Graphs, cont.

A university needs to schedule final exams for its large classes. There are 3 possible locations for exams and 10 possible timeslots. In addition some classes cannot have exams at the same time because they share students. How can the university perform the scheduling?

timeslots don't overlap

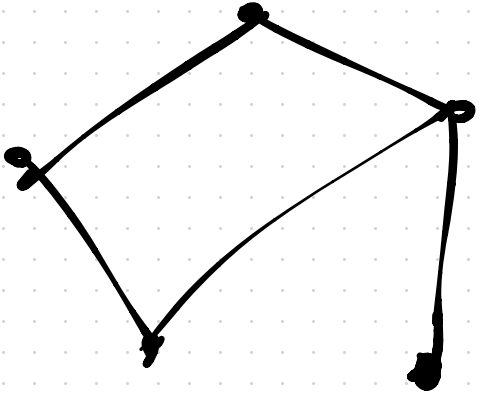
Need additional data:

- Which classes can't overlap (i.e. share a student)
- Which classes there are

Can represent this data as a graph.

Vertices = classes

An edge between two classes if they share a student.



Need to assign each vertex a time and room.

If two classes are adjacent, then they can't share a time.

"Color" each vertex with one of 10 possible times.
No two adjacent vertex can have the same color.
We require no more than 3 of a single color.

IP to solve the problem.

For each class i, and each time j

$$x_{ij} = \begin{cases} 1 & \text{if class } i \text{ is at time } j \\ 0 & \text{otherwise.} \end{cases}$$

Max 0

s.t. $x_{ij} = 0$ or 1 for all i, j

For all i, $\sum_{j=1}^{10} x_{ij} = 1$

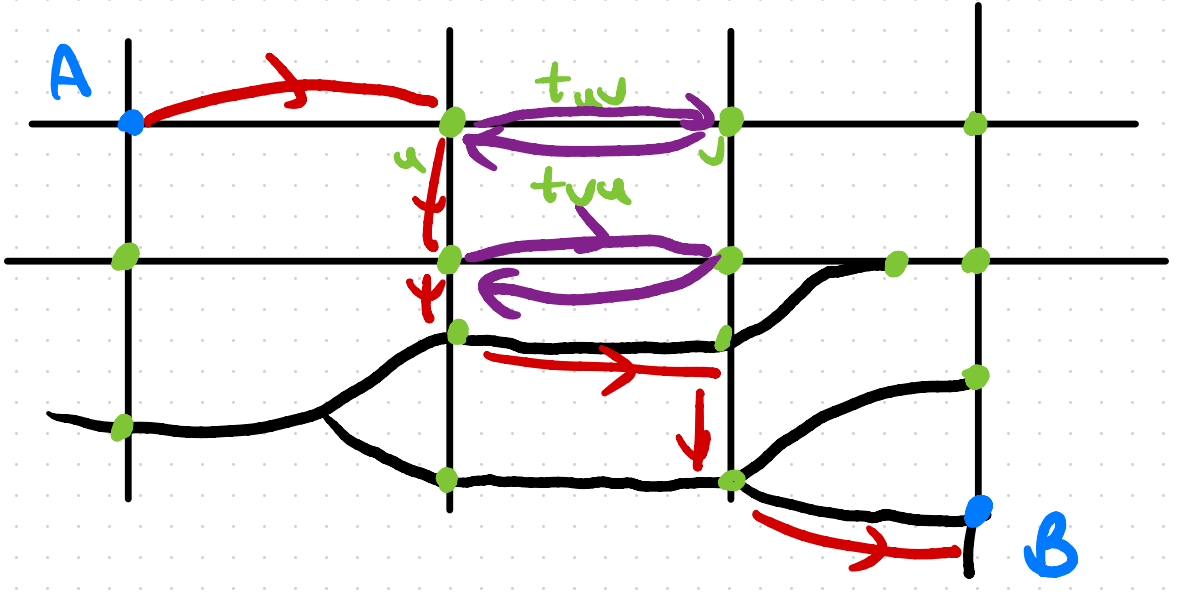
For every edge uv, and every time j

$$x_{uj} + x_{vj} \leq 1$$

For every time j

$$\sum_{i \text{ class}} x_{ij} \leq 3$$

Suppose we want to program a GPS to find the shortest driving time from point A to point B.



Consider a map of roads.

Place a vertex at each intersection.

For any two adjacent vertices u, v, let t_{uv} be the time it takes to drive from u to v.

We'll make the graph directed, with an edge between u and v if we can drive from u to v without going through another intersection.

We now find the shortest path from A to B through integer programming.

For each directed edge uv,

$$x_{uv} = \begin{cases} 1 & \text{if we use } uv \text{ in our path} \\ 0 & \text{otherwise} \end{cases}$$

constants

$$\text{Min} \sum_{uv \text{ edge}} t_{uv} x_{uv}$$

s.t. $x_{uv} = 0$ or 1 for all edges uv

Given a vertex u, let $\delta^-(u)$ be the set of vertices v with an edge vu
Let $\delta^+(u)$ be the set of vertis with an edge uv

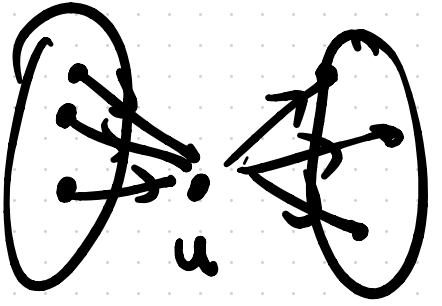
For every vertex u other than A and B,

$$\sum_{v \in \delta^-(u)} x_{vu} = \sum_{v \in \delta^+(u)} x_{uv}$$

and

$$\sum_{v \in \delta^+(A)} x_{Av} = 1$$

$$\sum_{v \in \delta^-(B)} x_{vB} = 1$$



in-neighborhood

out-neighborhood

In practice, one would use a specialized algorithm like Dijkstra's algorithm, rather than IP, to find the shortest path.

Fact: If you solve the above IP as an LP instead (and require vertex solutions), then you will always return an integer solution no matter what the t_{ij}'s are! This means the shortest path problem can be solved as a LP instead of an IP.