National University of Singapore
School of Computing

Semester 1, AY2021-22

CS4246/CS5446 AI Planning and Decision Making

Assignment 2 Solution Sketches

1 Homework Assignment (LumiNUS Quiz)

[Preparation]: *For this problem, while the numbers may be worked out manually, you are encouraged to use one of the academic or free trial versions of computer programs for decision modeling, e.g., Bayesfusion GeNIE modeler or SMILE engine, Netica, Syncopian DPL 9, TreeAge, etc., to manipulate the influence diagrams and the decision trees.*

1.1 Homework Problem 1: Decision Theory

[10 marks] The CEO of Orange Computer Inc., Tim Bean is facing a dilemma: The pandemic has caused serious disruptions in the main supply chain of its hit digital watch, iTick, while the demand is still so strong that the current production capacity is unable to keep up. A decision has to be made between two alternatives:

i. Keep to the current production unit

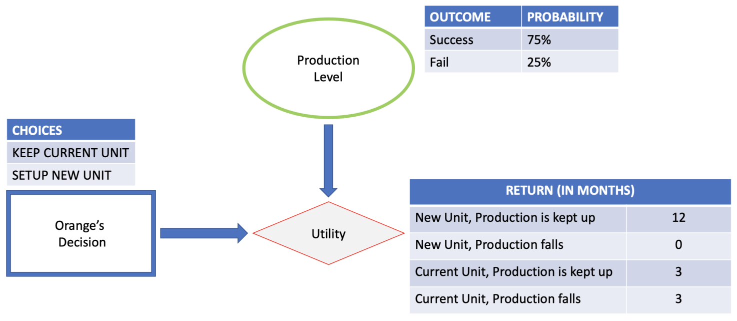
ii. Set up a new production unit

Without any increase in the production, production will definitely fall below demand in 3 months. However, since setting up a new production unit would require a complete rearrangement of work with the current production line, there is a 0.25 probability that production will fall below demand immediately. If the new production unit is successful, production is likely to be kept up for 1 year, at that time the product will be replaced with a new version.

a) *[2 marks]* Draw an influence diagram for this simple decision problem. Show the structure and the numerical parameters.

Solution:

Please remember these are not exact solutions, but a guide. Comments and suggestions are included in the boxes.



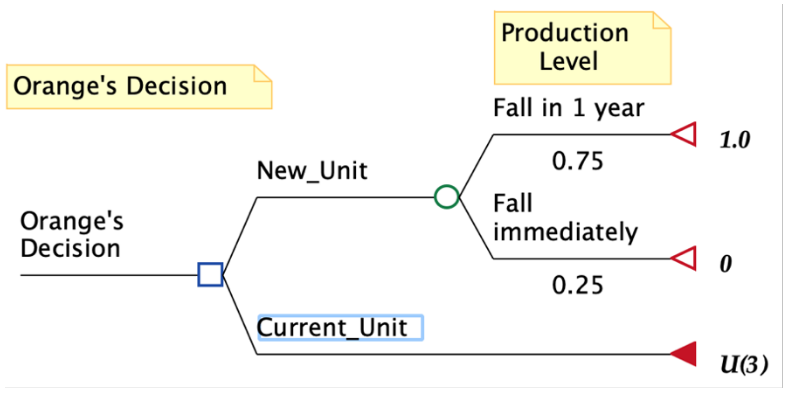
OUTCOME	PROBABILITY
Success	75%
Fail	25%

CHOICES
KEEP CURRENT UNIT
SETUP NEW UNIT

RETURN (IN MONTHS)	
New Unit, Production is kept up	12
New Unit, Production falls	0
Current Unit, Production is kept up	3
Current Unit, Production falls	3

b) *[2 marks]* Draw a decision tree for this simple decision problem. Show the structure and the numerical parameters.

Solution:



c) *[2 marks]* Let U(x) denote the company's utility function, taking into account its risk attitude and the potential revenue, where x is the number of months until production falls below demand. Assume that U(12)=1.0 and U(0)=0.0, how low can the company's utility for falling behind demand in 3 months be and still have the new production unit option be preferred?

Solution:

Expected utility of having new production unit
$= P(fallingbelowdemand) * U(0) + P(keepinguptodemand) * U(12)$
$= 0.25 * 0 + 0.75 * 1 = 0.75$ Expected utility of staying with current unit is U(3).
To find U(3):

i If the company is **risk neutral**, U(3) = 0.75 for the company to be indifferent between the new production unit and the current production unit ii If the company is **risk averse**, U(3) $>$ 0.75 for the company to prefer the new production unit.

iii If the company is **risk seeking**, U(3) $<$ 0.75 for the company to prefer the new production unit.

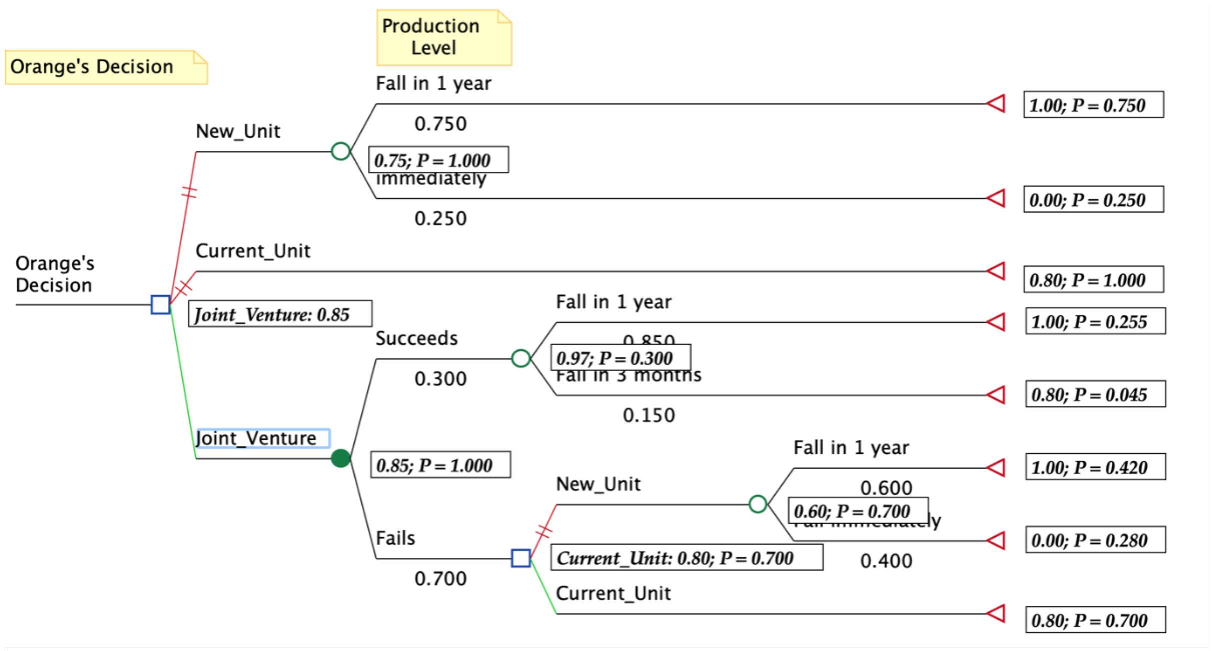
- You need to discuss three different risk attitudes separately.

For the rest of the problem, assume that U(3)=0.70. Now the main supplier for iTick's components, The Sparkling Chips, offers to form a joint venture with Orange to speed up production of the iTick. CEO Bean has to consider the new option (in addition to the 2 options above): If the new joint venture succeeds, there is a 0.15 chance that the production will fall below demand in 3 months' time, or production will be kept up for at least 1 year. If the joint venture fails, however, Orange can still consider the option of adopting the original plan (without working with the Sparkling Chips) for a new production unit, which now has 0.4 chance of falling below demand immediately, or otherwise able to keep up production for 1 year. There is a probability of 0.3 that the joint venture will succeed.

d) *[4 marks]* What is the best option for CEO Bean? Why?

Solution:

The best decision is to go for the joint venture; if the joint venture fails, keep working with the



current production unit.

- The company should also consider a new unit or the current unit after the venture failed.

1.2 Homework Problem 2: Markov Decision Process

[10 marks]

a) *[4 marks]* Can any finite search problem be translated exactly into a Markov decision problem such that an optimal solution of the latter is also an optimal solution of the former?

If so, explain precisely how to translate the problem and how to translate the solution back; if not, explain precisely why not (i.e., give a counterexample).

Solution:

A finite search problem is defined by an initial state s_0, a successor function $Succ(s)$ that returns a set of action–state pairs, a step cost function $c(s, a, s')$, and a goal test. An optimal solution is a least-cost path from s_0 to any goal state.

To construct the corresponding MDP, define $R(s, a, s') = -c(s, a, s')$ unless s is a goal state, in which case $R(s, a, s') = 0$; define $T(s, a, s') = 1$ if $(a, s') \in Succ(s)$; and $\gamma = 1$. An optimal solution to this MDP is a policy that follows the least-cost path from each state to its nearest goal state.

- You need to specify ”how to translate the solution back”
- The reward function should be defined as the negation of the cost function, as we will maximize the rewards in MDP but will minimize the costs in the FSP.

b) *[6 marks]* Sometimes MDPs are formulated with:

i a reward function $R(s)$ that depends only on the current state s, or
ii a reward function $R(s, a)$ that also depends on the action a taken in the state, or
iii a reward function $R(s, a, s')$ that also depends on the action a and the outcome state s'.

Write the Bellman equations for these formulations.

Solution:

i $R(s) + \gamma \max_{\alpha \in A(s)} \Sigma_{s'} P(s'|s, a)U(s')$
ii $\max_{\alpha \in A(s)} [R(s, a) + \gamma \Sigma_{s'} P(s'|s, a)U(s')]$
iii $\max_{\alpha \in A(s)} \Sigma_{s'} P(s'|s, a)[R(s, a, s') + \gamma U(s')]$

2 Programming Assignment (aiVLE submission)

2.1 Programming problem 1: Value Iteration

[10 marks] In the crossing the road task in Assignment 1, we assumed all the cars move with a constant speed of -1. Most of the times, this assumption does not hold. There is inherent noise in the car speeds due to reasons like bumpy roads, uneven driving, etc. This non-deterministic behaviour is not captured in the PDDL format.

For problems that are not very large, planning algorithms like Value Iteration can be used to handle non-deterministic behaviours. In this task, we model the problem as a Markov Decision Process and use Value Iteration algorithm to find the optimal policy.

We have provided the script which models the problem as an MDP. Every state in this MDP is represented as a tuple of features: $\{\text{Agent}_x, \text{Agent}_y, \text{Car}^1_x, \text{Car}^1_y, \cdots, \text{Car}^N_x, \text{Car}^N_y, \text{isTerminal}\}$, where:

- $\text{Agent}_x, \text{Agent}_y$ denote the x and y coordinates agent.
- $\text{Car}^N_x, \text{Car}^N_y$ denote the x and y coordinates of Car^N.
- isTerminal is a boolean variable denoting whether the state is terminal.

Your task is to fill up the function that implements the Value Iteration algorithm which is marked with "FILL ME".

You can test your code with the test configurations given in the script by running `python __init__.py N`, where N is an int value between 0 to 5, on the docker (using the `docker run ...` command).

HINT : You can use Matrix multiplication to optimise the code, but refrain from using the function `np.matmul()`/`np.multiply()`/`np.dot()` as these functions interfere with the evaluation script. Instead you can use function `matmul()` defined in the same script.

Solution:

- Compute the new values in next time step using the Bellman update equation.
- Use the matrix multiplication.

2.2 Programming problem 2: Monte Carlo Tree Search

[20 marks] Unfortunately, we moved to a bigger parking lot and Value Iteration is not feasible as it does not scale well with large state spaces. However, we can use Monte Carlo Tree Search (MCTS) to handle such cases.

We have provided a separate script which solves the problem with MCTS, Your task is to complete the script by filling up 2 code snippets inside the functions marked with "FILL ME".

The functions required to be filled up are:

a) backpropagate() : This function should implement the backpropation step of MCTS.

b) chooseBestActionNode() : Populate the list bestNodes with all children having maximum value. The value of the i^{th} child node can be calculated with the formula:

$$\frac{v_i}{n_i} + c\sqrt{\frac{\log N}{n_i}}$$

where,

- v_i : sum of returns from the i^{th} child node
- n_i : Number of visits of the i^{th} child node
- N : Number of visits of the current node
- c : The exploration constant. We set it to be 1.

For more details on MCTS, it might be helpful to look here. You are also encouraged to look through the rest of the code to see how the entire MCTS algorithm is implemented.

You can test your code with the test configurations given in the script by running `python __init__.py N`, where N is an int value between 0 to 5, on the docker (using the `docker run ...` command).

Solution:

- The *backpropagate* function implements the backpropation step of MCTS, which updates the reward and the visit times for each node in the tree.
- To choose the best node, you need to compute the UCT for each node.
- The base of log should be e.
- If there are more than one nodes with equivalent best UCT values, you need include all of them in the candidate list.