Planning and Optimization

M. Helmert, G. Röger
C. Büchner, T. Keller, S. Sievers

University of Basel
Fall Semester 2021

Exercise Sheet F

Due: December 16, 2021

Important: for submission, consult the rules at the end of the exercise. Non-adherence to these rules might lead to a penalty in the form of a deduction of marks or, in the worst case, that your submission will not be corrected at all.

The files required for this exercise are in the directory `exercise-f` *of the course repository (*`https://github.com/aibasel-teaching/planopt-hs21`*). All paths are relative to this directory. Update your clone of the repository with* `git pull` *to see the files. In the virtual machine,* `/vagrant/plan-opt-hs21` *is the repository.*

Exercise F.1 (3+1 marks)(Lecture F1)

You are in charge of a large Mediterranean fish farm that produces prawns. Once every catching season, you need to decide what part of the prawn population you will catch and commercialize, and what part you will leave in the farm and allow to reproduce. Assuming that the prawn population in any given season is denoted by x, the reward you get if you commercialize $y \leq x$ prawns is $10y$, whereas the number of prawns on the fish farm if you leave $z \leq x$ prawns to reproduce is a random variable. For the sake of simplicity,[1] we assume that with probability 0.7, the population next season will be $\lceil 1.4z \rceil$ (a standard year), with probability 0.2, it will be $\lceil 1.8z \rceil$ (a great year!), and with probability 0.1 it will be $\lceil 0.9z \rceil$ (a not-so-good year).

(a) Assume you start with a population of 1000 prawns, and you value the reward of one catching season twice as high as the reward for the next one (after all, climate change is likely to turn your fish farm into a tropical wave pool for tourists soon enough). You might also assume that your fish farm has a maximum capacity of N. Formalize your fish farm as a *Markov Decision Process* $\mathcal{M} = \langle S, A, T, R, s_0, \gamma \rangle$, clearly and formally defining each of the elements of the problem: what are the states in your model, what the possible actions, what are the transition probabilities and rewards, what is the initial state, and what is the discount factor.

(b) Discuss the influence of the discount factor in this scenario. To do so, answer the following two questions:

- In general, what changes about your business model if you increase the discount factor?
- How does the *optimal policy* of your fish farm change when you increase the discount factor?

[1] The actual law governing the growth of prawn population is more complex, but results nonetheless in a probability distribution, which is what interests us here.

Exercise F.2 (1+2+4+4+1 marks)(Lectures F2, F3, F4)
Consider the discounted reward MDP $\mathcal{T} = \langle S, A, R, T, s_0, \gamma \rangle$ with

- $S = \{s_0, s_1, s_2, s_3\}$,
- $A = \{a_0, a_1, a_2, a_3\}$,
- $R(s, a_0) = 0$ for all $s \in S$
  $R(s, a_1) = 1$ for all $s \in S$
  $R(s, a_2) = 5$ for all $s \in S$
  $R(s, a_3) = 10$ for all $s \in S$,
- $T(s_0, a_0, s_1) = 1$
  $T(s_1, a_0, s_0) = \frac{1}{3}$ and $T(s_1, a_0, s_2) = \frac{2}{3}$
  $T(s_2, a_0, s_0) = \frac{2}{3}$ and $T(s_2, a_0, s_3) = \frac{1}{3}$
  $T(s_1, a_1, s_0) = \frac{2}{3}$ and $T(s_1, a_1, s_2) = \frac{1}{3}$
  $T(s_2, a_2, s_0) = 1$
  $T(s_3, a_3, s_0) = 1$
  $T(s, a, s') = 0$ for all other $s, s' \in S$ and $a \in A$
- $\gamma = 0.9$.

(a) Provide $\mathcal{T}$ graphically. Make sure that it is clear which numbers represent rewards and which numbers represent transition probabilities, and make sure that it is clear which outcomes belong to which action.

(b) Provide the LP that encodes $\mathcal{T}$ as an input file for SoPlex. Then solve the LP and provide the objective value.

Instructions on how to install and use SoPlex are in the file `soplex-readme.txt`.

(c) Consider the policy π_0 that maps s_i to a_i for all $s_i \in S$ and perform *two iterations* of policy iteration, starting with initial policy π_0. Use iterative policy evaluation in the policy evaluation step and round state-values $\hat{V}_{\pi_0}(s_i)$ and $\hat{V}_{\pi_1}(s_i)$ for $i \in \{1, 2, 3, 4\}$ to two decimals (round the final result for each state-value, not intermediate results). Use $\epsilon = 0.235$ to terminate policy evaluation or after computing $\hat{V}^4_{\pi_i}(s_i)$, whichever comes first. Use $\hat{V}^0_{\pi_0}(s_0) = 10$, $\hat{V}^0_{\pi_0}(s_1) = 11$, $\hat{V}^0_{\pi_0}(s_2) = 13$, and $\hat{V}^0_{\pi_0}(s_3) = 18$ for iterative policy evaluation, and use the final state-values of the first iteration as initial state-values for the second iteration.

Provide all intermediate values of iterative policy evaluation under π_0 and π_1 in the two tables below (insert $\max_{s \in S} |V^i_{\pi_k}(s) - V^{i-1}_{\pi_k}(s)|$ in the last column) as well as the computed policies π_1 and π_2. Does policy iteration terminate after the second iteration or does it perform further iterations?

$\pi_0(s_0) =$ __a_0__ $\pi_0(s_1) =$ __a_1__ $\pi_0(s_2) =$ __a_2__ $\pi_0(s_3) =$ __a_3__

i	$\hat{V}^i_{\pi_0}(s_0)$	$\hat{V}^i_{\pi_0}(s_1)$	$\hat{V}^i_{\pi_0}(s_2)$	$\hat{V}^i_{\pi_0}(s_3)$	max. change
0	10	11	13	18	n/a
1					
2					
3					
4					

$\pi_1(s_0) =$ _____ $\pi_1(s_1) =$ _____ $\pi_1(s_2) =$ _____ $\pi_1(s_3) =$ _____

i	$\hat{V}^i_{\pi_1}(s_0)$	$\hat{V}^i_{\pi_1}(s_1)$	$\hat{V}^i_{\pi_1}(s_2)$	$\hat{V}^i_{\pi_1}(s_3)$	max. change
0					n/a
1					
2					
3					
4					

$\pi_2(s_0) =$ _____ $\pi_2(s_1) =$ _____ $\pi_2(s_2) =$ _____ $\pi_2(s_3) =$ _____

(d) Perform *five iterations* of value iteration with initial values $\hat{V}^0(s_0) = 10$ and $\hat{V}^0(s) = 0$ for $s \in \{s_1, s_2, s_3\}$. Provide the values for all states after each iteration in the table below, and provide the policies that result from the values after the 1st, 2nd, 3rd, 4th and 5th iteration. Would value iteration need to perform further iterations after the 5th iteration?

i	$\hat{V}^i(s_0)$	$\hat{V}^i(s_1)$	$\hat{V}^i(s_2)$	$\hat{V}^i(s_3)$
0	10	0	0	0
1				
2				
3				
4				
5				

$\pi_1(s_0) =$ _____ $\pi_1(s_1) =$ _____ $\pi_1(s_2) =$ _____ $\pi_1(s_3) =$ _____

$\pi_2(s_0) =$ _____ $\pi_2(s_1) =$ _____ $\pi_2(s_2) =$ _____ $\pi_2(s_3) =$ _____

$\pi_3(s_0) =$ _____ $\pi_3(s_1) =$ _____ $\pi_3(s_2) =$ _____ $\pi_3(s_3) =$ _____

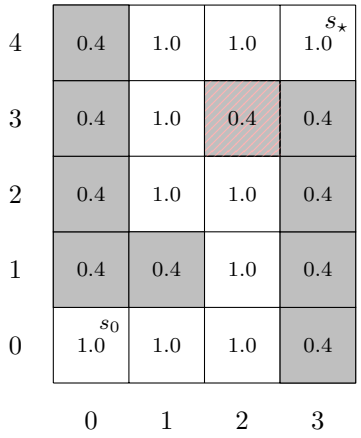
$\pi_4(s_0) =$ _____ $\pi_4(s_1) =$ _____ $\pi_4(s_2) =$ _____ $\pi_4(s_3) =$ _____

$\pi_5(s_0) =$ _____ $\pi_5(s_1) =$ _____ $\pi_5(s_2) =$ _____ $\pi_5(s_3) =$ _____

(e) Assume you run a variant of value iteration that terminates after a fixed number of iterations n. What happens with $\hat{V}_n(s_0)$ when $n \to \infty$? Justify your answer.

Exercise F.3 (4 marks)(Lecture F4)

Recall the SSP example from the lecture where an agent should move from an initial state to the goal state in a grid world. The figure below shows the grid with the initial state s_0 and the goal state $s_\star$. Possible moves are N, E, S, and W for the four directions, but actions are only applicable if they would not lead the agent out of the grid. In the goal state, no actions are applicable. The numbers in the grid indicate the probabilities p of an action to succeed, i.e., to move the agent in the intended direction. With probability $1 - p$, the agents stays where she is. The costs of applying an action is 1 in all cells except in the striped cell (2, 3), where it is 3.



In this exercise, you have to implement a variant of asynchronous value iteration for the grid world problem, which picks a single random state in each iteration that is updated in place and terminates based on a fixed number of iterations. Complete all functions with a TODO in the file `asynchronous-vi/asynchronous-vi.py`. You do not need to modify or submit other files.
You can make yourself familiar with the problem representation by looking at `asynchronous-vi/instance.py`*. Executing that file prints the probabilities and costs of all cells as well as the applicable actions for all states and the resulting successors.*
To test your implementation, you can execute the file `asynchronous-vi/asynchronous-vi.py`*. It takes an optional argument* `--num-iterations` *which specifies for how many iterations to run the algorithm (the default is 300).*
We recommend to implement the functions in the following order, since you should never need to use a function further down in the order in any of the earlier functions.

- `compute_q_value`: compute the Q-value (called action-value in the lecture) for a given state and action and given state-values.
- `compute_greedy_action_and_q_value`: for a given state and given state-values, compute the greedy (=best) applicable action and the resulting Q-value of that action.
- `bellman_update_in_place`: perform an in-place Bellman update of the state-value of a random non-goal state of the SSP.
- `asynchronous_value_iteration`: perform asynchronous value iteration.

Submission rules:

- Exercise sheets must be submitted in groups of three students. Please submit a single copy of the exercises per group (only one member of the group does the submission).
- Create a single PDF file (ending .pdf) for all non-programming exercises. Use a file name that does not contain any spaces or special characters other than the underscore "_". If you want to submit handwritten solutions, include their scans in the single PDF. Make sure it is in a reasonable resolution so that it is readable, but ensure at the same time that the PDF size is not astronomically large. Put the names of all group members on top of the first page. Either use page numbers on all pages or put your names on each page. Make sure your PDF has size A4 (fits the page size if printed on A4).
- For programming exercises, only create those code textfiles required by the exercise. Put your names in a comment on top of each file. Make sure your code compiles and test it. Code that does not compile or which we cannot successfully execute will not be graded.
- For the submission: if the exercise sheet does not include programming exercises, simply upload the single PDF. If the exercise sheet includes programming exercises, upload a ZIP file (ending .zip, .tar.gz or .tgz; *not* .rar or anything else) containing the single PDF and the code textfile(s) and nothing else. Do not use directories within the ZIP, i.e., zip the files directly.
- Do not upload several versions to ADAM, i.e., if you need to resubmit, use the same file name again so that the previous submission is overwritten.