Artificial Intelligence II - Homework 1
Missionaries and cannibals problem

Jonáš Petrovský
Faculty of Business and Economics,
Mendelu University in Brno
Czech Republic
jond@post.cz

October 24, 2015

1 Introduction

The goal of the work is to implement an own solution to the "Missionaries and cannibals problem". You can see the problem description here: `https://en.wikipedia.org/wiki/Missionaries_and_cannibals_problem#The_problem`

2 Solution design

The problem can by solved by creating and searching a state space (tree). A single state has the following form: ((a1,b1,c1), (a2,b2,c2)), where a1, b1, c1 are numbers of missionaries, cannibals and boats on the left bank, and a2, b2, c2 is the same for the right bank.

We need a starting and a target state, operators (actions) and a criterion for finding the cheapest (shortest) path.

We can use Breadth-first search (BFS) or Depth-first search (DFS). BFS finds the optimal solution among all states in the state tree, while DFS finds the first available solution (which doesn't have to be optimal).

3 Implementation

The program was implemented in Python. There is a main class `MissionaryWorld`, which in its constructor accepts number of cannibals (same as missionaries) for choosing whether to solve 2+2, 3+3 or 4+4 problem.

The state tree nodes are saved in `state_tree['nodes']` variable under given ID and have a following format (example for root node):

```
root_node = {'left': self.start_state[0], 'right': self.start_state[1]
 'id': 0,  'parent_id': None, 'level': 0, 'used_op': None}
```

The main method `generate_tree` accepts a search type (DFS or BFS) as its parameter and finds the desired solution.

1. Create a data structure for nodes (queue for BFS and stack for DFS)
2. Insert root node (starting state)
3. While the data structure is not empty:
   (a) Generate new possible states from the examined node: `_generate_tree_leve`
   (b) Add new nodes to the tree. If the new node is the targe state, end the program and show the solution.
   (c) Only for DFS – check if the level is not too deep. If it is, do not add more children.
   (d) Add new nodes to the data structure.

A method `_check_new_state(current_state, operator)` is called inside the `_generate_tree_level(current_state)` method and checks for all operators if the generated state is possible – does not violate any conditions: enough people for transport, no more cannibals than missionaries on both banks and that the new state is not the same as the previous state of the current state.

Finally a method `_show_solution(target_node)` reconstructs the path and counts total number of steps. Format description:
`<level n.> (<node ID>): <state left>, <state right> <- <used operator>`

4 Results

The program can be run by typing the following commond into the operating system terminal (Python 2.7 has to be installed on the system):
`python run.py <number of cannibals> <search type>`
The first parameter can be 2, 3 or 4 and the second BFS or DFS.

4.1 3+3

The original problem of 3 missionaries and 3 cannibals can be solved with no problems. Both BFS and DFS found the optimal solution.

- path price (number of river crossings): 11
- BFS – target state was the 55. examined state.
- DFS – target state was the 15. examined state.

For this problem type is better DFS because of lower number of examined states needed.

4.2 2+2

The edited problem was also solved easily.

- path price (number of river crossings): 5
- BFS – target state was the 20. examined state.
- DFS – target state was the 15. examined state.

For this problem type is better DFS because of lower number of examined states needed.