Algorithmic Methods for Mathematical Models

Lab Session 4 - Greedy, Local Search

2022-2023

NIKOLAOS BELLOS

ANASTASIOS PAPAZAFEIROPOULOS

---

a) Pseudocode for Greedy algorithm

```
''' =========
      GREEDY
''' =========

# 1. Get a task
# 2. Get a list of feasible assignments
# 3. Select the assignment with the lowest highest load

def greedy():
    solution = [0]*len(computers)
    sortedTasks = sort(Tasks, key=taskResources, desc)
    for task in sortedTasks:
        candidateComputers = findAvailableComputers(task)
        if not candidateComputers:
            break
        sortedComputers = sort(candidateComputers, key=computerLoad,
asc)
        solution[sortedComputers[0].id].append(task.taskResources)

    return solution
```

b) Pseudocode for local search

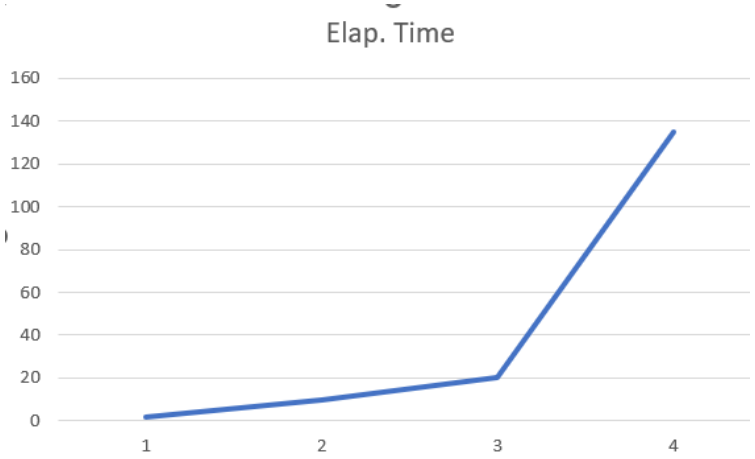
```
''' =======
      LOCAL SEARCH
''' =======

# 1. Get the initial solution
# 2. Get the fitness score of the solution
# 3. Get the neighbors of the solution
# 4. Get the fitness score of the neighbors
```

```
# 5. If there are no neighbors or time limit is exceeded, stop
iterating

# Generate pseudocode for the above instructions
def local_search():
    solution = initialSolution() # from greedy
    fitness = solution.getFitness()
    while True:
        neighbors = solution.getNeighbors()
        if not neighbors:
            break
        for neighbor in neighbors:
            neighbor_fitness = neighbor.getFitness()
            if neighbor_fitness < fitness:
                solution = neighbor
                fitness = neighbor_fitness
                break

    return solution
```

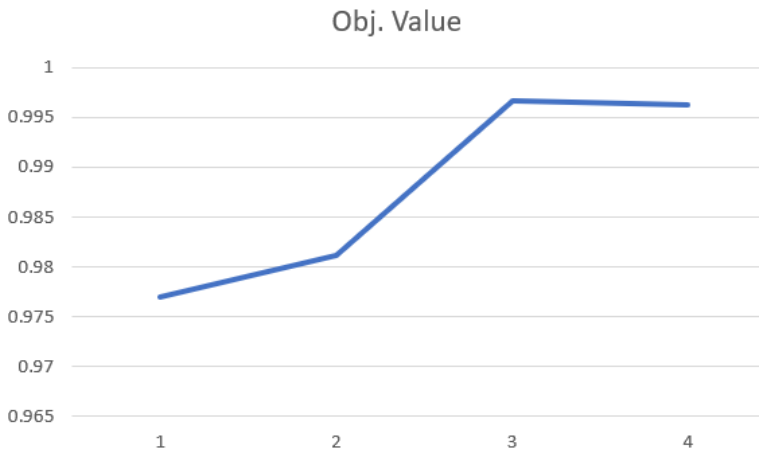
c) Generated data:
https://drive.google.com/drive/folders/1gQV30FW0YESfoNyycwvzLn8XEn-rsf_1?usp=share_link

Small: 10 CPUs, 30 Tasks
Small to Medium: 15 CPUs, 40 Tasks
Medium: 20 CPUs, 60 Tasks
Large: 40 CPUs, 100 Tasks

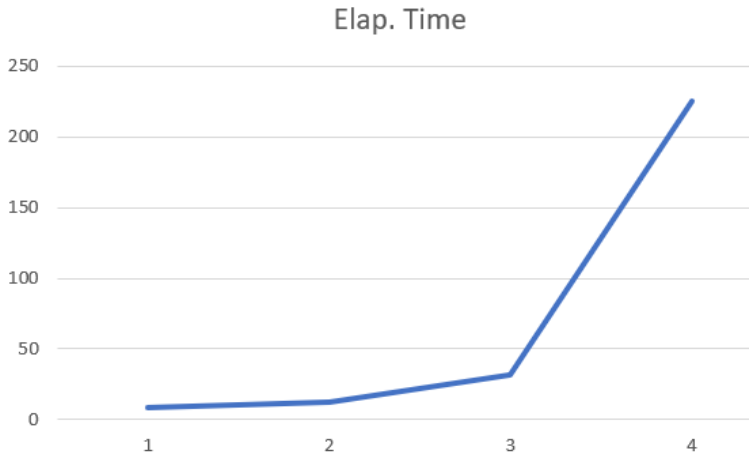
d) The following diagrams were produced:
Random: i) Time to solve/category (small = 1, ..., large = 4):



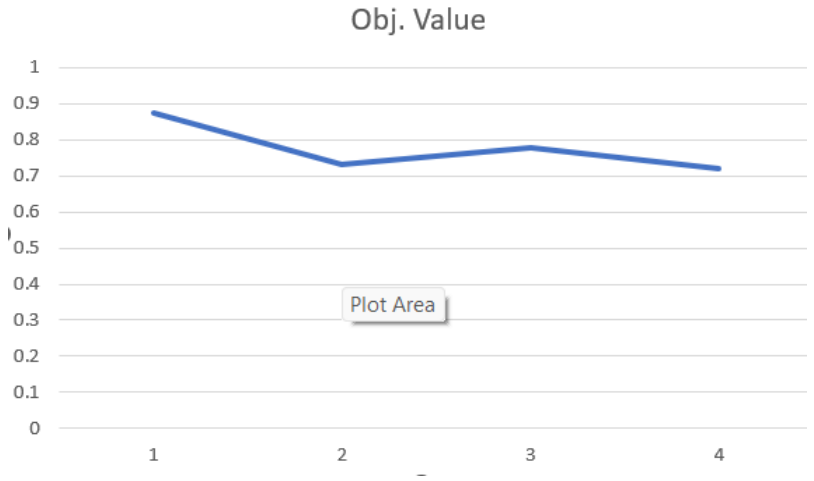
ii) Quality of the solutions/category (small = 1, … , large = 4):



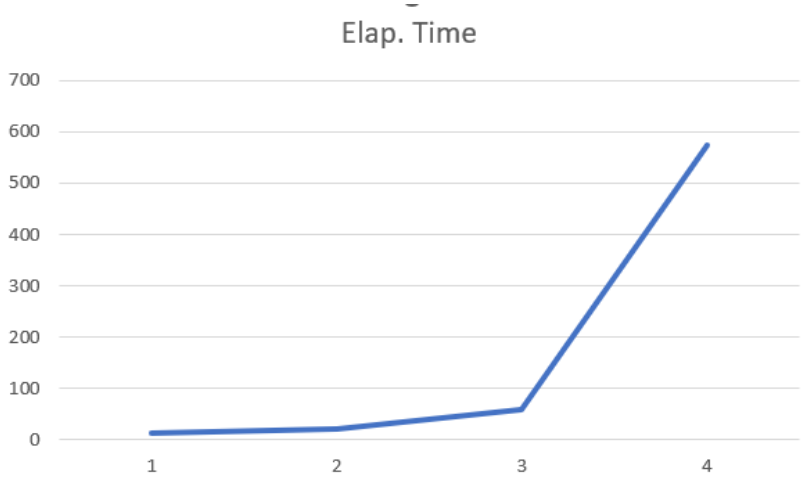
Greedy: i) Time to solve/category (small = 1, …, large = 4):



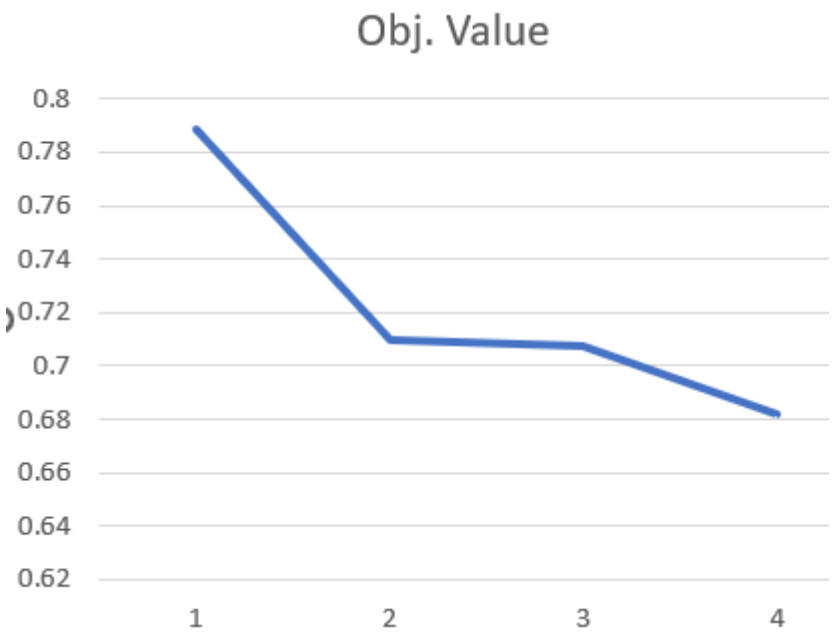
ii) Quality of the solutions/category (small = 1, … , large = 4):



Greedy + Local Search: i) Time to solve/category (small = 1, …, large = 4):



ii) Quality of the solutions/category (small = 1, … , large = 4):



As we notice, the objective value is the minimum in the Greedy + Local Search case and in the Greedy case is lower than the random one. In addition, the elapsed time is the lowest in the random case and the highest in the Greedy + Local Search case, as we expected.

e)
After one random example, we took the following results:

```
Total load 185084.596613385
Total capacity 200000
Computers have enough capacity
Max load 93.456666676%
CPU 1 loaded at 92.980232945%
CPU 2 loaded at 91.477796453%
CPU 3 loaded at 91.857466608%
CPU 4 loaded at 91.362154153%
CPU 5 loaded at 92.865617343%
CPU 6 loaded at 92.275132922%
CPU 7 loaded at 92.86104028%
CPU 8 loaded at 93.456666676%
CPU 9 loaded at 92.225042061%
CPU 10 loaded at 93.138297563%
CPU 11 loaded at 92.553335991%
CPU 12 loaded at 92.73587814%
CPU 13 loaded at 92.748536988%
CPU 14 loaded at 93.131487705%
CPU 15 loaded at 92.516340425%
CPU 16 loaded at 90.851655615%
CPU 17 loaded at 93.321861836%
CPU 18 loaded at 93.197762268%
CPU 19 loaded at 93.296378973%
CPU 20 loaded at 91.99328119%
```