1.

A. Hill climbing search:

Hill climbing attempts to maximize an objective function f(x) where x is a state of the problem. Iteratively, the agent generates solutions from current state which have higher f(x) and chooses the highest one as the next state. In this way, the agent will always find the local optima of the problem.

As for TSP, the initial state can be a random sequence of paths through all the nodes. For example, in a five nodes problem, initial state [2,3,1,4,5] means we move towards 2->3->1->4->5 as the initial solution. And then, we use the negated sum of all path cost as objective function f(x), so agent can minimize path cost by maximize objective function. At every state, the agent generates solutions by swap any two nodes in the state sequence and then calculate f(x). Afterwards, the agent will choose the solution which has highest f(x) and change current state to that solution.

B. Simulated annealing

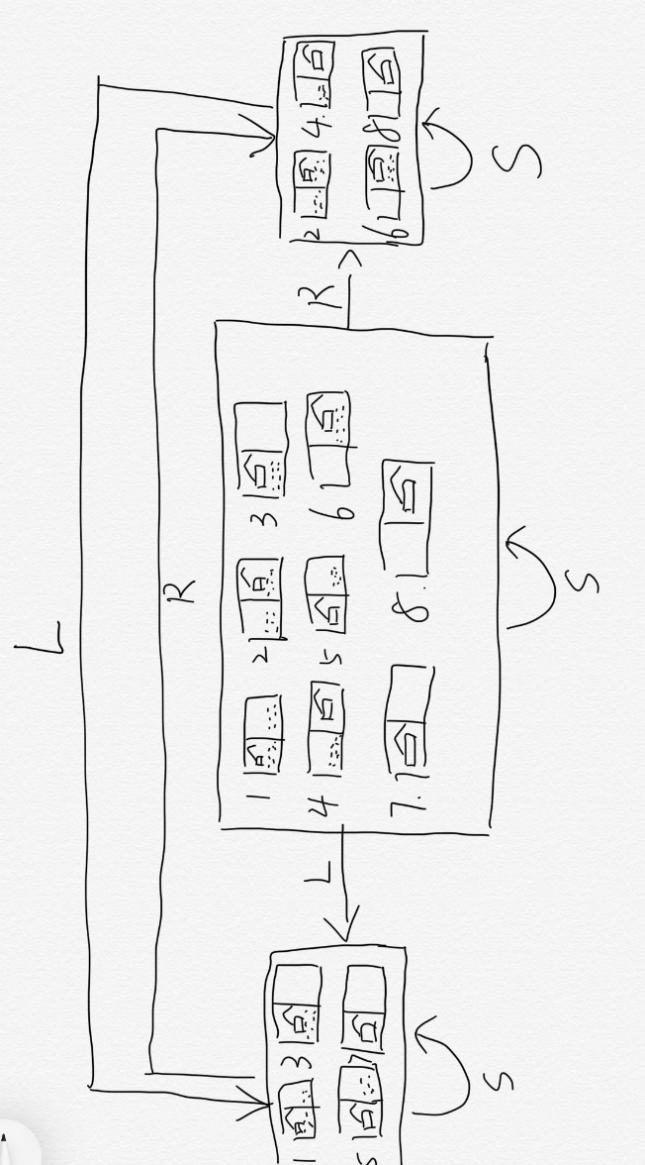
Simulated annealing aims to give agent a chance to jump out from local optimum and continues to find global optimum. It uses “temperature” as a parameter to decide a chance whether agent will jump out to search other paths when it encounters the local optimum. The temperature will decrease every step.

The process of annealing starts with a path which simply lists all of the nodes in the order their positions were randomly selected. At each temperature step, a number of solutions are generated and agent will find the solution which has higher value. If there is no better solution, the agent will have a probability to go on searching. The probability is determined by the value of temperature at this step.

2.

The states are any configuration of the pieces. As for actions, there are discrete operations such as removing any piece or connecting it anywhere else. And there are also continuous operations such as rotating the pieces for any degree. We define the value v = a\*number of gaps + b\*number of misconnected pieces + c\*sum of the sizes of gaps and at every temperature step T, the agent has value and the value in next state is ,the agent has a probability to go to the next state.

3.



According to the belief-state in the picture above, the agent will fall in loop whatever action it does. And there is no solution in the whole belief-state, so this problem is unsolvable.

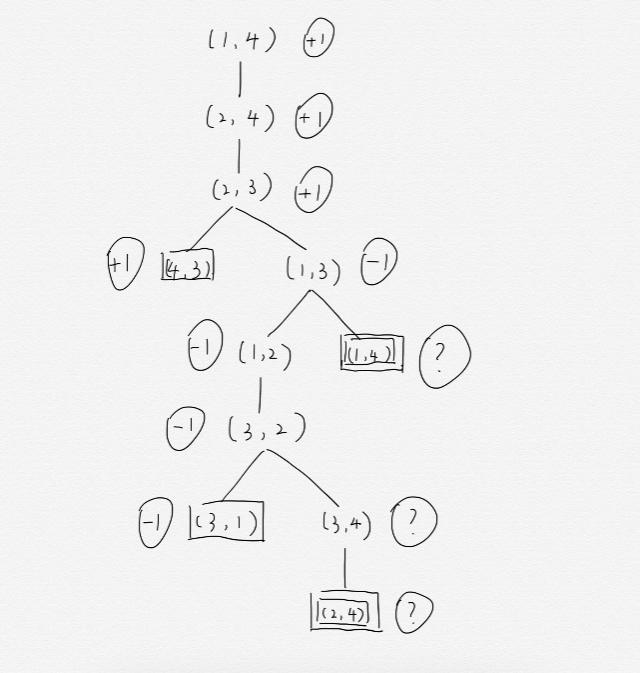
4.

The online search problem can be viewed as an offline search in belief-state space. As for the initial position of the agent is fixed, the initial states are all the possible environment configurations. Because the agent does not know where the internal walls are, and there are 12 possible positions of the internal walls, so the initial belief-state contains configurations. The agent can be in any of different squares so the amount of initial belief states are . Because the map is a 3\*3 grid, so the amount of belief states is .

5.

6.

A.



Because ? means uncertainty and every player wants to win the game. So I suppose min(?,-1) = -1 and max(?,1) = 1.

B.

For two-player non-zero-sum games, the minimax algorithm works as for multiplayer games. The evaluation function is a vector of values for every player. And when the player turns to move, the evaluation function will choose which vector is the highest value. As for alpha-beta pruning, I think it will work as well, because the two players are cooperative and the unexamined leaf node might be optimal for both of them.

7.

1. The game tree will be constructed as following. At each state, the player can choose to drop one disc from any column. Suppose there are 7 columns and any of them are not filled, then the player will have 7 different legal actions to move. Use all of the states as node and all of the legal actions as edges, we can construct a game tree.
2. In the Connect Four game, the two players’ goals are conflict, each of them tries to win by firstly getting four of her discs in a row.

Pseudocode:

function Connect\_Four(state,depth,turn) returns an value:

if(Terminal-Test(state) or depth == 0)

return Value(state)

else{

if(turn == max\_turn)

v = -inf

for each action in Actions(state,turn){

v = max(v,Connect\_Four(Result(state,action),depth-1,min\_turn))

return v

}

else if(turn == min\_turn)

v = inf

for each action in Actions(state,turn){

v = min(v,Connect\_Four(Result(state,action),depth-1,max\_turn))

}

return v

}

Diagram:

1. In this game, alpha-beta decreases nodes evaluated in minimax algorithm. It gives a better subtree and gives an ability to do deeper search.
2. The Cutting Off Search will have an evaluation function to present the winning chance of one state rather than minimax value. In Connect Four, I think the heuristic function can be the count of how many four-cell in all the four-cell combinations have a chance to win the game.