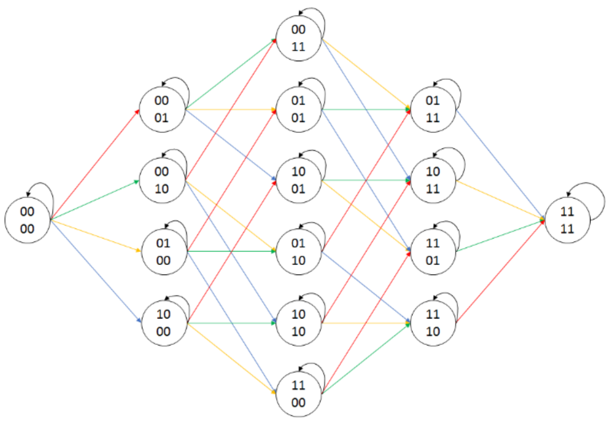
1. Exercise 1

1.1 Markov Decision Processes

a) Formalize the above described problem as a Markov Decision Process. You do not have to write out the individual elements completely (i.e. name all states explicitly or provide a full transition matrix), but define the contents in unambiguous expressions.



State set:

Each state is represented as an encoding of the solved tasks. E.g.:

01
10
means, that the tasks 2 and 3 have been solved, but the tasks 1 and 4 have not been solved yet (0110 = 6)

00
11
means, that the tasks 1 and 2 have been solved, but the tasks 3 and 4 have not been solved yet (0011 = 3)

Action set:

The actions are referenced by colors: $\mathcal{A} = \{a_{Task1}, a_{Task2}, a_{Task3}, a_{Task4}\}$

Transition function:

The state transition is defined by an arrow, colored accordingly to the action, which must be taken to perform the state transition. The color encoding contains the information of the probability to propagate from one state to another: 0.1 for action 1, 0.8 for action 2, 0.3 for action 3 and 0.5 for action 4. For better readability, all self-loops have been colored black which means, that all the outgoing edges to other states can also keep the agent in its current state.

Reward function:

Like the actions the rewards are encoded on the colored edges. The agent is rewarded 4 points for action 1, 1 point for action 2, 3 points for action 3 and 2 points for action 4. Black edges have a reward of 0 points.

b) How would you model the risk of failing the exam in this scenario?

Hint: You can introduce terminal rewards.

A terminal reward of -4 is introduced for all states were the sum of the points is lower than 4.

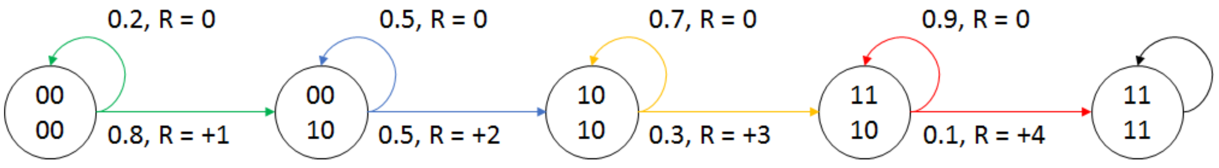
If the final reward is 0 or higher the student passed the exam. If it is lower, the student failed.

All probability of taking a branch where the student ends up in a failed task are then summed up.

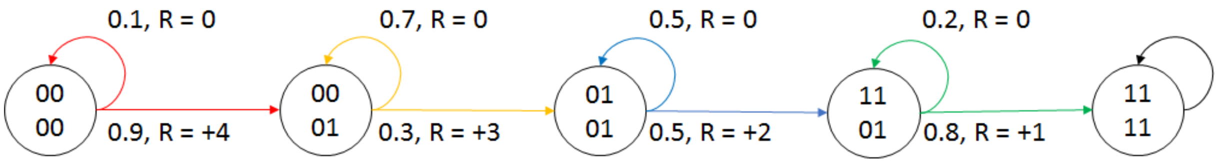
c) Student S considers two possible policies, π_A^S and π_B^S, that determine in which order the tasks will be solved. The first policy follows the tasks in increasing difficulty, i.e. with decreasing possibility of success p_i^S . Using policy π_B^S, the tasks are tackled in reversed order (reversed with respect to π_A^S). Compare both policies by determining the path costs for both policies $\pi_A^S$and π_B^S.

Remember: The student knows about the success of solving a task after dealing with it and may choose to solve a task again in the case of failing.

Markov Chain for π_A^S:



Markov Chain for π_B^S:



$$v_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s]$$

Remark: We put a lot of effort into writing a general solving algorithm for this type of MDP problems (see code). However, after the meeting on Friday we knew, we made things too complicated and we had understanding issues what really is expected from us to program for the finite horizon problem.

d) Derive a stationary policy π_C^S that is more promising than π_A^S and π_B^S. In what order does this strategy choose the tasks?

The stationary policy π_C^S chooses the action with the highest expected direct reward:

$$\pi_C^S(a|s) = \begin{cases} 1 & if\ a = arg \max_{a \in \mathcal{A}} \mathbb{E}_\pi[\mathrm{R}_t | S_t = s, A_t = a] \\ 0 & otherwise \end{cases}$$

The order is 4-3-2-1.

e) Suggest a method that reduces the probability of failing for each of the above policies. You can describe in non-formal sentences. *Hint: Use non-stationary policies.*

For policy π_A^S: When the agent solves task 2 on the third try, it will try to solve task 3 directly because solving task 4 will not help the agent preventing to fail.

For policy π_B^S: When the agent did not solve task 1 at least on the third try, it will try to solve task 2.

- In case of success it will try to solve task 3 to get the needed 4 total points.
- In case of failure it will try to solve task 1 again.

For policy π_C^S: When the agent did not solve task 4 at least on the third try, it will try to solve task 2.

- In case of success it will try to solve task 3 to get the needed 4 total points.
- In case of failure it will try to solve task 1 to get the for points at once.

f) Student $\mathcal{T}$ has learned selectively and did not prepare the topics of tasks 2-4 ($p_i^{\mathcal{T}} = 0.0$ for $i \in \{2,3,4\}$). How good must student $\mathcal{T}$ be prepared for the topic of task 1 (i.e. how high has $p_1^{\mathcal{T}}$ to be at least) in order to be at least as successful as student S, if student S follows policy π_A^S?

Student $\mathcal{T}$ must prepare for task 1 so that the $p_1^{\mathcal{T}} \approx 22.97\%$ to perform as good as student S with π_A^S. This result has been calculated with Excel.

1.2 Bellman Equation

a) Show exemplary for state $s_{3,3}$ in the middle of the grid with $v\left(s_{3,3}=0.7\right)$ that the Bellman equation is satisfied for all neighboring states.

$$v\left(s_{3,3}\right)=\mathcal{R}_{s_{3,3}}+\gamma\sum_{s\prime\in\mathcal{S}}\mathcal{P}_{s_{3,3}s\prime}v(s')$$

$$\mathcal{R}_{s_{3,3}}=0,\qquad \gamma=0.9$$

$$\mathcal{P}_{s_{3,3}s_{2,3}}=\mathcal{P}_{s_{3,3}s_{4,3}}=\mathcal{P}_{s_{3,3}s_{3,2}}=\mathcal{P}_{s_{3,3}s_{3,4}}=0.25$$

$$v\left(s_{2,3}\right)=2.3,\qquad v\left(s_{4,3}\right)=-0.4,\qquad v\left(s_{3,2}\right)=0.7,\qquad v\left(s_{3,4}\right)=0.4,$$

$$v(s)=0+0.9*[0.25*(0.4+0.7+2.3-0.4)]=0.675\ \approx 0.7$$

b) Explain why the value of state *B* is higher than the direct reward. Why does this not hold for state *A*

A state valuation function tries to judge the goodness of the actual state. For that it takes the expectation of the direct reward **and** the discounted expected state valuation of the next possible states into account. This can directly be seen by examining the Bellman equation:

$$v(s)=\mathcal{R}_s+\gamma\sum_{s\prime\in\mathcal{S}}\mathcal{P}_{ss\prime}v(s')$$

Expected direct reward (first part of the Bellman equation $\mathcal{R}_s$):

For both states A and B the actual reward is determined because every action will give the reward for jumping into A' and respectively B'.

$$\mathcal{R}_A=10,\qquad \mathcal{R}_B=5$$

Expected state valuation (second part of the Bellman equation $\gamma\sum_{s\prime\in\mathcal{S}}\mathcal{P}_{ss\prime}v(s')$):

When the agent will be in state A, every action will take it to state A', respectively from B to B'.

$$\mathcal{P}_{AA\prime}=\mathcal{P}_{BB\prime}=1$$

$$v(A)=\gamma\sum_{s\prime\in\mathcal{S}}\mathcal{P}_{As\prime}v(s')=0.9*[1*(-1.3)]=-1.17\approx-1.2$$

$$v(B)=\gamma\sum_{s\prime\in\mathcal{S}}\mathcal{P}_{Bs\prime}v(s')=5+0.9*[1*(0.4)]=0.36\ \approx 0.4$$

The expectation of the next states valuation is positive for B and negative for A.

Both parts combined:

$$v(A)=\mathcal{R}_A+\gamma\sum_{s\prime\in\mathcal{S}}\mathcal{P}_{As\prime}v(s')=10-1.2\approx 8.8$$

$$v(B)=\mathcal{R}_B+\gamma\sum_{s\prime\in\mathcal{S}}\mathcal{P}_{Bs\prime}v(s')=5+0.4=5.4$$

The second part of the equation makes the difference between state valuation and direct reward.

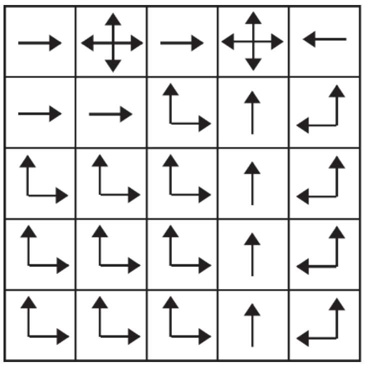
The optimal policy π_* is shown in Figure 2. Now assume that we reduce the reward for jumping from A to A' to 4.

(c) How does the optimal policy change? What are the new values of states *A* and *B*?

Before the reward change the optimal policy for the agent to do was to loop the behavior of striving towards A and being teleported to A'. (Only exception is B, because it doesn't matter what the agent is doing, it will be teleported to B').

After the reward change it becomes more rewarding to loop between B and B' because the distance between B and B' is shorter than the distance between A and A'. Plus, the reward is higher for looping between B and B' than looping between A and A'.

In a graphical way, the optimal policy looks now like this:

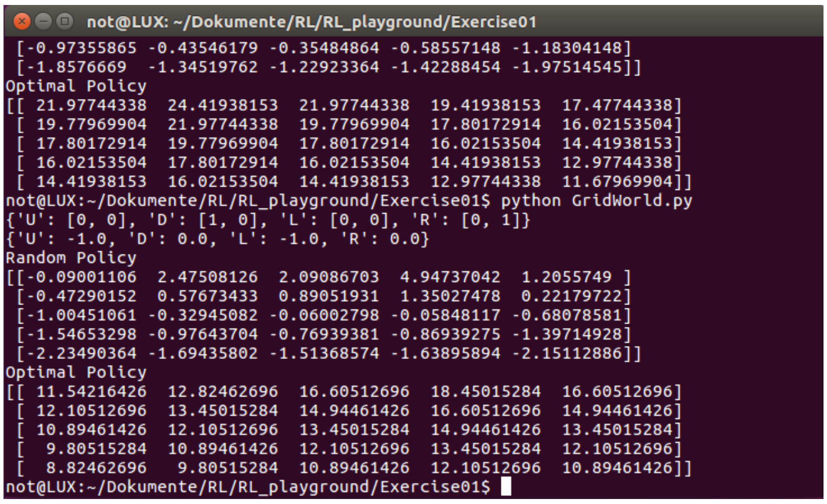


The policy defines the action so that the agent should move towards B with every step. An interesting behavior to point out is that the agent chooses to move from state $s_{1,1}$ to A. The agent might lose some reward for getting teleported to A' and being therefore teleported away from B but the direct reward outweighs this 'distance loss'.

The following state valuations of A and B have been calculated with the program 'GridWorld.py'

$$v_{\pi_*}(A) \approx 12.8$$

$$v_{\pi_*}(B) \approx 18.45$$



```
not@LUX: ~/Dokumente/RL/RL_playground/Exercise01
 [-0.97355865 -0.43546179 -0.35484864 -0.58557148 -1.18304148]
 [-1.8576669  -1.34519762 -1.22923364 -1.42288454 -1.97514545]]
Optimal Policy
[[ 21.97744338  24.41938153  21.97744338  19.41938153  17.47744338]
 [ 19.77969904  21.97744338  19.77969904  17.80172914  16.02153504]
 [ 17.80172914  19.77969904  17.80172914  16.02153504  14.41938153]
 [ 16.02153504  17.80172914  16.02153504  14.41938153  12.97744338]
 [ 14.41938153  16.02153504  14.41938153  12.97744338  11.67969904]]
not@LUX:~/Dokumente/RL/RL_playground/Exercise01$ python GridWorld.py
{'U': [0, 0], 'D': [1, 0], 'L': [0, 0], 'R': [0, 1]}
{'U': -1.0, 'D': 0.0, 'L': -1.0, 'R': 0.0}
Random Policy
[[-0.09001106  2.47508126  2.09086703  4.94737042  1.2055749 ]
 [-0.47290152  0.57673433  0.89051931  1.35027478  0.22179722]
 [-1.00451061 -0.32945082 -0.06002798 -0.05848117 -0.68078581]
 [-1.54653298 -0.97643704 -0.76939381 -0.86939275 -1.39714928]
 [-2.23490364 -1.69435802 -1.51368574 -1.63895894 -2.15112886]]
Optimal Policy
[[ 11.54216426  12.82462696  16.60512696  18.45015284  16.60512696]
 [ 12.10512696  13.45015284  14.94461426  16.60512696  14.94461426]
 [ 10.89461426  12.10512696  13.45015284  14.94461426  13.45015284]
 [  9.80515284  10.89461426  12.10512696  13.45015284  12.10512696]
 [  8.82462696   9.80515284  10.89461426  12.10512696  10.89461426]]
not@LUX:~/Dokumente/RL/RL_playground/Exercise01$ 
```

1.3 Bonus: Experiences

Submit an *experiences.txt*, where you provide a brief summary of your experience with this exercise, the corresponding lecture and the last meeting. As a minimum, say how much time you invested and if you had major problems – and if yes, where.

Nico

	Hours	Experience	Problems
Exercise 1	13+3	Bellman part quite easy. Just use the formula. MDP/MRP task was good for understanding. In general, the exercise sheet conception is excellent.	Problem formulation for the MDP/MRP part was a confusing. Especially Task 1c).
Lecture 2	4	Really had to look up the mathematical definitions sometimes to be absolute clear what David meant.	Posted in the Google Doc
Meeting	1.5	More time should be used on the exercises.	
General		Flipped classroom concept works out nicely. Splitting tasks between teachers benefits a lot. Kahoot is cool, but takes a lot of time. Somehow you are forced to answer with time pressure which is a good exercise, however,	

Hendrik

	Hours	Experience	Problems
Exercise 1	12+8	**Task1:** subtasks a) and b) were good to understand and pretty much straightforward but I did not manage to solve c) and the following subtasks on my own. **Task 2:** good and well-structured task	1c) was very confusing for me with the path cost expression and I did not know, what value is wanted here
Lecture 2	3		
Meeting	1.5		
General			