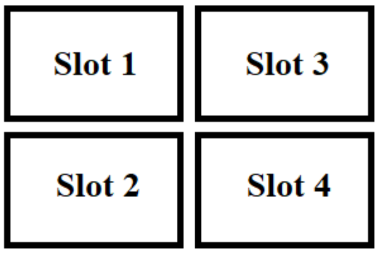
Artificial Intelligence 2020/2021

Exercise Sheet 4: Optimization/Meta-Heuristics

4.1 Timetabling Problem Solving using Local Search/Simulated Annealing

Suppose that you have access to the information on students enrolled in various elective subjects of a master's/doctoral program. Each subject has only one weekly class. We want to build a schedule/timetable using only a specified number of slots (available hours) and minimizing the number of incompatibilities for students (*i.e.*, subjects that enrolled students will not be able to attend because of temporal overlaps in the respective schedules). To do this, the class of each subject must be assigned to a slot (from the available ones). Consider the example problem represented below:



Facts Representation

```
slots(4).
disciplines(12).
students(12).
discipline(1, [1,2,3,4,5]).  % Students 1,2,3,4,5
                             % enrolled in discipline 1
discipline(2, [6,7,8,9]).
discipline(3, [10,11,12]).
discipline(4, [1,2,3,4]).
discipline(5, [5,6,7,8]).
discipline(6, [9,10,11,12]).
discipline(7, [1,2,3,5]).
discipline(8, [6,7,8]).
discipline(9, [4,9,10,11,12]).
discipline(10, [1,2,4,5]).
discipline(11, [3,6,7,8]).
discipline(12, [9,10,11,12]).   %Students 9,10,11,12
                                % enrolled in discipline 12
```

Text File Representation

```
4 12 12      // Problem with 4 Slots and 12
             // Disciplines and 12 students
1 2 3 4 5    // Students 1,2,3,4,5 enrolled in discipline 1
6 7 8 9
10 11 12
1 2 3 4
5 6 7 8
9 10 11 12
1 2 3 5
6 7 8
4 9 10 11 12
1 2 4 5
3 6 7 8
9 10 11 12    // Students 9,10,11,12
              // enrolled in discipline 12
```

a) Define a means of representation of a solution schedule and create a method that allows you to randomly create a solution.

b) Build a function (on a language of your choice) that allows you to calculate an incompatibilities table, that is, for each pair of subjects (discipline) calculates the number of students who are enrolled in both.

c) In this problem, the simplest representation for a solution (assignment of disciplines to slots) consists of associating, to each discipline 1..*nd*, a slot 1..*ns*, where *nd* is the number of disciplines and *ns* is the number of slots. For this purpose, we can use a list of integers *nd*, whose values are numbers from 1 to *ns*. The list index identifies the respective discipline, and the value inside the list represents the slot to which it has been assigned. For example, in list [4,1,2,3,2,4,1,1,2,2,2,3] discipline 1 has been assigned to slot 4, such as discipline 6. Implement a function that allows you to evaluate a particular solution, by calculating the total number of students enrolled in overlapping subjects.

d) Define one or more neighboring spaces and neighbor functions capable of calculating the neighbors of a solution.

e) Build a program that allows you to use the following methods to find the optimal (or sub-optimal) solution to this problem:
  - Hill Climbing (multiple versions)
  - Simulated Annealing

f) Consider the following initial solutions and try the various methods developed:

```
initial([1,1,1,1,1,1,1,1,1,1,1,1]).   % Solution with equal values
initial ([1,1,1,2,2,2,3,3,3,4,4,1]).  % Almost the solution?
initial ([1,1,4,2,2,2,3,3,3,4,4,1]).  % Local Minimum?
initial ([1,2,3,4,1,2,3,4,1,2,3,4]).   % What now? Another local minimum?
```

g) Try other initial solutions; use a random initial solution and the "Random Restarts" method.

h) Compare the implemented methods in terms of the quality of the obtained solution and the time it takes to obtain the solution, from the examples given.

i) Try creating several instances of the problem, with different dimensions (varying the number of disciplines and slots) and difficulties (varying the students enrolled in the disciplines).

4.2 Timetabling Problem Solving with Genetic Algorithms

Continuing the problem of the previous exercise, build a program (in a language of your choice) that allows you to apply Genetic Algorithms to find the optimal (or sub-optimal) solution. To this end, address the following issues:

a) Define the method for representing individuals (solutions) and create a method that allows you to randomly create an individual (solution). *Note: This exercise should have already been solved in exercise 4.1.*

b) Conveniently define a fitness function (evaluation function). *Note: This exercise should have already been solved in exercise 4.1.*

c) Define one or more crossover operators that allow creating new elements by combining individuals from the population.

d) Define one or more parameterizable selection methods, including tournament and roulette methods.

e) Define one or more mutation operators. *Note: This exercise should already be solved in exercise 4.1.*

f) Define the central engine of application of genetic algorithms, that starts by creating an initial population with N elements, and during G generations evaluates the elements and applies the selection, crossover and mutation operators to evolve the population.

g) Try creating several instances of the problem, with different dimensions (varying the number of disciplines and slots) and difficulties (varying the students enrolled in the disciplines).

h) Solve the various instances of the problem using different parameterizations of the algorithm, namely by varying the population size, number of generations, selection method, crossover method, mutation method, etc.

i) Compare the developed program, based on Genetic Algorithms, with the Hill climbing local search strategies (in its various versions) and Simulated Annealing, in terms of the quality of the solution obtained and the time it takes to obtain the solution, for different problems.