CP468 Artificial Intelligence
Asignment 1

Group 7

September 2018

Problem: Move all 3 Missionaries and 3 Cannibals to the other side of the river without letting Cannibals outnumber the Missionaries at any point of time.
Initial State: 3 Missionaries and 3 Cannibals with the boat on the left side of the river.
Actions: Move 1 Missionary to the other side of the river, Move 2 Missionaries to the other side of the river, Move 1 Cannibal to other side of the river, Move 2 Cannibals to other side of the river, Move 1 Missionary and 1 Cannibal to other side of the river.
Transition model: Moving 1 or 2 people from 1 side of the river to the other will change their location and the side that the boat is on.
Goal test: Everyone is transported to the other side of the river.
Path cost: 1 cost per each boat trip across the river.
Optimal Solution: Tuple describing the state follows following model:

[number of missionaries on left side, number of cannibals on left side, number of missionaries on right side, number of cannibals on right side, 0 if boat is on the left side]

11 steps:
From [33000] to [22111]
From [22111] to [32010]
From [32010] to [30031]
From [30031] to [31020]
From [31020] to [11221]
From [11221] to [22110]
From [22110] to [02311]
From [02311] to [03301]
From [03301] to [01321]
From [01321] to [11220]
From [11220] to [00331]

Code:

```
def search():
    root =[3,3,0,0,0,0]
    steps=0
    capacity= (1,2)
    found_sol = False
    visited = []
    queue = []
    queue.append(root)
    while(len(queue)!=0 and not found_sol):          # main loop
        current = queue.pop(0)
        hash = hash_func(current)
        if hash in visited:      #checking if visited
            continue
        else:
            visited.append(hash)#add to visited set
```

```
            if current[3]==3 and current[2]==3: #reached the end
                found_sol=True
            else:
                find_moves(current, capacity, queue)

    if found_sol:
        while(current!=root):
            steps+=1
            current=current[5]

        print("Found solution! Needed {0:d} steps to reach solution."
            .format(steps) )
    else:
        print("No solution found")

def hash_func(vertex):      #hashing vertex to check if visited
    left_m = vertex[0] << 7
    left_c = vertex[1] << 5
    right_m = vertex[2] << 3
    right_c = vertex[3] << 1
    if vertex[4]==0:
        return left_m + left_c + right_m + right_c + 1
    return left_m + left_c + right_m + right_c

def find_moves(vertex,capacity,queue): #add possible moves that don't
    #lead to game over
    count=0
    left_m = vertex[0]
    left_c = vertex[1]
    right_m = vertex[2]
    right_c = vertex[3]
    boat_on_left = vertex[4]==0
    if boat_on_left:
        #can take one or two people right?
        for i in capacity:
            if left_m>=i:
                v = [left_m-i,left_c,right_m+i,right_c,1,vertex]
                if not game_over(v):
                    queue.append(v)
            if left_c>=i:
                v= [left_m,left_c-i,right_m,right_c+i,1,vertex]
                if not game_over(v):
                    queue.append(v)
        if left_c>0 and left_m>0:
```

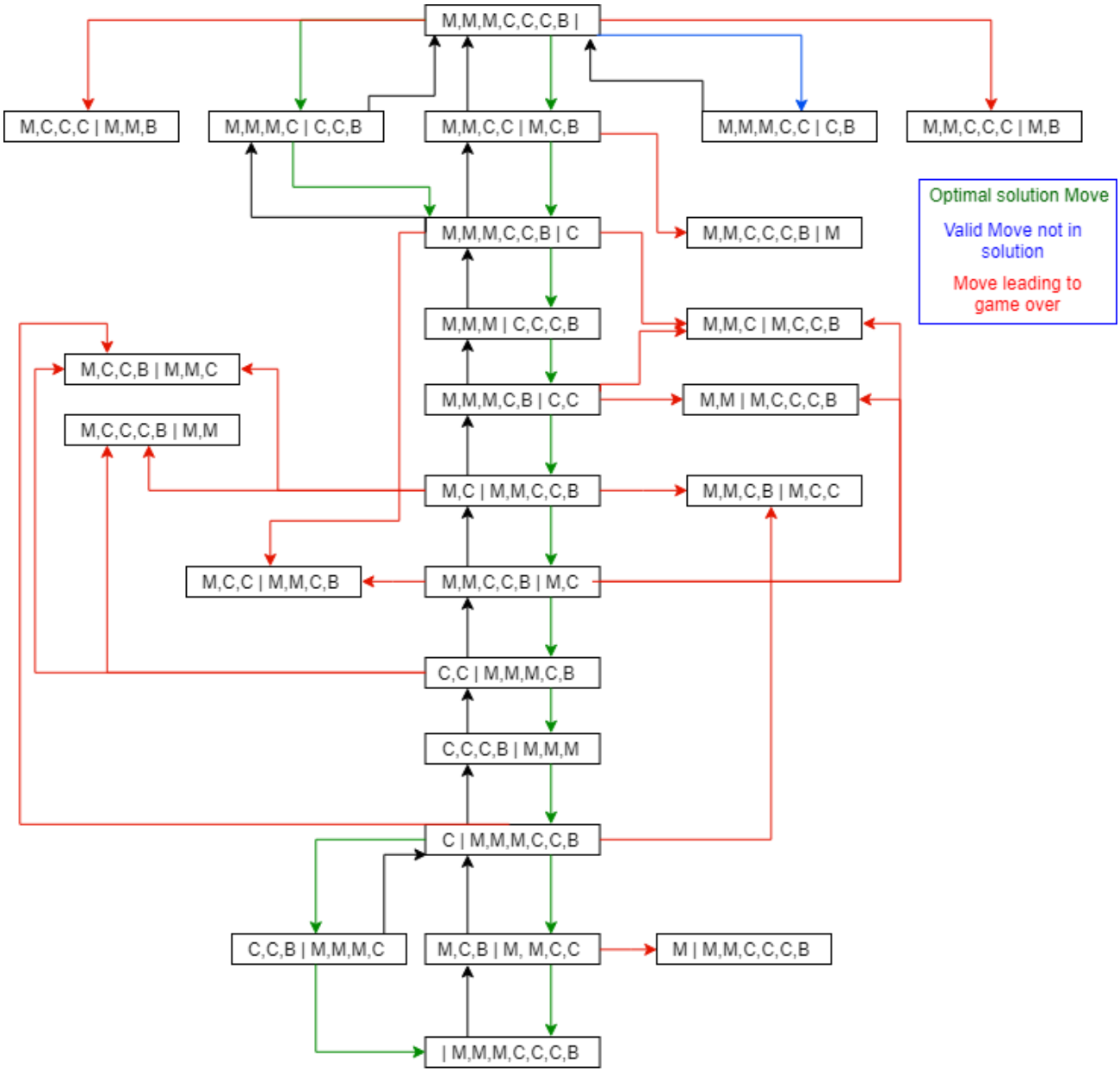
```
            v = [left_m-1,left_c-1,right_m+1,right_c+1,1,vertex]
            if not game_over(v):
                queue.append(v)

    else:
        # can take one or more people left?
        for i in capacity:
            if right_m>=i:
                v = [left_m+i,left_c,right_m-i,right_c,0,vertex]
                if not game_over(v):
                    queue.append(v)
            if right_c>=i:
                v= [left_m,left_c+i,right_m,right_c-i,0,vertex]
                if not game_over(v):
                    queue.append(v)
        if right_c>0 and right_m>0:
            v = [left_m+1,left_c+1,right_m-1,right_c-1,0,vertex]
            if not game_over(v):
                queue.append(v)

def game_over(vertex):

    return  ((vertex[1] > vertex[0]) and vertex[0]!=0) or ((vertex[3]
        > vertex[2]) and vertex[2]!=0)


search()
```


M,M,M,C,C,C,B |
M,C,C,C | M,M,B
M,M,M,C | C,C,B
M,M,C,C | M,C,B
M,M,M,C,C | C,B
M,M,C,C,C | M,B
Optimal solution Move
Valid Move not in solution
Move leading to game over
M,M,M,C,C,B | C
M,M,C,C,C,B | M
M,M,M | C,C,C,B
M,M,C | M,C,C,B
M,C,C,B | M,M,C
M,M,M,C,B | C,C
M,M | M,C,C,C,B
M,C,C,C,B | M,M
M,C | M,M,C,C,B
M,M,C,B | M,C,C
M,C,C | M,M,C,B
M,M,C,C,B | M,C
C,C | M,M,M,C,B
C,C,C,B | M,M,M
C | M,M,M,C,C,B
C,C,B | M,M,M,C
M,C,B | M, M,C,C
M | M,M,C,C,C,B
| M,M,M,C,C,C,B