EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

MATH. - NATURWISS. FAKULTÄT
FACHBEREICH INFORMATIK
KOGNITIVE SYSTEME · PROF. A. ZELL

Artificial Intelligence

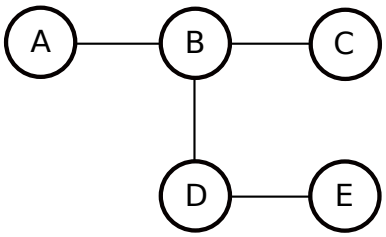
Assignment 5

Assignment due by: 03.12.2021

Question 1 Searching with/without observations (3+2+2 = 7 points)

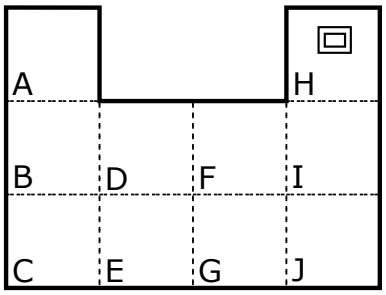
Given are agents in a known environment shown by the images. The agents know the movements: upwards (U), downwards (D), right (R) and left (L). There are no illegal moves, i.e. if the agent cannot move, it remains in the same state.

(a)



Consider a sensorless agent in the known environment depicted by the graph. Starting from an initial belief-state of total ignorance (the agent has no clue where it is), draw the belief-state graph that connects all possible belief-states that the agent accepts. It should include all four movements for each belief-state. Label the arrows with the corresponding movements.

(b)



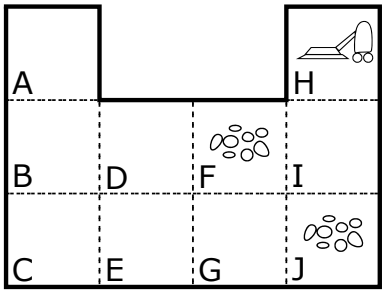
Consider a vacuum cleaner in the known environment where each possible discrete position (A ... J) fully describes a state. Unfortunately, the light has been turned off, and the vacuum cleaner doesn't know and see where it is. Fortunately, it can turn on the light by simply rolling over the switch at position 'H'. Specify a preferably short sequence of movements that the vacuum cleaner should follow to get from any starting point to the goal state 'H'. Draw the corresponding belief-state graph.

(c) Consider the same problem but with a vacuum cleaner which has a pressure sensor mounted in front of it, able to detect walls in front. The direction of the pressure sensor is always the continuation of its previous movement (i.e. the robot is directed downwards if it previously moved down). A state is fully described by the discrete position and the robots direction, e.g. ('A', d), with d: downwards. Specify a preferably short sequence of movements to reach any of the goal

states ('H', any direction). Draw the corresponding belief-state graph and update your belief-state with your observation (given by the pressure sensor) after each action.

Question 2 AND-OR Trees (3 points)

The vacuum cleaner could finally turn on the lights and is located at position 'H'. It turns out, that it is a very slippery vacuum cleaner and only by pure luck, it could reach the switch without any slip so far.



With its camera and turned on lights it can detect its own position and dirt on it. However its actions are non-deterministic and have the following error: Any movement may lead to a 90°deviation clockwise, i.e. up movement may lead to right movement instead, right may lead to down movement, etc. Starting from state ('H', clean) express a solution to clean the room with an AND-OR Tree which takes the non-deterministic actions of your vacuum cleaner into account. Include multiple actions as OR-nodes if it is reasonable to do so. End your tree as soon as 'F' and 'J' are clean (do not check the rest of the room for dirt). Each leaf node is a state in which the room is completely clean.

Question 3 Genetic Algorithms with Python (1+2+2+2+1+2 = 10 points)

We want to solve the $n\ queens\ problem$ using a genetic algorithm. The task is to place n non-attacking queens on an $n \times n$ chessboard (similar to the eight queens problem as shown in the lecture slides). A state can be represented as n digits, each in the range of 1 to n. Each digit represents one column of the board and the value itself represents the row the queen occupies in that column. Use the template file `genetic.py` for this task.

(a) Implement a function `initializePopulation(population_size, num_queens)`, which takes the population size (number of individuals in a population) and the size of the board (which also is the number of queens), and returns a randomly generated population.

(b) Implement a function `findConflicts(individual)`, which returns the number of attacking pairs of queens (or $\#\text{conflicts}$). For example the state $[1, 1, 1, 1, 1, 1, 1, 1]$ or $[1, 2, 3, 4, 5, 6, 7, 8]$ has 28 $\#\text{conflicts}$ and $[2, 4, 7, 4, 8, 5, 5, 2]$ has only 4.

(c) Implement a function `randomSelect(population, λ)`, which chooses a parent from the population based on the fitness value. You should use the fitness function $f = e^{-\lambda x}$ (where x is the $\#\text{conflicts}$ and λ is a decaying factor) and roulette-wheel selection for parents (probability of being chosen as a parent $p_i = \frac{f_i}{\sum_j f_j}$).

(d) Implement a function `crossover(parents)`, which takes a list of parents and reproduces an offspring by combining parts of each parent at randomly chosen crossover points.

(e) Implement a function `mutate(individual)`, which assigns a random position between 1 and n to each queen with the probability of $1/n$.

(f) Using the functions in previous parts, implement the complete genetic algorithm. The function should return its solution (an individual with zero conflicts) and the amount of individuals (population size * number of generations) it took to find it. Test your algorithm on the 4, 8 and 12 queens problem with varying decay factor λ and number of parents per child. You may limit the number of parents to a maximum of $n/2$. Vary these parameters over experiments but keep them fixed during one experiment. Report your findings.