Andrew Madrid, Alan Caldelas

CS 4320 Artificial Intelligence

HW3: Search for Room Scheduling

**Individual Contributions:**

Andrew Madrid:

* Simulated Annealing

Alan Caldelas

* Backtracking

**Simulated Annealing Results:**

When comparing the results of simulated annealing to the native baseline strategy, it performs better. Starting off with a smaller set of arguments (1 building, 10 rooms, 10 courses), simulated annealing is able to score higher, 621.4497271460676, compared to the native baseline score of 205.39348050338003.

As we start moving into large set of arguments (100 buildings, 125 rooms, 250 courses), we start to notice a trend. The native baseline scores at 5104.679178464228, while simulated annealing scores at 6466.129053993989. While simulated annealing is able to continue to score higher, does take more time to calculate.

These tests were done with an initial temperature of 10000. The decision to keep the initial temperature at 10000 is to allow it to be possible to make a move on the current solution. When lowering the initial temperature to 1000, simulated annealing is still able to perform better than the baseline, averaging in at 6030.014687302584. It is ease to tell that having a higher initial temperature means that there will be a higher chance of receiving a better score. The initial temperature is changed in the Main.java file.

**Simulated Annealing Output Examples:**

Initial Temperature at 10000

Number of Buildings: 1

Number of Rooms: 10

Number of Courses: 10

Time limit (s): 100000

Algorithm number: 1

Random seed: 1

Deadline: 1554447718761

Current: 1554347718833

Time remaining: 99999928

Score: 594.4007418043191

Number of Rooms: 125

Number of Courses: 250

Time limit (s): 100000

Algorithm number: 1

Random seed: 1

Deadline: 1554447375566

Current: 1554347377534

Time remaining: 99998032

Score: 6572.6792875359915

Initial Temperature at 1000

Number of Buildings: 1

Number of Rooms: 10

Number of Courses: 10

Time limit (s): 100000

Algorithm number: 1

Random seed: 1

Deadline: 1554447743845

Current: 1554347743871

Time remaining: 99999974

Score: 566.527814229027

Number of Rooms: 125

Number of Courses: 250

Time limit (s): 100000

Algorithm number: 1

Random seed: 1

Deadline: 1554447182093

Current: 1554347182249

Time remaining: 99999844

Score: 5943.552604647003

Initial Temperature at 100

Number of Buildings: 1

Number of Rooms: 10

Number of Courses: 10

Time limit (s): 100000

Algorithm number: 1

Random seed: 1

Deadline: 1554447785319

Current: 1554347785323

Time remaining: 99999996

Score: 455.49208573832266

Number of Rooms: 125

Number of Courses: 250

Time limit (s): 100000

Algorithm number: 1

Random seed: 1

Deadline: 1554447352807

Current: 1554347352879

Time remaining: 99999928

Score: 5564.254231584619

Native Baseline:

Number of Buildings: 1

Number of Rooms: 10

Number of Courses: 10

Time limit (s): 100000

Algorithm number: 0

Random seed: 1

Deadline: 1554447694070

Current: 1554347694070

Time remaining: 100000000

Score: 205.39348050338003

Number of Buildings: 100

Number of Rooms: 125

Number of Courses: 250

Time limit (s): 100000

Algorithm number: 0

Random seed: 1

Deadline: 1554447533851

Current: 1554347533852

Time remaining: 99999999

Score: 5104.679178464228

**Backtracking Results:**

Backtracking algorithm that was originally created was significantly slower than the baseline, this algorithm was the Recursive method that does not use any heuristics to implement. This causes the method to become slow due to the method going from 0 to the size of the courses.

The Min backtracking method used the outcomes of the constraints that were given by the course sizes this was used for the minimum remaining values approach by selecting the smaller outcome of the bubble sorting and storing it. After this we recursively did the same thing as the previous backtracking method but now it outperforms the original backtracking method.

With the max backtracking method the same approach was used, sorting by a heuristic, the approach used was bubble sort but in this example we implemented a different swap with the max value and storing it so it can be used for our recursive call for the backtracking method. Although we noticed that some values that we inputted caused the score to lower and sometimes go up, some Figs. show this change, we were not sure why.

Results showed that our methods scored the same with different increasing sizes for rooms, courses, seed, temperature and buildings. The method scored much higher than the Fig 1-5 demonstrate the results with 3rd to last argument of the file being the algorithm method. 0 (baseline), 2 (backtracking), 3/4 (min/max backtracking).





