Resit 2: Reinforcement learning and hyper-heuristics

Deadline: Jan 26 at 12h00 (late submissions are not tolerated)
Questions: via email (count two working days for an answer)

Guidelines

- Grading is transparent and is **entirely based on the marking scheme** available on Canvas.
- The assignment measures the same skills as the ones that were developed during the lectures and the workshops. Therefore, **you can use the lecture and workshop elements to get some inspiration** for your code, your graphs, and your answers.
- There is **no hard limit as far as the report length is concerned**. For indicative purposes, most reports will have between 7 and 13 pages (including graphs and tables), with an average of 10 pages.
- For assessing your work, we are **re-running your code**, or parts of your code at least. As a result:
  - Make sure that **your code is running** – the marking scheme is clear, it is 0 for the coding part if the code is not running;
  - Make sure that **your code is in its final version**: there should not be any major difference between the report and the Python implementation — major means, for example, a different reward system, not a different parameter value;
  - **We do not need your code to print every single information from the report**. It is okay to have some parts that are commented (with an indication within the code of what a commented part does if it is uncommented);
- If you realize in the middle of the report that there was a better representation for the problem and you do not have the time to start over, **stick with your first representation**. At the end of the report, you can mention your other (maybe better) ideas. **Do not mix representations and models in the middle of the report** as it makes things confusing.
- Make sure you **manage your time well**:
  - **Do not start the assignment at the last minute** since your agents will need some time to train;
  - Check the marking scheme and **focus first on the tasks that are given**;
  - If – and only if – you have some extra time and some extra interest in the topic, you can go the extra mile with advanced heuristic testing, evaluation of more than one agents' performance, fancy graphs, and other extra answers ... but be aware that **the extra effort might not be worth the reward**;
  - If you lack of time to finish the assignment, decrease the size of your training dataset. But be aware that this will also damage the accuracy and the performance of your agent;

Good luck with the assignment!

Exercise 1 [5 points]

We study a version of the game Yahtzee in which one throws a set of dices in order to create certain combinations. In that version of the game named "Baby-Yahtzee", the player throws three dices and receives a score equals to the face value of the three dices plus an additional reward for certain combinations:

- A pair (two identical numbers) brings 20 additional points
- A triple (three identical numbers) brings 60 additional points
- A straight (three consecutive numbers) brings 30 additional points

The player may pay a certain penalty in order to re-throw the dices:

- Re-throwing one dice costs 4 points
- Re-throwing two dices costs 7 points
- Re-throwing all three dices costs 6 points

The score of the game is determined once the player decides to stop re-throwing the dices. For example:

- If after throwing the three dices, the player obtains $[6, 3, 3]$ and decides to stop the game, they get $6 + 3 + 3 + 20 = 32$ points for the game.
- If after throwing the three dices, the player obtains $[6, 3, 3]$, decides to re-throw the 6, obtains $[3, 3, 3]$, and decides to stop the game afterwards, they get $-4 + 3 + 3 + 3 + 60 = 65$ points for the game.
- If after throwing the three dices, the player obtains $[6, 3, 3]$, decides to re-throw the 6, obtains $[5, 3, 3]$, decides to re-throw the two 3s, obtains $[5, 4, 6]$ and decides to stop the game afterwards, they get $-4 - 7 + 5 + 4 + 6 + 30 = 34$ points for the game.

Note that it is possible for the player to end the game with a negative number of points. The objective of the player is to maximize the number of points received for one game.

1. [0.75] Describe and implement some greedy strategies for the problem.
2. [0.75] Identify the initial, intermediary, and end states of a model for the problem suitable for reinforcement learning, together with the available actions and their rewards.
3. [2.5] Implement a Q-learning algorithm in Python to solve that problem. Explore the impact of several sets of parameters, outline the parameters you believe are the best, and motivate your choices.
4. [0.5] Compare the results obtained by the trained agent and by the greedy strategies found in question 1.
5. [0.5] Outline the strategy found by the trained agent and motivate its relevance.

Exercise 2 [5 points]

Note: In this exercise, training your hyper-heuristic will take around 5 seconds per episode. Take that information into account while designing your Q-learning framework!

We study the skiving stock problem that can be defined as follows: given a set of n items with weight w_i $(i = 1, \ldots, n)$ and a threshold c, the Skiving Stock Problem (SSP) consists in creating the maximum number of disjoint subsets $S_1, S_2, \ldots, S_{OPT}$ such that the sum of the weights in each subset is above the threshold c. A greedy heuristic for the SSP works as follows:

- Store the unassigned items in U and set $k = 1$
- While $|U| > 0$
  - Set $S_k = \emptyset$
  - While $\sum_{i \in S_k} w_i + \max_{i' \in U}\{w'_i\} < c$
    - Determine the index i' of the largest unassigned item
    - $S_k \rightarrow S_i \cup \{i'\}$
    - $U \rightarrow U \setminus \{i'\}$
  - Determine the index i'' of the smallest unassigned item such that $w_{i''} + \sum_{i \in S_k} w_i \geq c$
  - $S_k \rightarrow S_k \cup \{i''\}$
  - $U \rightarrow U \setminus \{i''\}$
  - $k = i + 1$

Informally, the heuristic aims at iteratively building a new subset using the largest unassigned items until adding that largest unassigned item would bring the weight of the new subset above the threshold c. When that is the case, instead of adding that particular item, the heuristic adds the smallest unassigned item that would bring the weight of the new subset above the threshold c. That heuristic usually provides good results, but this is not always the case. In the opposite, SSP instances can be solved to optimality by solving the following mathematical model with an ILP solver:

$$
\begin{aligned}
\max \quad & \sum_{k=1}^{K} z_k & & & \\
\text{s.t.} \quad & \sum_{i=1}^{n} w_i x_{ik} \geq c \cdot z_k & k = 1, \ldots, K, & & (1)\\
& \sum_{k=1}^{K} x_{ij} \leq 1 & i = 1, \ldots, n, & & (2)\\
& z_k \leq z_{k-1} & k = 2, \ldots, K, & & (3)\\
& x_{ik} \in \{0, 1\} & i = 1, \ldots, n, & k = 1, \ldots, K, & (4)\\
& z_k \in \{0, 1\} & k = 1, \ldots, K, & & (5)
\end{aligned}
$$

where K is a valid upper bound on the optimal solution value for the solved instance (note that $K = n$ is a valid upper bound). However, the time required can be very long. If a time limit is imposed, such a method may still return a feasible solution, but its quality is not necessarily good. Our goal is to create a hybrid approach that will combine the greedy heuristic and the use of an ILP solver to obtain solutions that are better than both the ones obtained by the greedy heuristic and the ones obtained by an ILP solver with a time limit.

1. [0.5] Implement the greedy heuristic and propose an (easy) instance for which the solution it obtains is not optimal.

2. [0.5] Implement the ILP formulation for solving the SSP and propose an instance for which the solution obtained by Gurobi after 5 seconds of computing time is not optimal.

3. [1.5] One way to combine the greedy heuristic and the ILP solver is the following:

   - Store the unassigned items in U and set $k = 1$
   - While a certain stopping criterion is not met
     - Set $S_k = \emptyset$
     - While $\sum_{i \in S_k} w_i + \max_{i' \in U}\{w'_i\} < c$
       - Determine the index i' of the largest unassigned item
       - $S_k \rightarrow S_i \cup \{i'\}$
       - $U \rightarrow U \setminus \{i'\}$
     - Determine the index i'' of the smallest unassigned item such that $w_{i''} + \sum_{i \in S_k} w_i \geq c$
     - $S_k \rightarrow S_k \cup \{i''\}$
     - $U \rightarrow U \setminus \{i''\}$
     - $k = i + 1$
   - Use the ILP solver for 5 seconds of computing time on the reduced SSP instance (i.e., the instance in which the items already assigned to a subset are removed or pre-assigned to a specific subset)

   Describe a Q-learning framework with the initial, intermediary, and end states together with the available actions and their rewards. Attention should be paid to (i) the rewards, (ii) the stopping criterion, and (iii) the generalizability of the agent.

4. [0.5] Create a SSP instance generator. You are free to use the parameters of your choice, but keep in mind that the generated instances should be diversified and allow you to show the competitive advantage of your hybrid approach over the greedy heuristic and the ILP solver with a 5-second time limit. You should attach the instance generator file to your assignment, **not** the (probably many) generated instances. To help you tune your instance generator, the following information is disclosed: an instance where $w_i \in [1, 180]$ (uniformly distributed and integers), $n = 200$ and $c = 200$ was solved with the three approaches and

   - the greedy heuristic found a solution with value 87
   - the hybrid approach in which the ILP solver was called for 5 seconds after 60 subsets were formed by the greedy heuristic found a solution with value 88
   - the ILP solver found a solution with value 81 after 5 seconds

5. [1.0] Implement the Q-learning algorithm you described previously to create the hybrid approach and train it. Motivate your choice regarding the set of parameters used. You do not need to test other sets of parameters.

6. [0.5] Compare the results obtained by the hybrid algorithm, the greedy heuristic, and the ILP solver with a 5-second time limit.

7. [0.5] Outline the strategy found by the trained agent and motivate its relevance.