CSG2341
Intelligent Systems

Genetic Algorithms for Scheduling

Scheduling Problems

- Good candidates for an evolutionary algorithm
  - high value,
  - complex,
  - difficult to approach analytically,
  - often NP-complete

GA design steps

- **Step 1:** Specify the problem, define constraints and optimum criteria.
- **Step 2:** Represent the problem domain as a chromosome.
- **Step 3:** Define a fitness function to evaluate the chromosome performance.
- **Step 4:** Construct the genetic operators
- **Step 5:** Run the GA and tune its parameters.

Step 1: Specify the problem, define constraints and optimum criteria

- **The problem constraints:**
- The maximum loads expected during four intervals are 80, 90, 65 and 70 MW;
- Maintenance of any unit starts at the beginning of an interval and finishes at the end of the same or adjacent interval. The maintenance cannot be aborted or finished earlier than scheduled;
- The net reserve of the power system must be greater or equal to zero at any interval.

The optimum criterion is the minumum of the net reserve at any maintenance period.

Data for the scheduling problem

Unit number	*Unit capacity, MW*	*Number of intervals required for unit maintenance*
1	20	2
2	15	2
3	35	1
4	40	1
5	15	1
6	15	1
7	10	1

See ScheduleProblem.java

Step 2: Represent the problem domain as a chromosome.

Unit 1:	1 1 0 0	0 1 1 0	0 0 1 1	
Unit 2:	1 1 0 0	0 1 1 0	0 0 1 1	
Unit 3:	1 0 0 0	0 1 0 0	0 0 1 0	0 0 0 1
Unit 4:	1 0 0 0	0 1 0 0	0 0 1 0	0 0 0 1
Unit 5:	1 0 0 0	0 1 0 0	0 0 1 0	0 0 0 1
Unit 6:	1 0 0 0	0 1 0 0	0 0 1 0	0 0 0 1
Unit 7:	1 0 0 0	0 1 0 0	0 0 1 0	0 0 0 1

Unit 1	Unit 2	Unit 3	Unit 4	Unit 5	Unit 6	Unit 7
0 1 1 0	0 0 1 1	0 0 0 1	1 0 0 0	0 1 0 0	0 0 1 0	1 0 0 0

Step 3: Define a fitness function to evaluate the chromosome performance.

The optimum criterion is the maximum of the net reserve at any maintenance period.

See ScheduleEvaluator.java

Step 4: Construct the genetic operators

- Examine DiscreteEvolvable code for this

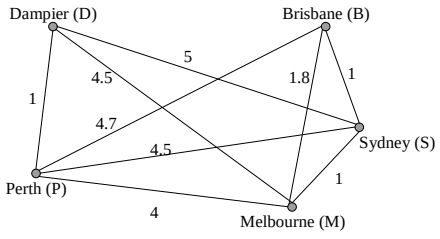
Step 5: Run the GA and tune its parameters.

- Let's try it!

Another Case Study – The Traveling Salesperson Problem

- Imagine a salesman who
  - has a number of cities to visit, and
  - wants to visit each of them once in such an order that
  - his round trip is as short as possible.
- An ordering of cities to visit is called a tour.
- Many real scheduling problems, such as scheduling of deliveries or crew rostering, turn out to be equivalent to the Travelling Salesman Problem (TSP).

TSP example



TSP with a GA

- Possible chromosome:
- 2 3 5 6 1 4 8 7 (a permutation of the city numbers)

TSP operators – mutation (2-opt)

A = 2 3 **5 6 1 4** 8 7

x x

A = 2 3 **4 1 6 5** 8 7

What does this mean in terms of tours?

TSP operators – crossover (PMX)

A = 2 3 **5 6 1** 4 8 7

x x

B = 7 1 **2 3 8** 6 5 4

A' = 5 6 **2 3 8** 4 1 7

B' = 7 8 **5 6 1** 3 2 4

Why not use normal crossover? What does this mean in terms of tours?

TSP operators

- See PermutationEvolvable.java
- Have a quick look at the rest of the TSP solution
- Try it!