CP 468 Assignment 1
missionaries and cannibals

Morouney, Robert
`robert@morouney.com, 069001422`

October 06 2016

1

River Crossing

1.1 The Problem

There are N Cannibals and M missionaries on one side of a river. All of the Cannibals and missionaries must cross the river using a single boat. The boat available is able to hold a maximum of 2 passengers and must at least have a single passenger to be the captain on each trip. As an added difficulty the missionaries cannot be outnumbered by cannibals on either shore.

States The problem can be represented using 2 vectors to represent either side of the shore. $[\vec{left}, \vec{right}]$ such that $\vec{left}, \vec{right} \in \mathbb{R}^3$ We can then use each vector to represent the number of cannibals missionaries and boats on each shore. Therefore $\vec{left}|\vec{right} =< m, c, b >$ Where $m, c, b \in N$ and $m \geq c$

Initial State The initial state for the problem is $[< M, N, 1 >, < 0, 0, 0 >]$ which represents all missionaries and cannibals on one side with a single boat.

Goal State The goal is to get all missionaries and cannibals to the opposite side of the river. Therefore for the above starting state the final state would be $[< 0, 0, 0 >, < M, N, 1 >]$

Actions In each state there can be a movement of: 1 cannibal and 0 missionaries | 2 cannibals and 0 missionaries | 1 cannibal and 1 missionary | 0 cannibals and 2 missionaries | 0 cannibals and 1 missionary. Each movement will add to one vector and subtract an equal value from the other, This is given that the aforementioned movement results in a valid state.

Heuristic Each state transition uses 1 movement which is counted in the function $g(x)$. Each state is then given a heuristic value with the function $h(x)$ such that $h(x) = SUM(\vec{(left)} - 1)$ This represents a diminishing values as the goal state approaches.

Transition Model As mentioned above each movement will be represented by a vector which will be applied inversely to each vector in the state model. For instance if the state is $[<1,1,1><0,0,0>]$ then a movement that causes the goal state would be represented as $<-1,-1,-1>$ and will be added to $\vec{left}$ and subtracted from $\vec{right}$.

Goal test Check if goal state is reached

Path Cost See heuristics above.

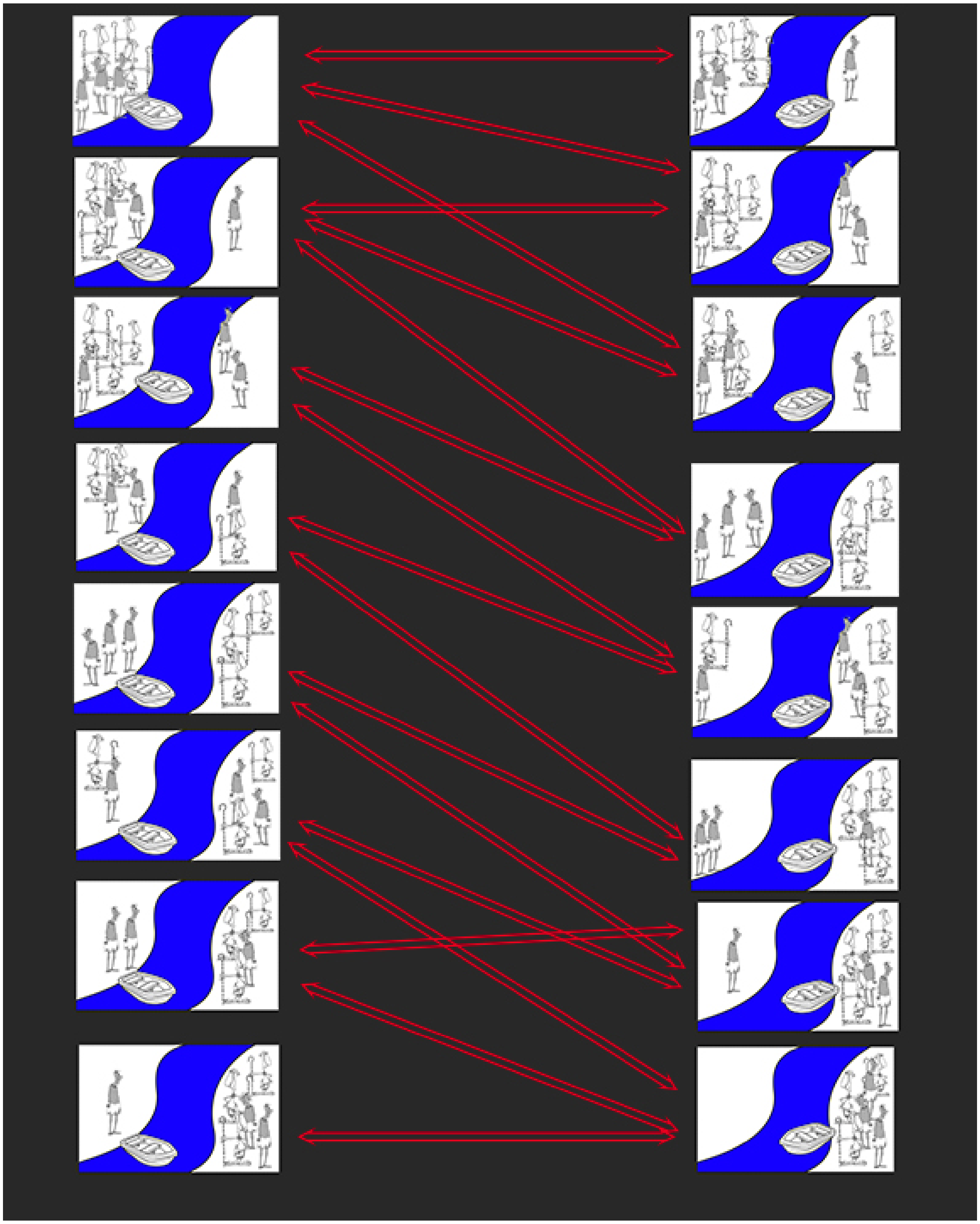
1.2 Solution

The problem states that an optimal solution must be found for an initial state with 3 cannibals and 3 missionaries and 1 boat. There are 4 optimal solutions for this initial state and there movements are listed below. Each transition model is considered optimal because the path cost for each is the same at 11 steps.

```
==================================================
SOLUTION #1   MOVES = 11


RIVER ~>       RIGHT SIDE    |   LEFT SIDE
State # 1 -> [[3, 3, 1], [0, 0, 0]]
State # 2 -> [[2, 2, 0], [1, 1, 1]]
State # 3 -> [[3, 2, 1], [0, 1, 0]]
State # 4 -> [[3, 0, 0], [0, 3, 1]]
State # 5 -> [[3, 1, 1], [0, 2, 0]]
State # 6 -> [[1, 1, 0], [2, 2, 1]]
State # 7 -> [[2, 2, 1], [1, 1, 0]]
State # 8 -> [[0, 2, 0], [3, 1, 1]]
State # 9 -> [[0, 3, 1], [3, 0, 0]]
State #10 -> [[0, 1, 0], [3, 2, 1]]
State #11 -> [[1, 1, 1], [2, 2, 0]]
State #12 -> [[0, 0, 0], [3, 3, 1]]   GOAL

==================================================
==================================================
SOLUTION #2   MOVES = 11


RIVER ~>       RIGHT SIDE    |   LEFT SIDE
State # 1 -> [[3, 3, 1], [0, 0, 0]]
State # 2 -> [[2, 2, 0], [1, 1, 1]]
State # 3 -> [[3, 2, 1], [0, 1, 0]]
State # 4 -> [[3, 0, 0], [0, 3, 1]]
State # 5 -> [[3, 1, 1], [0, 2, 0]]
State # 6 -> [[1, 1, 0], [2, 2, 1]]
State # 7 -> [[2, 2, 1], [1, 1, 0]]
State # 8 -> [[0, 2, 0], [3, 1, 1]]
State # 9 -> [[0, 3, 1], [3, 0, 0]]
State #10 -> [[0, 1, 0], [3, 2, 1]]
State #11 -> [[0, 2, 1], [3, 1, 0]]
State #12 -> [[0, 0, 0], [3, 3, 1]]   GOAL

==================================================
==================================================
```

```
SOLUTION #3  MOVES = 11


RIVER ~>      RIGHT SIDE   |  LEFT SIDE
 State # 1 -> [[3, 3, 1], [0, 0, 0]]
 State # 2 -> [[3, 1, 0], [0, 2, 1]]
 State # 3 -> [[3, 2, 1], [0, 1, 0]]
 State # 4 -> [[3, 0, 0], [0, 3, 1]]
 State # 5 -> [[3, 1, 1], [0, 2, 0]]
 State # 6 -> [[1, 1, 0], [2, 2, 1]]
 State # 7 -> [[2, 2, 1], [1, 1, 0]]
 State # 8 -> [[0, 2, 0], [3, 1, 1]]
 State # 9 -> [[0, 3, 1], [3, 0, 0]]
 State #10 -> [[0, 1, 0], [3, 2, 1]]
 State #11 -> [[1, 1, 1], [2, 2, 0]]
 State #12 -> [[0, 0, 0], [3, 3, 1]]  GOAL

==================================================
==================================================
SOLUTION #4  MOVES = 11


RIVER ~>      RIGHT SIDE   |  LEFT SIDE
 State # 1 -> [[3, 3, 1], [0, 0, 0]]
 State # 2 -> [[3, 1, 0], [0, 2, 1]]
 State # 3 -> [[3, 2, 1], [0, 1, 0]]
 State # 4 -> [[3, 0, 0], [0, 3, 1]]
 State # 5 -> [[3, 1, 1], [0, 2, 0]]
 State # 6 -> [[1, 1, 0], [2, 2, 1]]
 State # 7 -> [[2, 2, 1], [1, 1, 0]]
 State # 8 -> [[0, 2, 0], [3, 1, 1]]
 State # 9 -> [[0, 3, 1], [3, 0, 0]]
 State #10 -> [[0, 1, 0], [3, 2, 1]]
 State #11 -> [[0, 2, 1], [3, 1, 0]]
 State #12 -> [[0, 0, 0], [3, 3, 1]]  GOAL

==================================================
```

Listing 1.1: Missionaries and Cannibals



```python
#!/usr/bin/python3
from copy import deepcopy
import sys


if (len(sys.argv) != 3 ):
    print("ERROR IN", str(sys.argv))
    print("Wrong number of arguments! Should be run with ./Main.py M C")
    exit(0)
else:
    TOTAL_M = int(sys.argv[1])
    TOTAL_C = int(sys.argv[2])

LEFT_START = [TOTAL_M, TOTAL_C, 1]
START_STATE = [LEFT_START,[0,0,0]]
GOAL_STATE = [START_STATE[RIGHT], START_STATE[LEFT]]
M = 0
C = 1
B = 2
SOL = 0

class Node(object):
    def __init__(self, parent, depth, state):
        self.depth = depth
        self.parent = parent
        self.heuristic = depth + state[0][M] + state[0][C]
        self.state = state
        self.children = []

def generate_moves(state):
    RIGHT = state[1][B]
    LEFT = RIGHT-1
    for m, c in [(1, 0), (1, 1), (0, 1), (2, 0), (0, 2)]:
        _state = deepcopy(state)
        _state[RIGHT][M] -= m
        _state[RIGHT][C] -= c
        _state[RIGHT][B] = 0
        _state[LEFT][M] += m
        _state[LEFT][C] += c
        _state[LEFT][B] = 1
        for side in _state:
            legal = not any(((side[M] and
                              side[C] > side[M]),
                             side[M] // 4,
                             side[C] // 4))
            if not legal:
                break
        if legal:
            yield _state


def generate_nodes(queue):
    node = queue.pop()
    moves = generate_moves(node.state)
    for move in moves:
        unique = True
        tmp_node = node.parent
        while tmp_node:
```

```
            if tmp_node.state == move:
                unique = False
                break
            tmp_node = tmp_node.parent
        if unique:
            child = Node(node, node.depth+1, move)
            node.children.append(child)
            if move == GOAL_STATE:
                global SOL
                SOL += 1
                print("==================================================")
                print("SOLUTION #{}  MOVES = {}  \n\n".format(SOL,node.depth+1))
                print_node(child)
                print("\n==================================================")

            else:
                queue.insert(0, child)

def print_node(node):
    states = [node.state]
    tmp_node = node.parent
    while tmp_node:
        states.append(tmp_node.state)
        tmp_node = tmp_node.parent
    states.reverse()
    print("RIVER ~>     RIGHT SIDE   |   LEFT SIDE")
    for i, state in enumerate(states):
        print ("State #{0:2} -> {1}".format(i+1,str(state)))

if __name__ == '__main__':
    root = Node(0, None, START_STATE)
    queue = [root]
    while queue:
        generate_nodes(queue)
```

Listing 1.2: Missionaries and Cannibals