Automated Nurse Rostering System

Overview

This exercise involves solving a rostering problem using constraint-satisfaction techniques. We consider a hospital that requires a timetable specifying the work shifts for nurses over a specified period of time. The process of generating a roster must ensure that rostering decisions comply with the requirements of the hospital as well as those of the nursing staff. We are asked to automate the roster preparation task for the hospital based on the requirements and preferences.

Part A: Rostering with Constraints

- Assume that there are (N) nurses enrolled at the hospital. The hospital prepares a roster for the nursing staff for (D) days at a time.
- Nurses at the hospital work in shifts. There are three shifts in a day, namely, the morning (M) shift, the afternoon (A) shift or the evening (E) shift. All nurses can do all shifts. A nurse can take rest on any day of the week and is considered as taking the rest (R) shift for that day. All nurses are assumed to be equally qualified and interchangeable for shifts.
- Further, the rostering for the hospital staff must conform to the following requirements:
    a. No nurse can cover more than one shift on a single day.
    b. A nurse should not be allocated to work in two consecutive morning shifts (M).
    c. A nurse cannot work in a morning shift (M) if he/she took the evening shift (E) the preceding day.
    d. The shift coverage requirements are that there should be exactly (m) nurses in the morning shift, (a) nurses in the afternoon shift and (e) nurses in the evening shift every day. Note that **(m + a + e) <= N.**
    e. The hospital must ensure that each nurse gets at least one rest day in a week.
- Please formulate a model of the above rostering problem as a CSP and implement an algorithm for generating rosters as per the above requirements. Think about how to model the requirement constraints and their impact on the run time of your algorithm.
- The input format is as follows. As an example, consider a rostering task with 5 nurses for a duration of 7 days with 2 nurses in morning shift, 1 nurse in afternoon shift and 1 in evening shift will be given as:

```
INPUT

N, D, m, a, e
```

5, 7, 2, 1, 1

- The output should provide an allocated shift for each nurse for the total number of days for rostering. The token “N-X” identifies the nurses where “X” denotes an index. The identified is followed by the shift allocated for each day. All the shifts must be specified for each nurse.

OUTPUT

N-0: A, M, A, M, A, M, R

N-1: M, A, M, A, M, R, M

N-2, M, R, M, R, M, A, M

N-3: E, E, R, M, R, M, A

N-4: R, M, E, E, E, E, E

- In case, there is no feasible solution for the problem, For example, if we reduce total nurses to 4 in the above example then indicate “NO-SOLUTION”.

OUTPUT

NO-SOLUTION

- The rostering solution is assessed in terms of quality (constraints satisfied) and the time taken to solve the problem.

Part B: Incorporating Preferences

- Next, assume that in the hospital, there are (S) senior nurses who have administrative responsibilities in addition to clinical duties.
- The senior nurses prefer the morning or the evening shifts over the afternoon shifts as they need to complete paperwork during that time. Their morning, afternoon, evening and rest preference constraints can be assigned weights as 2.0, 1.0, 2.0 and 1.0 respectively. All other constraints in the problem can be considered as of weight 1.0.
- Extend your algorithm to incorporate the above *soft preferences* in addition to the hard requirements specified in the previous part. Note that it is desirable that soft preferences are met but not meeting them does not make the schedule invalid.
- In this part, a rostering solution is assessed in terms of satisfying hard constraints and the degree to which the soft preferences can be met.
- The problem now involves finding the *maximal-weight* solution that respects the hard constraints. The presence of weighted constraints generalizes a constraint-graph to a model known as the *factor graph.* Explore ways to incorporate a notion of weights in your search process. The weight of an assignment is calculated as:

  Weight = $\prod(w_{Ni\_j})$ where i $\in$ {0,1,..... N-1}, j $\in$ {0,1,..... D-1}
- The input parameters for this part is assumed to additionally include:
  - The number of senior nurses (S) where (S <= N), where the senior nurses are assumed to be indexed from 0 to (S-1).
  - A time window parameter (T) for generating the solution within the specified time duration (in seconds).
- As an example, consider a rostering task with 5 nurses for a duration of 3 days with 2 nurses in morning shift, 1 nurse in afternoon shift and 1 in evening shift. We have 2 senior nurses so, N-0 and N-1 will be considered as senior nurses.

INPUT

N, D, m, a, e, S, T

5, 3, 2, 1, 1, 2, 120

OUTPUT

N-0: M, A, M
N-1: M, E, E
N-2: E, R, M
N-3: A, M, A
N-4: R, M, R

- In the example output, please note the following:
  - All hard constraints mentioned previously in Section 1 are satisfied.

- The soft constraints are incorporated as much as possible. N-0 and N-1 are senior nurses and so given preference in allotting morning or evening shifts as long as any hard constraint is not violated.
- The weight of this assignment = $\prod(W_{Ni\_j})$ where i $\in$ {0,1,..... N-1}, j $\in$ {0,1,..... D-1}
  Weight = (2 x 1 x 2) x (2 x 2 x 2) x (1 X 1 X 1) x (1 X 1 X 1) x (1 X 1 X 1) = 32

Implementation Guidelines

- Please use Python 3.6 for your implementation. Your code should use only standard python libraries. Please do not include any dependencies on third party libraries or existing algorithmic implementations.
- The input problem will be specified in a ***csv* file** with following format
    - Header for part A - {N,D,m a,e} and header for part B - {N,D,m,a,e,S,T}
    - Each line after this would correspond to a new test case.
    - A sample of *input_a.csv* for part A:

```
N,D,m,a,e
5,3,2,1,1
10,9,3,2,2
```

    - A sample of *input_b.csv* for part B:

```
N,D,m,a,e,S,T
5,3,2,1,1,2,180
6,2,2,2,2,3,120
```

- Please provide your code in a single file called **A2.py**. Your code should run with the following command:

```
python3 A2.py input_a.csv
```

- Dump your output as a **list of dictionaries**, each dictionary corresponding to a satisfiable assignment for each test case, to a json file named - **“solution.json”** in the same directory as your python script.

```
soln_list = [ {"N0_0": "R", "N1_0": "R", "N2_0": "A", "N0_1": "R", "N1_1":
"M", "N2_1": "E"} , {....} , ... ]

with open("solution.json" , 'w') as file:

   for d in soln_list:

       json.dump(d,file)

       file.write("\n")
```

- The dictionary for a satisfiable assignment must have keys - “N{i}_{j}” where “i” is the nurse id (indexing from 0) and “j” corresponds to j^{th} day's assignment (starting from 0th day) for all variables in the CSP and value should be a single character - ‘M’/ ‘A’/ ‘E’/ ‘R’
    - Ex: N=2, D=2 : dictionary = {‘N0_0’: ‘M’, ‘N0_1’: ‘E’, ‘N1_0’: ‘E’, ‘N1_1’: ‘M’}
    - In case of “No-Solution”: dictionary = { }
- A maximum cut-off time would be kept (for part A) to ensure that we can stop implementations that may not terminate.
- We foresee a typical running time of 10 minutes for a particular rostering problem and the ability to schedule a roster for up to <100 total nurses and <100 days.
- If the above guidelines need refinement in due course, students will be notified.

Submission Instructions

- This assignment is to be done individually or in pairs. The choice is entirely yours.
- The assignment is to be submitted on Moodle.
- Submit a single zip file named *<A2-EntryNumber1-EntryNumber2>.zip or <A2-EntryNumber>.zip*. Upon unzipping this should yield a single file named - **A2.py**. Exactly one of the team members needs to submit the zip file.
- Please submit an additional text (max. 2 pages in standard 11 point Arial font). Please describe the core ideas of your implementation.
- **The submission deadline is 5pm on October 25, 2021.**
- This assignment will carry 15% of the grade.
- Late submission deduction of (10% per day) will be awarded. Late submissions will be accepted till 2 days after the submission deadline. There are no buffer days. Please submit by the submission date. Please strictly adhere to the input/output syntax specified in the assignment. In case, the submission does not comply with the guidelines, it would lead to exclusion/penalty from the assessment.
- Queries (if any) should be raised on Piazza.
- Please only submit work from your own efforts. Do not look at or refer to code written by anyone else. You may discuss the problem, however the code implementation must be original. Discussion will not be grounds to justify software plagiarism. Please do not copy existing assignment solutions from the internet: your submission will be compared against them using plagiarism detection software. Copying and cheating will result in a penalty of at least -10 (absolute). The department and institute guidelines will apply. More severe penalties such as F-grade will follow.
