Spring 2018 - Artificial Intelligence

Assignment I, A Problem Solving Agent for HR- UnInformed Search - UPDATED

by Okan Tunalı → okan.tunali@ozu.edu.tr

Deadline for problem formalization: **26 February, 2018 Tuesday at 23:55.**
Deadline for final submission (Code+Report): **07 March, 2018 Wednesday at 23:55.**

Case definition:

Suppose that you are designing an intelligent Human Resource agent, which will decide to accept or reject the candidates. Table 1 shows how many employees will be hired for each department.

	Department 1	Department 2	Department 3
Required Employees	3	1	2

Table 1: Number of required employee for each department

In Table 2 you may see a list of candidates who applied for specific jobs with their expected salaries. For example, the candidate with ID 2 will expect to get $30 if he/she is hired for Department 2. Note that, zero expectation denotes that the candidate has no willingness for those departments. For instance, the candidate 2 applied only for the Department 2.

The candidates were evaluated *one after another* and once a decision is made about a candidate; it could not be undone.

Please follow the rest knowing that the agent will be doing an offline search, so that the candidates actually applied in order as in Table 2. And this table is given to your agent as a map and the agent is expected to show which goal state will be achieved and how they are achieved (path) by the algorithms BFS, DFS and UCS.

- For example, C1 and C2 is hired, “but what would happen if we rejected C2” will be the neighbour branch.
- Candidates come one after another. Parent - child relations should not be broken.
- This will be an offline search.
- “Future is unknown” in the sense that when the agent visits a state (evaluate if it’s a goal state; and if not, it will explore the children nodes), it does not know children of its own children.
    - Example from Romania map: When you visit Arad and see that it’s not the goal state Bucharest; you explore Timi, Sibiu and Zerind but their children are not known yet.

The agent reaches one of one of the goal states when 3 candidates are hired for D-1, 1 candidate is hired for D-2, and 2 candidates are hired for D3 as denoted in Table 1. In addition, when if there is no any candidate left to apply, the search will not terminate. Intuitively, uniform cost search would reject any of the candidates they expect some salary and cost of rejection is zero.

Remark: You will be given a .csv file for expected daily salary of candidates table.

	Expected Daily Salary of Candidates		
CandidateID	Department1	Department2	Department3
1	50	0	0
2	0	**30**	0
3	85	0	0
4	0	0	55
5	0	0	45
6	0	0	30
7	80	0	0
8	0	55	0
9	30	0	0
10	35	0	0
11	0	70	0
12	0	90	0
13	0	0	20
14	90	0	0
15	0	55	0
16	0	0	35
17	0	35	0
18	0	0	45
19	40	0	0
20	0	80	0

Table 2: Sample dataset showing the salary expectation of each candidate in order

When the agent performs search, ensure that each state node in the search tree keeps the parent of the current node.

The goal of your agent is to hire the required candidates for the given positions. You are given two tasks:

- Prepare a formal representation of this problem for solving through uninformed search (e.g. how you can represent the states, actions, initial state, transition model, goal state). You should submit one-page report showing your problem formulation **until 26 February, 2018 Tuesday at 23:55.**

- Implement and run the following search strategies: breadth-first search, depth-first search and uniform-cost search. You should submit your code and assignment report showing the outputs **until 07 March, 2018 Wednesday at 23:55.**

Constraints:
Please make sure that you applied those constraints:

A. For each department
    a. you can hire the candidates for a given department if and only if his/her salary expectation is greater than zero.
    b. { Hired employees for Department_i } ⊂ { Candidates of Department_i }
    c. # of Hired employees for Department_i = # of Required employees for Department_i

Important Notes:

- Allowed languages: Java, Python
    - If coded in Java
        - Deliver: Importable Project Folder
- Be careful about dependencies. Since we will also run the code, you may assume that we do not have external libraries installed.
- LinkedLists, ArrayLists or just Lists (for Python) will probably be enough.
- Please do not include local path definitions like
    - C:\user\myAIDocument\…
    - /home/user/Documents/myAIDocument/…
- You may limit your candidates to first 10 candidates. As we will still have
    - 5 candidates to Department_1,
    - 2 candidates to Department_2,
    - 3 candidates to Department_3, we have many possible goal states that can be reached by our search methods.

- When your algorithm achieves a goal state, the search ends. Then you need to print out that path from initial state to goal state.
    - Path is easy to track when we define it as a vector. If you remember our sample representation:
        - [{},{},{},3,1,2]
        - [{C1},{},{},2,1,2]
        - …

- Order of the candidates will stay the same. We will not shuffle them.
- Applicants give you the information to which department they apply for

- Impoting source codes from web is for sure a kind of plagiarism.

- Final note about submission:
    - Please send your report in **.pdf** format.
    - Organize your scripts as follows:
        - AI_BFS.java or AI_BFS.py
        - AI_DFS.java or AI_DFS.py
        - AI_UCS.java or AI_UCS.py
        - YourAssignmentReport**.pdf**