

iyr/**

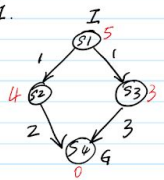
- Please feel free to write down your answers.
- Open discussion is welcome
- Please do not delete the contents here, leave your comment and your solution if you think something is not accurate.

**/

Question 1

(a). [6 marks] You are given a graph with 4 states: s_1 is the initial state and is connected to s_2 and s_3 ; s_2 is connected to s_4 , and s_3 is connected to s_4 . s_4 is the goal state. Is it possible to design a cost function $c(s)$ and heuristic function $h(s)$ such that A* algorithm will not find an optimal solution? Justify your answer by showing the execution of the algorithm with the cost and heuristic function you've chosen. To show the working of the algorithm, show how the open and closed lists evolve after each node expansion.

Question 1.
(a)



	cost	$h(s)$
$s_1 \rightarrow s_2$	1	5
$s_1 \rightarrow s_3$	1	4
$s_2 \rightarrow s_4$	2	3
$s_3 \rightarrow s_4$	3	0

In the case that h is not admissible, A* will not be optimal

The h in the table is neither admissible nor consistent. As such, A* will not find optimal solution

	open	closed	best-g
Iteration 0	$s_1(5)$	\emptyset	nil
Iteration 1	$s_2(4), s_3(5)$ <i>new</i>	$s_1(5)$	5
Iteration 2	$s_4(4), s_2(5)$ <i>new</i>	$s_1(5), s_3(4)$	4
Iteration 4	$s_4(5)$	$s_1(5), s_3(4), s_4(0)$ <i>Goal STOP</i>	4

Finally, the path explored by A* is $s_1 \rightarrow s_3 \rightarrow s_4$
not optimal

Notes:

1. Consistent \rightarrow admissible \rightarrow goal-aware
goal-aware \rightarrow safe

```
A* (with duplicate detection and re-opening)
open := new priority queue ordered by ascending  $g(\text{state}(n)) + h(\text{state}(n))$ 
open.insert(make-root-node(nil))
closed :=  $\emptyset$ 
best-g := 0 // maps states to numbers
while not open.empty():
    n := open.pop-min()
    if state(n) in closed:
        // re-open if better than previous state but worse g
        // are behind n in open, and will be skipped when their turn comes
        closed := closed  $\cup$  {state(n)}
    best-g[state(n)] := g(n)
    if is-goal-state(n): return extract-solution(n)
    for each (n', e) in succ(state(n)):
        n' := make-node(n, n', e)
        if h[state(n')] <  $\infty$ : open.insert(n')
return unsolvable
```

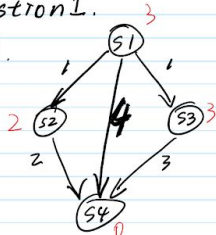
reopen when hit duplicates

frank

(b). [6 marks] Given an optimal heuristic h^* , can A* expand more nodes than Hill Climbing with the same heuristic? Use an example to explain your answer using a graph of at most 4 nodes.

Question 1.

b).



	$h(s)$
s1	3
s2	2
s3	3
s4	0

A* will expand $s1 \rightarrow s2 \rightarrow s4$

Hill climbing will expand $s1 \rightarrow s4$

Notes:

Hill Climbing

```

Hill Climbing
function hillClimbing(problem)
    node ← makeNode(problem.start())
    while not isGoalNode(node)
        if not isGoalNode(node)
            return failure
        node ← bestChild(node)
    return node

```

Remarks:

- Minimizes search only if $h(x) > 0$ for $x \in S^*$.
- Is this complete or optimal? No.
- Can always get stuck in local minima where immediate improvements of $h(x)$ are not possible.
- Many variations: hill-climbing strategies, restarts, ...

Enhanced Hill Climbing

```

Enhanced Hill Climbing: Procedure Improve
function improve(node)
    if not isGoalNode(node)
        return node
    while not isGoalNode(node)
        if not isGoalNode(node)
            return failure
        node ← bestChild(node)
    return node

```

Remarks:

- Minimizes search only if $h(x) > 0$ for $x \in S^*$.
- Is this complete? No.
- Is this optimal? In general, no. Under particular circumstances, yes. Assume that h is good enough.
- Procedure improve fails: no state with strictly smaller h -value reachable from a fixed start assumption; goal not reachable from s .
- This can, for example, not happen if the state space is undirected, i.e., if for all transitions $s \rightarrow t$ in this there is a transition $t \rightarrow s$.

Enhanced Hill Climbing, full

```

Enhanced Hill Climbing
function enhancedHillClimbing(problem)
    node ← makeNode(problem.start())
    while not isGoalNode(node)
        if not isGoalNode(node)
            return failure
        node ← improve(node)
    return node

```

Remarks:

- Minimizes search only if $h(x) > 0$ for $x \in S^*$.
- Is this complete? No.
- Is this optimal? In general, no. Under particular circumstances, yes. Assume that h is good enough.
- Procedure improve fails: no state with strictly smaller h -value reachable from a fixed start assumption; goal not reachable from s .
- This can, for example, not happen if the state space is undirected, i.e., if for all transitions $s \rightarrow t$ in this there is a transition $t \rightarrow s$.

Question 2

This question concerns classical planning. (13 Marks)

A robot can move between room A and room B. It has 1 hand that can pick-up or put-down 1 ball at a time. Every action has cost 1. Initially the robot starts at position A with 3 balls, and the goal is to have the 3 balls at room B. The robot can pick-up a ball if both are at the same location.

In answering the sub-questions below, you are allowed to use variables as arguments for the actions (action schemes), specifying the values of the variables. Note: it is not compulsory to use PDDL syntax, as long as you can convey the main ideas.

(a). [6 marks] Describe briefly in STRIPS how to model the actions 'move(from, to)' and 'pickup(from, ball)' and 'putdown(from, ball)' using the predicates robot(l), ball(b,l) holding(b) and free for each possible location l and ball b, where robot(l) and ball(b,l) stand for the robot and ball location, and holding(b) and free stand for the hand holding a ball and the hand when is free. You can use extra predicates if needed.

Question 2.



(a) STRIPS $P = \langle F, O, I, G \rangle$, Rooms = {A, B}, Balls = {b₁, b₂, b₃}

Atoms

$F = \{ \text{robot}(t), \text{ball}(b, t), \text{holding}(b), \text{free} \mid t \in \text{Rooms}, b \in \text{Balls} \}$

Operations (Actions)

$A = \{ \text{move}(x, y) :$

- Prec: robot(x)
- Add: robot(y)
- Delete: robot(x)

$\mid x, y \in \text{Rooms} \}$

$A = A \cup \{ \text{pickup}(\text{from}, b) :$

- Prec: ball(b, from), free, robot(from)
- Add: holding(b)
- Delete: ball(b, from), free

$\mid b \in \text{Balls}, \text{from} \in \text{Rooms} \}$

$A = A \cup \{ \text{putdown}(\text{to}, b) :$

- Prec: holding(b), robot(to)
- Add: ball(b, to), free
- Delete: holding(b)

$\mid b \in \text{Balls}, \text{to} \in \text{Rooms} \}$

Initial State

$I = \{ \text{free}, \text{ball}(b, A), \text{robot}(A) \mid b \in \text{Balls} \}$

Goal state

$G = \{ \text{ball}(b, B) \mid b \in \text{Balls} \}$

A Basic Language for Classical Planning: Strips

- A problem in STRIPS is a tuple $P = \langle F, O, I, G \rangle$:
 - F stands for set of all atoms (Propositions)
 - O stands for set of all operators (actions)
 - $I \subseteq F$ stands for initial situation
 - $G \subseteq F$ stands for goal situation
- Operators $o \in O$ represented by
 - the Add list $\text{Add}(o) \subseteq F$
 - the Delete list $\text{Del}(o) \subseteq F$
 - the Precondition list $\text{Pre}(o) \subseteq F$

(b). [6 marks] Model in STRIPS the Initial State s_0 and compute $h^{\max}(s_0, G)$ and $h^{\text{add}}(s_0, G)$ for the goal G of the problem.

Question 2. (b)

	robot(A)	robot(B)	holding(b1)	holding(b2)	holding(b3)	free	ball(b1, A)	ball(b1, B)	ball(b2, A)	ball(b2, B)	ball(b3, A)	ball(b3, B)
0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	1	1	1	0	0	0	0	0	0	0
2	0	1	1	1	1	0	0	add: 3 max: 2	0	add: 3 max: 2	0	add: 3 max: 2
3	0	1	1	1	1	0	0	add: 3 max: 2	0	add: 3 max: 2	0	add: 3 max: 2

$$h^{\max}(s_0, G) = \max(2, 2, 2) = 2$$

$$h^{\text{add}}(s_0, G) = \text{sum}(3, 3, 3) = 9$$

Notes:



$$h^{\max} \leq h^* \leq h^{\text{add}} \leq h^* \leq h^{\text{add}}$$

h^{add} can be larger or smaller than h^*
 h^{add} is similar to h^* but it will never overcount

h^* - the cost of an optimal plan for S , perfect heuristic (optimal)
 or indicates no plan exists

h^* - the optimal delete relaxation heuristic (the minimum effort needed to reach the goal) (hard to compute (NP))

h^{\max} - to estimate h^* optimistically by the most costly singleton sub-goal (too small for too optimistic)

h^{add} - pessimistically by summing over all singleton sub-goals (might overcount)

h^{add} - computed based on the plan extracted from h^{\max} or h^{add} (never overcount)

Extra exercise: calculate hff (Optional)

Rplan = {move(x,y), putdown(x,b), pickup(x,b) | x, y ∈ {A,B}, b ∈ {b0, b1, b2}}

00hff = 7

Extra exercise: h^+

	robot(A)	robot(B)	holding(b1)	holding(b2)	holding(b3)	free	ball(b1, A)	ball(b1, B)	ball(b2, A)	ball(b2, B)	ball(b3, A)	ball(b3, B)
0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	1	1	1	0	0	0	0	0	0	0
2	0	1	1	1	1	0	0	add: 3 max: 2	0	add: 3 max: 2	0	add: 3 max: 2
3	0	1	1	1	1	0	0	add: 3 max: 2	0	add: 3 max: 2	0	add: 3 max: 2

$move(A, B)$ $pickup(A, b1)$ $pickup(A, b2)$ $pickup(A, b3)$ $putdown(B, b1)$ $pd(B, b2)$ $pd(B, b3)$

Step 1: best supporter function

best supporter func for ball(b1, B)

$$putdown(B, b1) + pecon \left\{ \begin{array}{l} holding(b1) \\ robot(B) \end{array} \right\} \left. \begin{array}{l} h^{add} = 3 \\ h^{max} = 2 \end{array} \right\}$$

same for others

Step 2: Relaxed Plan Extraction

Initial State

$$I = \{ free, ball(b, A), robot(A) \mid b \in Balls \}$$

Goal state:

$$G = \{ ball(b, B) \mid b \in Balls \}$$

Code:

Iteration	Open	closed	RPlan
0	$G \setminus I = \{ ball(b1, B), ball(b2, B), ball(b3, B) \}$	\emptyset	\emptyset
1	$ball(b1, B), ball(b3, B), holding(b1), robot(B)$	$ball(b1, B)$	$putdown(B, b1)$
2	$ball(b2, B), holding(b1), robot(B), holding(b2)$	$ball(b1, B), ball(b2, B)$	$putdown(B, b2)$
3	$ball(b3, B), robot(B), holding(b2), holding(b3)$	$ball(b1, B), ball(b2, B)$	$putdown(B, b3)$
4	$robot(B), holding(b2), holding(b3)$	$holding(b1)$	$pickup(A, b1)$ $pickup(A, b2)$ $pickup(A, b3)$ $move(A, B)$

$\Rightarrow 7 \text{ actions}$
 $h^+ = 7$

Relaxed Plan Extraction for state s and best-supporter function bs

Open := $G \setminus s$; Closed := \emptyset ; RPlan := \emptyset

while Open $\neq \emptyset$ do:

 select $g \in$ Open

 Open := Open $\setminus \{g\}$; Closed := Closed $\cup \{g\}$;

 RPlan := RPlan $\cup \{bs(g)\}$; Open := Open $\cup (pre_{bs(g)} \setminus (s \cup Closed))$

endwhile

return RPlan

Question 3

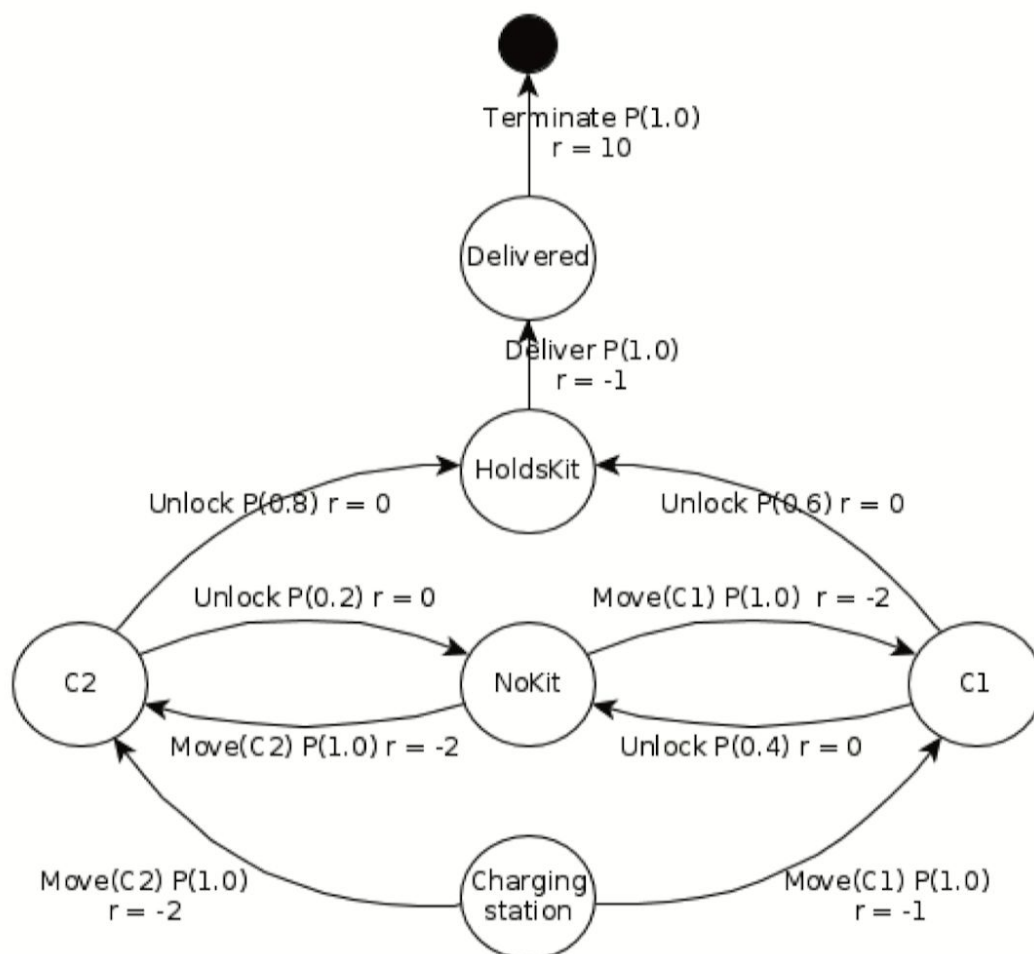
Question 3

(12 Marks)

Consider a robot called *MedAssist*, which is tasked with assisting medical staff in a hospital. Its primary task is to carry medical kits between rooms.

MedAssist is at its charging station fully charged and it receives a request to take a blood-testing kit to a doctor, who is in room 6.12. One blood-testing kit is in cupboard C1 and another is in cupboard C2. Cupboard C1 is closer to the charging station than cupboard C2. It would cost 1 unit (reward of -1) to move to cupboard C1 from the charging station and 1 unit to move from cupboard C1 to room 6.12. It would cost 2 units to move from the charging station to cupboard C2 and 1 unit to move from cupboard C2 to room 6.12. However, *MedAssist* assesses that there is a 60% chance that cupboard C1 will be locked, while only an 80% of cupboard C2 being locked. *MedAssist* is a bit stupid, so if it notes that a cupboard is locked, it will not update these probabilities. Delivering the kit successfully to room 6.12 costs 1 unit, but this terminates the MDP, receiving a reward of 10.

The following diagram shows the transition probabilities, rewards, and values:



- (a). [3 marks] Assume that MedAssist has calculated the following *non-optimal* value function V for this problem using value iteration with $\gamma = 1.0$. After 3 iterations it arrives at the following:

State	Iteration			
	1	2	3	4
V(Delivered)	10.0	10.0	10.0	
V(HoldsKit)	0.0	9.0	9.0	
V(NoKit)	0.0	0.0	0.0	
V(C1)	0.0	0.0	5.4	
V(C2)	0.0	0.0	7.2	
V(CS)	0.0	0.0	0.0	

If MedAssist is at the charging station (CS), which action should it choose to maximise its reward in the next state? Assume we are using the values for V after three iterations. Show full working to demonstrate that you understand the concepts.

$$\pi(CS) = \text{Move}(C2)$$

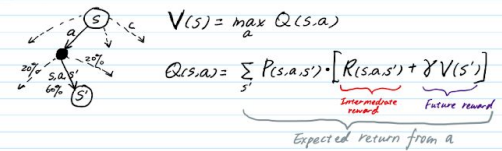
(a) $\gamma = 1.0$

$$V(CS) = \max \begin{cases} Q[CS, \text{move}(C1)] = 4.4 \\ Q[CS, \text{move}(C2)] = 5.2 \text{ choose to move}(C2) \end{cases}$$

$$Q[CS, \text{move}(C1)] = 1 \cdot [-1 + 1 \cdot \underbrace{V(C1)}_{5.4}] = 4.4$$

$$Q[CS, \text{move}(C2)] = 1 \cdot [-2 + 1 \cdot \underbrace{V(C2)}_{7.2}] = 5.2$$

Notes:



- (b). [5 marks] Complete the values of these states for iteration 4 using value iteration. Fill the answers into the table, but show your working in the box. The first two have been completed already.

State	Iteration			
	1	2	3	4
V(Delivered)	10.0	10.0	10.0	10.0
V(HoldsKit)	0.0	9.0	9.0	9.0
V(NoKit)	0.0	0.0	0.0	
V(C1)	0.0	0.0	5.4	
V(C2)	0.0	0.0	7.2	
V(CS)	0.0	0.0	0.0	

$$V_4(NK) = \max(-2+5.4, -2+7.2) = 5.2$$

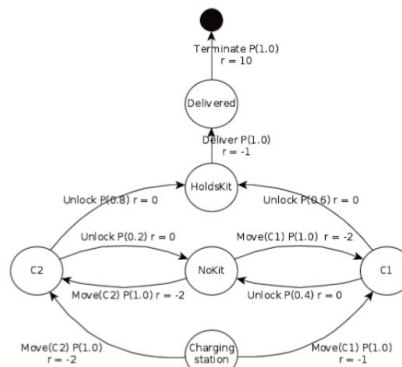
$$V_4(C1) = 0.6(9) + 0.4(0) = 5.4$$

$$V_4(C2) = 0.8(0) + 0.2(0) = 7.2$$

$$V_4(CS) = \max(-2+7.2, -1+5.4) = 5.2$$

(b)

State	Iteration			
	1	2	3	4
V(Delivered)	10.0	10.0	10.0	10.0
V(HoldsKit)	0.0	9.0	9.0	9.0
V(NoKit)	0.0	0.0	0.0	5.2
V(C1)	0.0	0.0	5.4	5.4
V(C2)	0.0	0.0	7.2	7.2
V(CS)	0.0	0.0	0.0	5.2



$$V(\text{NoKit}) = \max \left\{ \begin{aligned} Q(\text{NoKit}, \text{move C1}) &= 1 \cdot (-2 + 1 \cdot \underbrace{V(\text{C2})}_{7.2}) = 5.2 \\ Q(\text{NoKit}, \text{move C2}) &= 1 \cdot (-2 + 1 \cdot \underbrace{V(\text{C1})}_{5.4}) = 3.4 \end{aligned} \right\} \max = 5.2$$

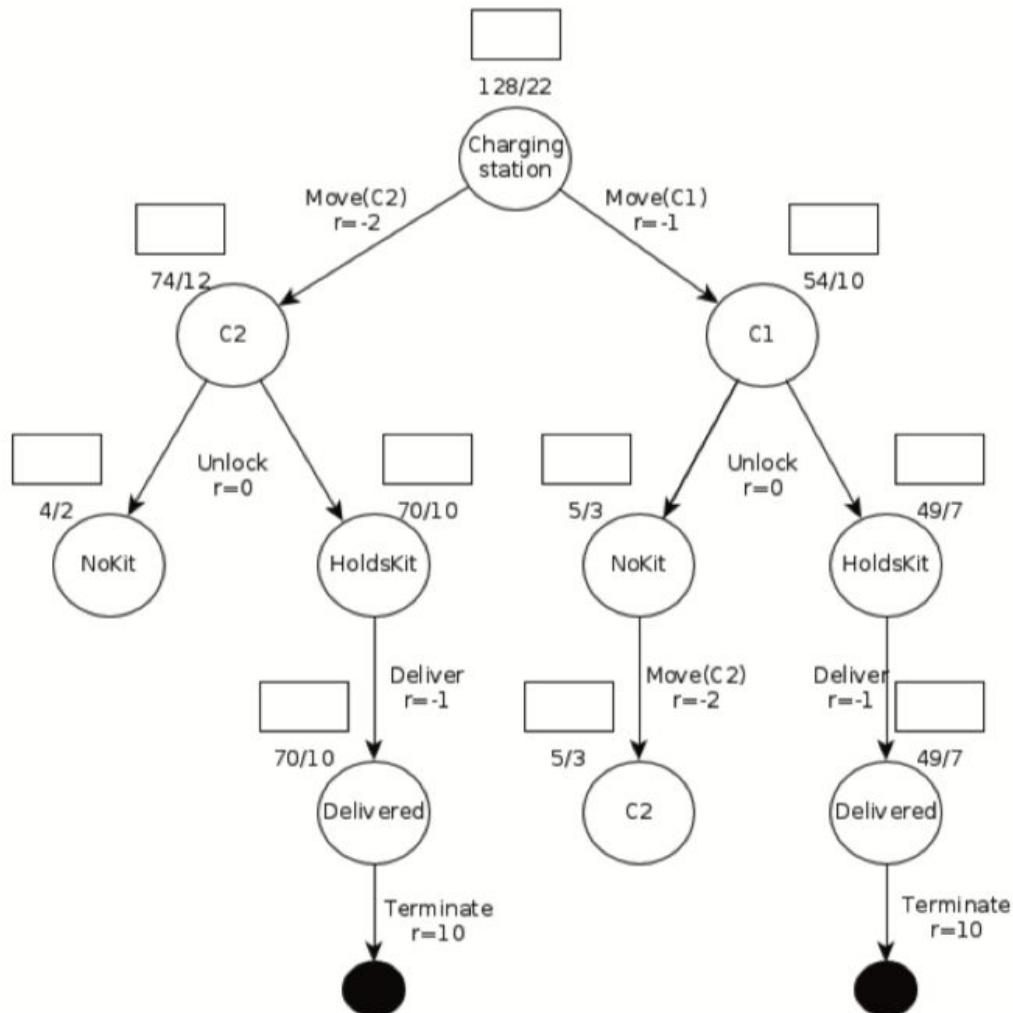
$$V(\text{C1}) = \max \{ Q(\text{C1}, \text{unlock}) \} = \left\{ \begin{aligned} 0.6 \cdot (0 + 1 \cdot \underbrace{V(\text{HoldsKit})}_{9}) &= 5.4 \\ + \\ 0.4 \cdot (0 + 1 \cdot \underbrace{V(\text{NoKit})}_{0}) &= 0 \end{aligned} \right\} \text{sum} = 5.4$$

$$V(\text{C2}) = \max \{ Q(\text{C2}, \text{unlock}) \} = \left\{ \begin{aligned} 0.8 \cdot (0 + \underbrace{V(\text{CS})}_{0}) &= 7.2 \\ + \\ 0.2 \cdot (0 + \underbrace{V(\text{NoKit})}_{0}) &= 0 \end{aligned} \right\} \text{sum} = 7.2$$

- (c). **[4 marks]** You decide to implement a Monte-Carlo Tree Search (MCTS) approach for MedAssist to scale better. Below is a tree from MCTS in which 22 roll-outs have been performed. The black nodes represent end states (or terminal states). The score/payoff received for each roll-out is the cumulative reward for the path. That is, the value for any path that terminates is 10 (the reward for successful termination) plus the other rewards accumulated along the path (including negative rewards). The notation X/Y indicates that Y number of roll-outs have been performed, with a cumulative score of X , and thus X/Y is the average score and value for that state.

Using a greedy multi-armed bandit strategy (that is, ϵ -greedy with $\epsilon = 0$), and using the values on the tree as the base policy (that is, do not use the probabilities from the previous question), perform a complete iteration of the Monte-Carlo Tree Search algorithm. That is, perform *selection*, *expansion* (if required), and *back-propagation*. Note that simulation is not required: the selection/expansion should reach a terminal state.

To reach your answer, fill in the back-propagated values in the boxes on the diagram below. For those states not encountered during the iteration, leave the box empty.



NOT SURE

CS 141/24

C2 83/13

NK 4/2

HK 79/11

D 80/11

C1 61/11

NK 3/3

C2 5/3

NK 58/8

D 59/8

(an problem: cs:143/24? For the right side the nokit may be the same without -2)?

Another version(seems to be wrong according to the discussion on the right hand side):

CS 128/23

C2 74/13

C2->HoldsKit 79/11

C2->HoldsKit->Delivered 80/11

Sorry I don't understand why C2 is 74/13 rather than 84/13

Corrected version:

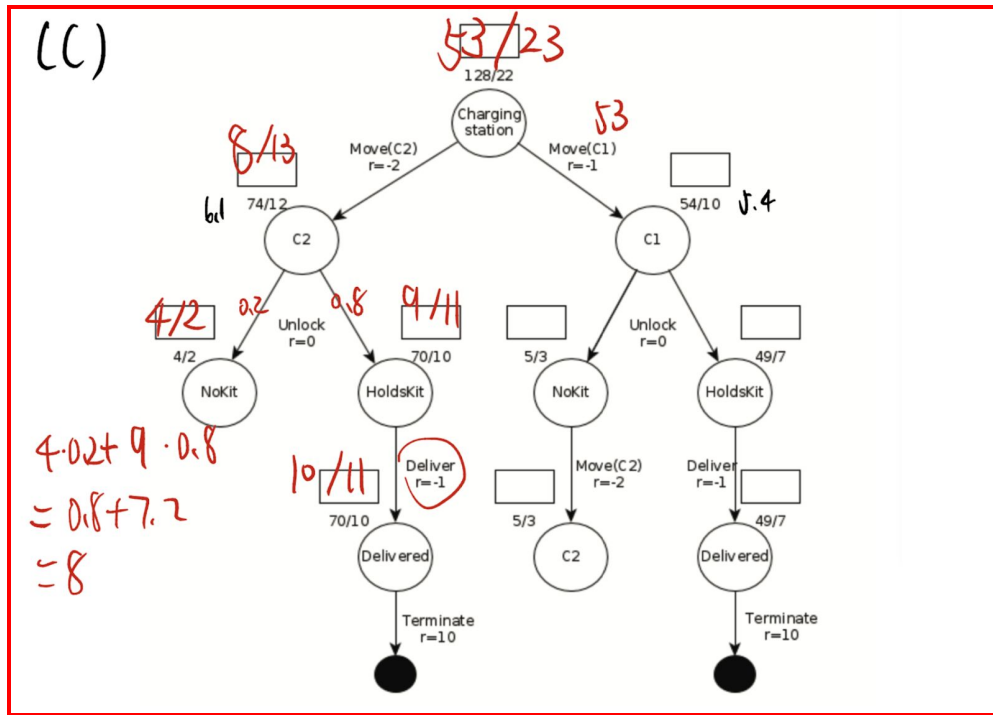
CS 137/23

C2 83/13

C2->HoldsKit 79/11

C2->HoldsKit->Delivered 80/11

Not quite sure



Question 4

The hospital are so happy with the new MedAssist robot that they would like to order a second one. This requires two MedAssist robots to coordinate. On their first day, they receive requests for two kits, kit K and L, to be delivered to three different rooms R1, R2, and R3. They clearly cannot deliver two kits to three rooms simultaneously, so they need to each decide which rooms to go to. They cannot communicate.

Robot A is carrying kit K, while robot B is carrying kit L. Some of the instruments in kit K are also in kit L, so delivering both does not double the total reward.

The payoffs for each arriving in the different rooms are outlined below.

Circle all *pure strategy equilibria* of this game, if any exist.

		Robot B (Kit L)		
		R1	R2	R3
Robot A (Kit K)	R1	4, 4	5, 0	5, 5
	R2	5, 5	5, 0	5, 5
	R3	0, 5	0, 0	0, 5

All three (5, 5) (R1, R3), (R2, R1) (R2, R3)

Question 4

		Robot B (Kit L)		
		R1	R2	R3
Robot A (Kit K)	R1	4, 4	5, 0	5, 5
	R2	5, 5	5, 0	5, 5
	R3	0, 5	0, 0	0, 5

	Dominant Strategy
Robot A	R2
Robot B	R3

pure strategy Equilibria

circled in

Question 5

The two MedAssist robots need are in a second scenario. Robot A has two kits, K and M, while Robot B has just one kit: L. Robot A is ordered to go to room R1 and Robot B to room R2. However, Robot R2 needs kits L and M. Robot A does not need kit M. The two robots need to determine whether to meet to swap kit M in the hallway between the two rooms. However, this will incur a cost.

If Robot A delivers kit K to room R1, it will receive a payoff of 7. However, to first meet in the hallway will cost 2, giving a payoff of 5.

If Robot B delivers only kit L to room R1, it will receive a payoff of 4. To first meet in the hallway will cost 2, however, if the robot first meets and then delivers kits L and M, it will receive a payoff of 6 (8 for delivering the kits minus the cost of 2).

The normal form matrix for this is below.

		Robot B	
		Meet	Not meet
Robot A (kit K)	Meet	5, 6	3, 4
	Not meet	7, 2	7, 4

(a). [5 marks] Is there a unique Nash equilibrium in this game? Explain your working.

(b). [5 marks] Assume that a new secure network is installed, allowing the MedAssist robots to communicate. Robot B decides that it will try to improve its utility by announcing its strategy ('Meet' or 'Not meet') to Robot A, allowing Robot A to subsequently choose its strategy. Assuming the payoffs remain the same as the previous question, which option should Robot B choose? Does this improve its utility? Show your working.

a). (7, 4)

Question 5.

		Robot B	
		Meet	Not meet
Robot A (kit K)	Meet	5, 6	3, 4
	Not meet	7, 2	7, 4

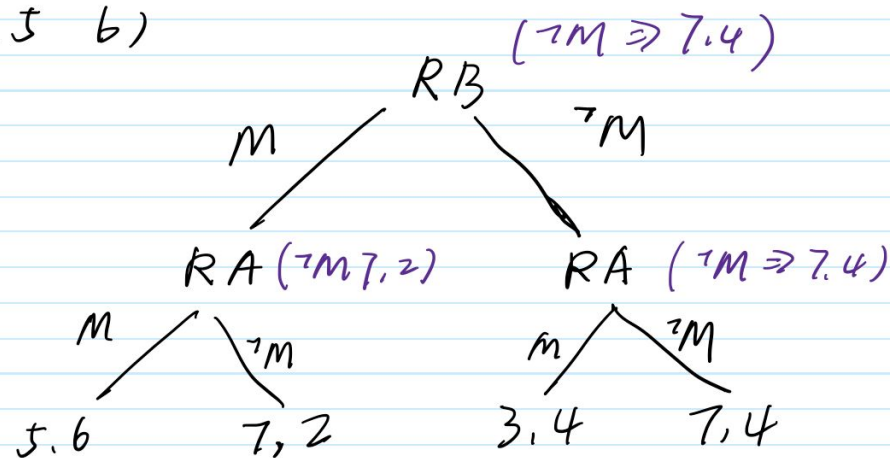
The Nash equilibrium is (Robot A : Not meet , Robot B : not meet)

If RB thinks NM will be the action RA takes, RB will make decision to take NM, which will result in highest available payoff for RB. Same reason for RA.

b). Still "Not meet". Since A has a dominant strategy (Not meet), what B choose does not influence A's decision.

This question is asking about extended form game, maybe should draw a tree and do backward induction.

Question 5 b)



B choose not meet, same as last one, no change on uti.