Jennifer Shi j77shi@uwaterloo.ca
CS486 AI 20662230

A1 AI

a) State definition:
Let M be missionaries, C be cannibals, and B be the boat. We can represent a state by two sets. For example: the initial state will have $Side_d$ = {M,M,M,C,C,C,B} and $Side_a$ = {}; $Side_d$ = departure side, $Side_a$ = arrival side. And our goal state is $Side_d$ = { } and $Side_a$ = { M,M,M,C,C,C,B }

Successor function:
A state's successor is valid if and only if it satisfies the following:
- The boat is on $Side_d$ and is moving to $Side_a$ or vise versa
- The boat contains 1 or 2 people that are on the same side of the boat
- On the side which the boat leaves, the number of missionaries of the side after moving is either 0 or it is greater than or equal to the number of cannibals
- On the side which the boat arrives, the number of missionaries of the side after moving is either 0 or it is greater than or equal to the number of cannibals

b) Sydney's formulation marks each M and C uniquely. This will greatly increase the number of possible states needed and affects the performance of the search algorithm. For example, the initial state is 111111d, and it will have111101a and 1111011a as its successors. In the first case, C2 is being moved to the arrival side whereas in the second case, C1 is being moved to the arrival side. This is not good for performance since they are essentially the same state, that is, 3 missionaries and 2 cannibals on the departure side, and 1 cannibal on the arrival side. There is no need in comparing these two cases separately and as new states.

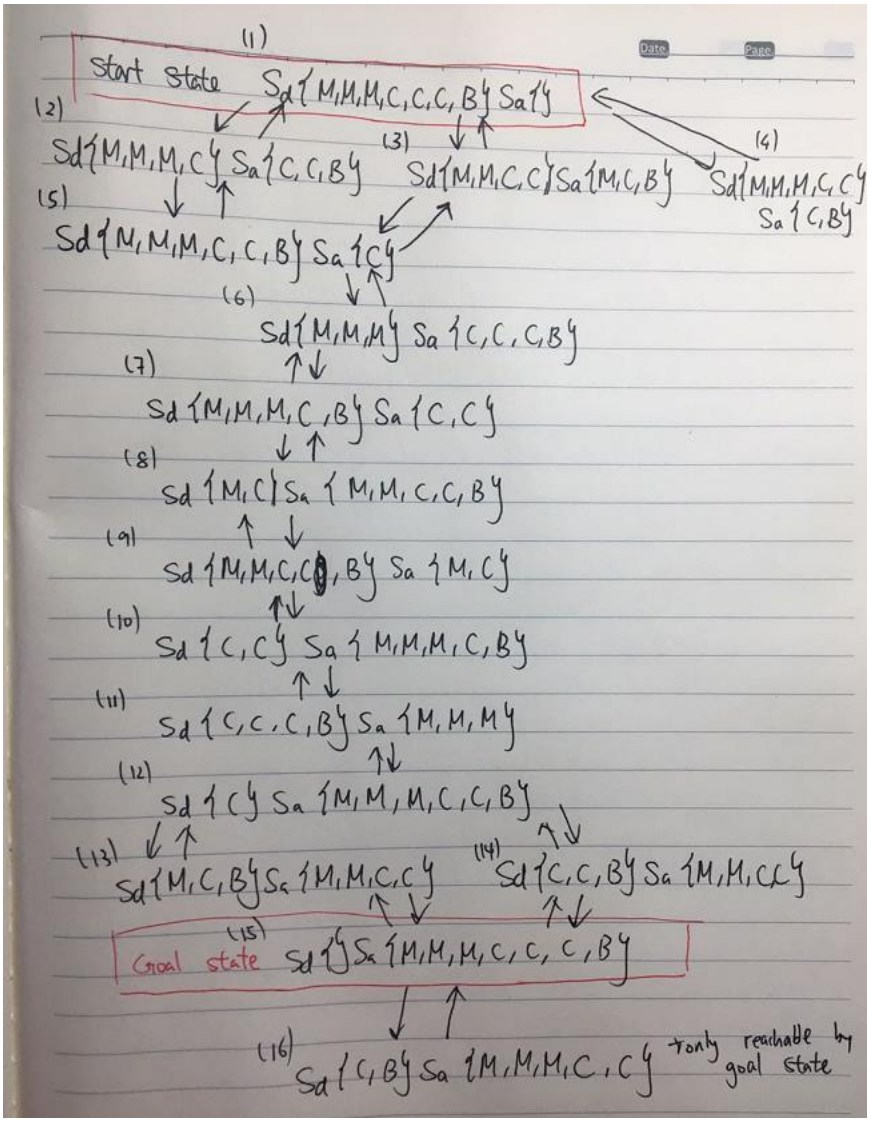
c) I think putting the constraints in the state definition is better (Sydney). Because a state exists iff it satisfies the state definition and the successor function determines which states can be reached for a given state. The state definition determines the set of states and invalid states should not be in the search graph whereas if you put the constraints in the successor function, then there may be states (by the incomplete state definition) are never reachable.

d) State definition:
Let M be missionaries, C be cannibals, and B be the boat. We can represent a state by two sets, let $Side_d$ be the set of the people in the departure side, and let $Side_a$ be the set of the people on the arrival side. A state is valid only if number of missionaries is either 0 or greater than of equal to the number of cannibals on both sides.
Initial state: $Side_d$ = {M, M, M, C, C, C, B} and $Side_a$ = {};
Goal state: $Side_d$ = {} and $Side_a$ = {M, M, M, C, C, C, B }
Invalid state: $Side_d$ = {M, C, C} and $Side_a$ = {M, M, C, B}

Successor function:
A state's successor is valid if and only if it satisfies the following:
- The boat is on $Side_d$ and is moving to $Side_a$ or vise versa
- The boat contains 1 or 2 people that are on the same side of the boat
- On the side which the boat leaves, the number of missionaries of the side after moving is either 0 or it is greater than or equal to the number of cannibals
- On the side which the boat arrives, the number of missionaries of the side after moving is either 0 or it is greater than or equal to the number of cannibals

Jennifer Shi
CS486 AI
j77shi@uwaterloo.ca
20662230

e)



There are total number of 16 states reachable labelled here in the search graph. Each arc indicates the possible successor of the state.
Initial state is state (1) S_d = {M,M,M,C,C,C,B} and S_a = {}; goal state is state (15) S_d = { } and S_a = { M,M,M,C,C,C,B }.

f) I would use depth first search with cycle pruning to perform the search. Since we can see from the search graph, the goal is reachable almost in a straight line of states, thus, we need to dig deep into the search graph to avoid expanding more states. Also, there are many possible loops, thus cycle pruning is needed to reduce the calculations.

g) A possible heuristic function is by calculating the number of moves needed to move all the people from the departure side to the arrival side without considering them eating by the cannibals. That is, the constraint where the number of missionaries must be greater than or equal to the cannibals is reduced.
i.e. if the state is $Side_d$ = {C, B} and $Side_a$ = { M,M,M,C,C}, then the heuristic value is 1, since only one move is required to move all the people from $Side_d$ to $Side_a$.
if the state is $Side_d$ = {C} and $Side_a$ = { M,M,M,C,C,B}, then the heuristic value is 2, since we need to move the boat to $Side_d$ first and then carry C back to $Side_a$.

Let p be the number of people on the departure side.

1) If boat is on the departure side, define the heuristic function to be

$$h(p) = (p-2)*2 + 1$$

because each time when you move 1 person from the departure side to the arrival side it will take 2 trips, since on the first trip we can bring 2 people, and the returning trip 1 person must come back with the boat. Except for the last trip, where the boat doesn't need to return, the p-2. And +1 at the end accounts for the last departing trip.

2) If the boat is on the arrival side, define the heuristic function to be

$$h(p) = p*2$$

Again, each time when you move 1 person, it requires 2 trips, and since the boat is on the arrival side right now, it needs to go back first to pick up new people, thus no need for p-2.

The initial state will have a heuristic of 9. The goal state will have a heuristic of 0.

This heuristic function is admissible because we solved a relaxed problem where we reduced the constraint of cannibals wouldn't eat the missionaries. Thus, we are just counting the number of moves needed to move the people from one side to the other while the boat must contain either 1 or 2 persons. And we solved the relaxed problem optimally since we are moving the max number (i.e. 2) of people in each trip, and 1 in returning the boat. There cannot be less trips taken to accomplish this task.

h) I think A* search algorithm may not be a better choice than the uninformed search algorithm. The search graph is pretty straight forward with only 16 states, we can reach the goal state without knowing how far we are from the goal state, just dig deep and without returning to the states that we visited. A DFS with cycle pruning can achieve this quite well.

Jennifer Shi j77shi@uwaterloo.ca
CS486 AI 20662230

2 The Hua Rong Dao Sliding Puzzle

a) I would store a state by using a 2D array of integers representing the graph. For example, the initial state of puzzle1 will be stored as [[2,1,1,3][2,1,1,3][4,6,6,5][4,7,7,5][7,0,0,7]]. I will keep track of the cost of the state and its parent. So it is easier to find the path when reaching the goal state. I will also keep track of where the zeroes (spaces) are in each state, for easier calculation of the successors.

I would implement the successor by finding the possible moves with each space. We need to consider cases of 2*2 piece, 1*2 pieces, and 1*1 piece, and where they are in relation to the two spaces available. We will first consider the first space, if the space surrounding it (up, down, right, left), is a 1*1 piece, we can simply just add it to the successor, if it is a 2*2 piece then we need to consider where the second space is, and if it possible for the 2*2 piece to move, if it is a 1*2 piece, we need to consider its orientation (vertical / horizontal), if it is horizontal, then we can move it right or left without considering the second space, if it is vertical, then we can move it up or down without consider the second space. Else, we need to consider the second space to see if the piece can be moved.

b) This heuristic function is admissible. It is solving a relaxed problem where the 2*2 piece can move freely without considering the blocking pieces.

e) My results for A* and DFS

	Puzzle1	Puzzle 2
A*	Cost: 116	Cost: 89
DFS	Cost: 51841	Cost: 2392

A* has significantly less cost than DFS. It is a better solution since it is solves the puzzle more efficiently.

	Puzzle1	Puzzle 2
A*	states expanded: 300422	states expanded: 85342
DFS	states expanded: 196834	states expanded: 6184

DFS expands less states than A*.

f) We should use A* to solve the HuaRongDao puzzle, because it gives the optimal solution.

DFS is better in terms of speed, it expands less states to find a solution, but it is brute force. It may take many many steps to reach that goal state.

A* finds the optimal solution, but it expands significantly more states. It takes much longer to find the solution.