Timetabling

Conor Newton

October 30, 2019

Introduction

What is a timetable?

- Given a set of activities, a set of rooms, and a range of times.
- A timetable is a choice of timeslot and room for each activity

Introduction

In practice, we have many more requirements than this.

Some other things to consider:

- Students
- Staff
- Room capacity
- Equipment Requirements

Hard & Soft Constraints

Hard-Constraints

- Our solution should have zero hard-constraint violations.

Soft-Constraints

- We aim to minimize the number of soft-constraint violations.

Hard & Soft Constraints Examples

Examples of Hard-Constraints

- Only one activity in a room at a time.
- The capacity of a chosen room must be greater than the number of students on an activity
- Equipment Requirements

Examples Soft-Constraints

- Student activity clashes
- Lunch Breaks
- Preferred rooms

Our Approach

- Stage 1 - Generate initial solution satisfying the hard constraints
- Stage 2 - Continuously make improvements to our existing solution

Stage 1 - Backtracking

To generate our initial solution we make use of a backtracking algorithm.

- Incrementally builds a solution, by looping through the activities and assigning them timeslots & rooms.
- When the solution cannot be extended any further it will "backtrack"

Stage 2 - Optimization

What we need?

- An initial solution.
- Objective function.
- Neighbourhood operators.
- Optimization algorithm (Simulated Annealing/Tabu Search).

Stage 2 - Optimization (Objective function)

- An objective function gives a score to a timetable (lower is better).
- It should consider soft-constraint violations

$$\text{obj}(A) = w_1 \cdot f_1(A) + w_2 \cdot f_2(A) + w_3 \cdot f_3(A)$$

Stage 2 - Optimization (Neighbourhood operators)

What are neighbourhood operators?

- Allows us to move to a different solution.

We make use of two neighbourhood operators

- Simple Swaps
- Kempe Swaps

Both are commonly used together for timetabling problems.

Stage 2 - Optimization (Neighbourhood operators)

Two kinds of Simple Swaps

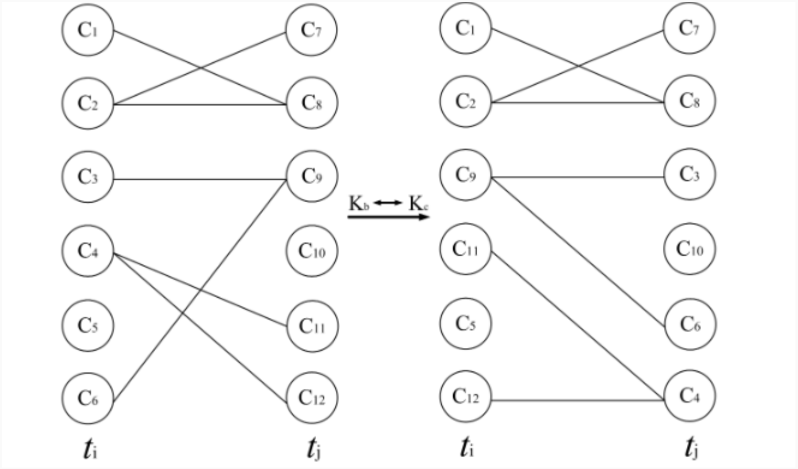
- Choose two random activities
- Swap their time and location

Alternatively,

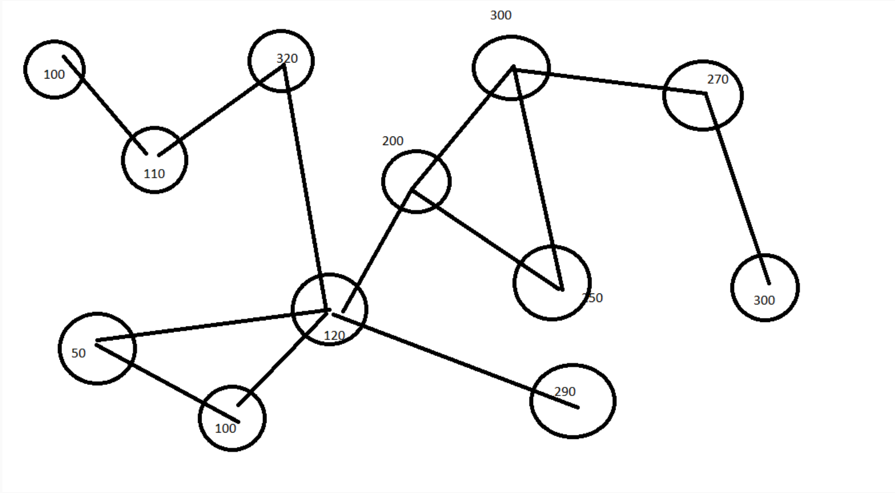
- Choose a random activity and new free timeslot and room
- Move an activity to this room and timeslot

Stage 2 - Optimization (Neighbourhood operators)

Kempe Swaps are a bit more involved...



Stage 2 - Optimization (Solution Space)



Stage 2 - Optimization (Simulated Annealing)

What is simulated annealing?

- A metaheurstic for finding the global minimum of a function.
- Uses probability to avoid getting stuck in local minimums.

How it works:

1. We choose a random neighbouring timetable.
2. If it's has a lower score, we set it as our current solution.
3. Otherwise, we set it as our current solution with probability $\exp(\frac{current\_score - new\_score}{temp})$
4. We set $temp = temp * cooling\_rate$
5. We repeat these steps until *temp* falls below a value.

Results & Implementation

- Implemented in C++
- Produces a solution with an objective score of < 1000 in less than 10 minutes.
- Produces a solution with an objective score of < 100 after 35 minutes.
- The data set had about 18000 students, 3000 activities, 600 rooms.
- Used a list of constraints that represented the university's requirements.
- There are still many more improvements for to be made!