected for the next iteration of the algorithm. The algorithm iterates through several generations, gradually improving the quality of the solutions.

**Visual Results**:

At each generation, we visualize the exam schedule to better understand how the algorithm evolves. Below is a sample of the final exam schedule output:

A screenshot of a computer

AI-generated content may be incorrect.

A close-up of a computer screen

AI-generated content may be incorrect.

This visual result shows the final assignments of courses to rooms, times, and teachers. It also ensures that constraints like room availability and student conflicts are satisfied. For larger data sets.

#### 5. **Analysis, Discussion, and Future Work**

**Analysis**:

The GA is effective at handling the complexities of the exam scheduling problem, generating schedules that respect the constraints and optimizing the use of available resources. The algorithm converges toward a solution after several generations, with solutions improving incrementally based on the feedback from previous generations.

**Discussion**:

While the GA performs well in generating feasible schedules, there are several areas where further improvements could be made:

1. **Scalability**: The GA approach works well for small- to medium-sized problems, but as the number of exams, rooms, or students increases, the time to find a satisfactory solution may grow exponentially. Parallelizing the algorithm could help improve efficiency for large-scale problems.
2. **Fine-Tuning Parameters**: The effectiveness of the GA is heavily influenced by parameters such as population size, crossover rate, and mutation rate. These parameters need to be fine-tuned to ensure that the algorithm converges to a good solution without premature convergence.
3. **Constraint Handling**: While the GA is capable of finding good solutions, more sophisticated constraint handling techniques (e.g., soft and hard constraints) can be employed to improve the quality of the solution further.

**Future Work**:

* **Integration of Local Search**: After running the GA, a local search algorithm (e.g., simulated annealing) can be used to refine the solutions, improving their quality.
* **Dynamic Scheduling**: Extend the solution to handle dynamic scenarios where new courses or exams are added mid-semester.
* **Real-Time Conflict Resolution**: Develop a system that allows real-time conflict resolution when scheduling changes need to be made after the initial timetable has been generated.
* **User Interface**: Develop a user-friendly interface for administrators to input data, visualize results, and modify schedules interactively.

**Literature Review: Exam Scheduling Algorithms Using Genetic Algorithms**

**Introduction**

Exam scheduling is a complex and critical problem faced by educational institutions worldwide. The goal is to assign exams to time slots and rooms while satisfying numerous hard and soft constraints, such as avoiding clashes for students, respecting room capacities, and minimizing consecutive exams for students or invigilators.

**Traditional Approaches**

Early approaches to exam scheduling used heuristic and rule-based algorithms. These methods rely on domain-specific rules and often require manual tuning. Though straightforward, they struggle with scalability and handling complex constraints.

**Genetic Algorithms (GA) in Exam Scheduling**

Genetic algorithms, inspired by natural selection and genetics, have emerged as powerful metaheuristic techniques to solve scheduling problems:

* Flexibility: GAs can handle multiple conflicting constraints by encoding possible schedules as chromosomes and evolving solutions over generations.
* Robustness: They explore a wide solution space and are less likely to get stuck in local optima compared to greedy or simple heuristics.
* Adaptability: GAs can be customized with problem-specific crossover, mutation, and selection operators.

**Key Studies**

* Burke et al. (1996) demonstrated the effectiveness of GAs in university exam timetabling, introducing specialized crossover and mutation operations to preserve feasibility.
* Evening (1995) showed that hybrid approaches combining GAs with local search heuristics improved convergence speed and solution quality.
* Qu et al. (2015) proposed a multi-objective GA to simultaneously optimize hard constraints (no conflicts) and soft constraints (student convenience), using a fitness function balancing these goals.
* Malkawi and Shatnawi (2018) introduced a GA with adaptive mutation rates to better explore the search space and avoid premature convergence in large-scale exam scheduling.

**Challenges and Improvements**

* Constraint Handling: Ensuring hard constraints are never violated remains challenging; many works embed penalty functions or repair mechanisms in fitness evaluation.
* Scalability: Large institutions with thousands of students and exams require efficient representations and operators to keep computation manageable.
* Hybridization: Combining GAs with other optimization methods (e.g., simulated annealing, tabu search) is a growing trend to leverage complementary strengths.

**Conclusion**

Genetic algorithms offer a promising and flexible framework for exam scheduling, balancing solution quality with computational efficiency. Continuous advancements in hybrid methods, constraint handling, and adaptive operators enhance their applicability to real-world academic timetabling problems.