A Case Study
Timetabling with Memetic Algorithms

Content
- The Problem
- Representation
- Memetics
- Methods and Operators
- Conclusions

Timetabling
The Napier Timetabling Challenge:
- 2000 Events
- 45 Timeslots
- 183 Rooms
- 669 Lecturers
- 978 Student groups

We want to produce
- Timetables that work:
  - people don't have to be in two places at once, only one class in a room, room have correct equipment etc.
- Timetables which are ‘good’ e.g.
  - Students don’t get a single class on a day
  - Minimum movement between classes
  - Best use of rooms
Constraints - Two types

- **Hard Constraints**
  - Those which if broken make the timetable infeasible
  - E.G. A person cannot be in two places at once

- **Soft Constraints**
  - Those we prefer not to be broken
  - E.G. A lecturer should have a teaching free day

---

### Instantaneous Site Changes

This counts the number of people expected to make instantaneous site changes. These should be avoided because people do not have time to change sites.

### Site Changes

The site changes are the number of people changing sites (from one class to another) over the week.

### Location Changes

Location changes counted are the number of people moving from one part of a building to another over the week.
Room Changes

Room changes are the number of people changing from one classroom (or lab) to another classrooms over the week.

Single Classes on a Day

This counts the number of students having a single class on a day. This should be avoided as it is inconvenient for the students.

Gaps in Student Day

This counts the number of students having a gap of four hours or more between classes on the same day. This should be avoided as it is inconvenient for the students.

Overloaded Students

This counts the number of students with more than two intensive classes (e.g. lectures but not labs) in a row.
Overloaded Lecturers

This counts the number of lecturers with three or more hours of teaching in a row.

No Teaching Free Day

This counts the number of lecturers who do not have at least one day when they have no teaching. Lecturers need whole days free to concentrate on administration and research.

Wednesday Afternoon Classes

This counts the number of students who have a class on a Wednesday afternoon. This time is traditionally reserved for sports.

Five O’clock Classes

This counts the number of students with classes between 5.00 and 6.00pm. This is the last hour of the day and generally unpopular, so its use is avoided if possible.
Multi-Objective Optimisation

- Users can select targets for each of the objectives
- Users can select weights for each of the targets
- The weight specifies how important it is to reach the target

Building Timetables

A simple direct representation

We can represent a candidate solution in the following way:
A permutation of the events

| 3 | 6 | 4 | 8 | 7 | 1 | 2 | 5 | 9 | 0 |

A list of timeslots - one for each event

| 13 | 23 | 43 | 1 | 12 | 10 | 34 | 34 | 6 | 0 |

Producing Timetables

- We go through the events in the order specified by the permutation
- An attempt is made to place each event in the timeslot specified in the timeslot list
- If the event cannot have the resources it requires at that timeslot (because some other event has used then) then it remains unplaced.
Making a New Individual

- Permutation initialised by a heuristic:
  Those events with fewer possible slots are considered first
- Timeslot list is initialised randomly from the list of possible slots for each event
  (those in which the event could take place if there were no other events)

Comparing Candidate Solutions for Contests

- If one has fewer unplaced events than the other then it wins the contest
- If they have the same then look at how good the timetables are. The best one wins.

Operators

- Recombination is a uniform crossover of the timeslot list
- The permutation is inherited from one parent
- Binary tournament selection
- Replace oldest
- Steady state population with elitism

The Evolution Process

- Selection
- Recombination
- Mutation
- Replacement
Results

Typical Run

Acceptable time approx 1 min (on Pentium III 500MHz)

Results not Good

- Timetables are evolved and get better and better.
- But not good enough in a reasonable time
- Something better is needed

A Better Method - Memetic Algorithms

- Early work by Pablo Moscato and Mike Norman
- Memetic algorithms are based on cultural evolution rather than biological evolution.
- The evolution of memes rather than genes..
- “Meme” – Richard Dawkins’ name for the unit of cultural transmission

Genetic Vs Memetic Evolution

- Similar
  - Ideas are created
  - Good ideas are more likely to survive than bad ones
  - Ideas can be combined together
  - Ideas can mutate through misunderstanding
- Different
  - Ideas are improved by those that hold them
- A memetic algorithm is an evolutionary algorithm with some added technique to improve the genetic material
Addition to Representation

A permutation of the events

\[3, 6, 4, 8, 7, 1, 2, 5, 9, 0\]

A list of timeslots - one for each event

\[13, 23, 43, 1, 12, 10, 34, 34, 6, 0\]

A list of reserve timeslots - one for each event

\[45, 2, 33, 2, 7, 17, 44, 52, 9, 8\]

The reserve timeslots come from the parent not contributing the main timeslot.

Producing Timetables

- We go through the events in the order specified by the permutation
- An attempt is made to place each event in the timeslot specified in the timeslot list
- If the event cannot have the resources it requires at that timeslot (because some other event has used then) then we try the reserve timeslot
- If this doesn’t work then we try all the other timeslots
- The representation is now indirect

Memetic Results

Typical Run

Penalty Points in One Hour with Varying Local Search

Acceptable time approx 1 min (on Pentium III 500MHz)
Using Problem Specific Knowledge

- Adding the memetic search greatly increases the power of the algorithm by making use of problem specific knowledge.
- There are other pieces of problem specific knowledge that we can make use of to introduce heuristic mutation operators.

Targeted Mutation

- Some events cause particular problems in a timetable.
  - E.G. An event might take place on the only day in which a lecturer has no other teaching.
- Targeted mutation increases the likelihood that the genetic material for these events will mutate.

How Mutation is Targeted

- During evaluation each event is given a penalty score based on its contribution towards problems in the timetable.
- The probability of an event’s genetic material being mutated is then increased by an amount proportional to the square of its penalty.

Mutation

- When a child is created the timeslot for each event has a chance of mutating.
  - This chance is normally \( \frac{1}{\text{number of events}} \).
- Three mutation operators - equal chance of being used.
  - One blind mutation operator
    - simply reinitialise the timeslot.
- Two directed mutation operators
  - Selfish Mutation
  - Co-operative Mutation
Selfish Mutation

- An event finds another event which currently has a timeslot which it could use.
- It then steals the slot by copying it to its position in the chromosome - and ensures that it gets priority over the slot by placing itself in front of the other event in the permutation.

Co-operative Mutation

- The event finds another event such that each could use the other’s slot
- The two events then move to the first and second positions in the permutation - ensuring that they get to use their new slot.

Combined Directed Mutation and Targeted Mutation

Conclusions

- Real timetabling can be achieved using evolutionary methods
- Direct representations don’t work
- Heuristic operators are beneficial