Julian Gonzales (jgonzalez107@miners.utep.edu)
Payam Kelich (pkelich@miners.utep.edu)

Programming Assignment #2

In the recent assignment, we used two methods to solve the scheduling problem: BackTracking and the Genetic Algorithms. Here is a brief explanation of how we implemented these methods:

BackTracking

There are two methods used for the backtracking algorithm. The first one is the main one classed by **Main.java**. The first method is only used to create an empty solution Schedule to pass to the recursive method with the problem and size parameter of zero within the recursive method; the first thing that is done is to check the size of the problem and its courses to see if the recursive parameter size is equal to it and if it is we return solution, the empty Schedule. The main portion of the method contains a **for loop** that goes through a temporary value of problem time slot length. The first thing that is done is a check to see if the time slot value array at ***i*** is greater than -1 if not loop again. Inside the if statement, we loop through the length of the problem room's size. A single if statement checks if the solution schedule at ***i*** and ***j*** is less than zero. And if it is, we set the input parameter size to it and create another temporary Schedule that recursively calls the method. With **recursive backtracking**, we can reduce our problem down to a simpler and more easily manageable problem of the same kind, allowing us to improve from the naive algorithm. The recursive backtracking algorithm also affords us a more manageable code set to read and depends on the problem set to solve our calls quicker than other algorithms.

Genetic Algorithms

The genetic algorithm contains six methods. **SolveGenetic** method loops through an array of Schedule populations initialized to the size of 8. Inside the method, we are looping to find the fittest individual through 8 generations and store them. Afterward, we call **getFittest** while calling the **evolvePopulation** method. Evolvepopulaton creates populations randomly and uses these randomly generated populations for crossover. MutateSchedule helps an individual schedule to create random values and sets them to an individual schedule "gene". Crossover helps generate child schedules from the parent. **getFittest** takes the population array and problem and compares the two to see the fittest schedule to be used for the solution. The Genetic algorithm improvement is the ability to create better solutions than many algorithms. This is done by evolving our solutions/individuals over generations until we stopped at our set criteria, eight generations. The genetic algorithm also produced better scores than the recursive backtracking method and naive method through some preliminary testing that was observed, attributed to its evolution method. One other significant improvement was that there was always a solution to be had, and if we wanted to could return a list of multiple solutions to be used for our problem.

Running the code

*)java -jar pa2.jar a1 a2 a3 a4 a5 a6

The code is an argument based code which the order of the arguments is defined below:

a1= Number of Buildings
a2= Number of Rooms
a3= Number of Courses
a4= Time limit (s)
a5= Algorithm number
a6= Random seed
a5) if **1** it means the backtracking algorithm and **2** means the genetic algorithms

Note for genetic Algorithm: during the method runs in some steps, it will show the "ERROR: Invalid schedule" but at last, it will find the solution.

Examples:

```
Genetic Algorithms
java -jar pa2.jar 2 6 25 2 12
result:
Number of Buildings: 2
Number of Rooms: 6
Number of Courses: 25
Time limit (s): 10
Algorithm number: 2
Random seed: 12
Genetic Algorithms is selected
ERROR: invalid schedule dimensions
..................................
ERROR: invalid schedule dimensions

Deadline: 1618197231097
Current: 1618197221174
Time remaining: 9923
Score: 134066.52629500465
```

```
Backtracking Algorithm
java -jar pa2.jar 2 6 25 10 1 12
Number of Buildings: 2
Number of Rooms: 6
Number of Courses: 25
Time limit (s): 10
Algorithm number: 1
Random seed: 12
BackTacking Algorithms is selected
Deadline: 1618197342415
Current: 1618197332415
Time remaining: 10000

Score: 661.0535812629722
```