Exam Scheduling via Graph Colouring

Graph Colouring Project — Bonus Application

Problem Overview

University examination scheduling is a classical constraint satisfaction problem: given a set of courses and student enrollments, assign each exam to a timeslot such that **no student has two exams simultaneously**. This is a direct application of the **graph colouring problem**.

Reduction to Graph Colouring

The exam scheduling problem reduces to graph colouring as follows:

- **Vertices** $\rightarrow$ Courses (each course requiring an exam)
- **Edges** $\rightarrow$ Conflicts (two courses share at least one student)
- **Colours** $\rightarrow$ Timeslots (non-overlapping exam periods)

Formally, let $C = \{c_1, c_2, \ldots, c_n\}$ be courses and $S = \{s_1, s_2, \ldots, s_m\}$ be students. Define the *conflict graph* $G = (V, E)$ where:

$$V = C, \quad E = \{(c_i, c_j) : \exists s_k \in S \text{ enrolled in both } c_i \text{ and } c_j\}$$

A valid k-colouring of G yields a conflict-free schedule using k timeslots.

Connection to the Project

Our `exam_scheduler.py` application directly implements the graph colouring algorithms from this project:

- **DSatur Algorithm**: The primary solver, using saturation degree ordering for efficient greedy colouring. Ideal for real-time scheduling of moderate-sized instances.
- **Exact Backtracking**: For small graphs (< 15 courses), finds the optimal chromatic number via pruned search.

The application provides a graphical interface where users:

1. Add courses to the system
2. Register students by selecting their enrolled courses (checkboxes)
3. Click "Generate Schedule" to compute a conflict-free timetable

Why This Matters

Aspect	**Significance**
NP-Hardness	Optimal scheduling is NP-complete; heuristics essential
Real-World Use	Universities worldwide solve this problem each semester
Algorithm Selection	DSatur balances speed and solution quality
Chromatic Number	Minimum timeslots = $\chi(G)$ of the conflict graph

Example

Consider 4 courses (A, B, C, D) with students: Alice{A, B}, Bob{B, C}, Carol{C, D}.

- Conflict edges: (A,B), (B,C), (C,D)
- Path graph P_4: chromatic number $\chi = 2$
- Solution: Slot 1 = {A, C}, Slot 2 = {B, D}

This demonstrates how the abstract problem of graph colouring directly solves a practical scheduling challenge, validating the algorithms studied throughout this project.