Block 5: RL-xPPU

(Python and) Reinforcement Learning

Subtask 1:

Scenario:

(Proceed with 1 WP) the active PPU parts consists of a stack (no.1) a conveyor (no.4) a stamp (no.3) and a crane (no.2).

Black WPs are separated at the stack and subsequently transported directly to the conveyor. In contrast, white plastic work pieces and metallic work pieces are transported from the stack to the stamp, processed there and then transported to the conveyor.

Task:

You are provided the files

- *actionspace_Template.JSON*: Open the file with a text editor. The file lists all possible high-level functions of the xPPU. Functions can be activated by changing *false* to *true*. All activated functions can be activated by running the *step* function: *environment.step((action_idx,))*
  action_idx can be retrieved from the order of the activated actions in *actionspace.JSON*
- *statespace_Template.JSON*: This file lists all possible state variables. They are divided into system and workpiece variables. System variables only exist once whereas for each workpiece (Subtask 1: only one WP) in the system another block of workpiece variables is added to the observation array.
- *Subtask1_goal_Template.py*: This file contains the goal function for this subtask. Here the computation of the reward takes place. Complete the given functions inside. Further instructions are given in the file.
  Hint 1: the implementation of the xPPU simulation requires an array containing at least 3 integers for describing whether the goal was reached or not. Don't get confused here. Just use the expressions as given in the file.
  Hint 2: WP reached the conveyor? You will probably need the coordinates of the WP at the target position. Run the simulation manually to get them.
- *Subtask1_DeepQLearning_Template.py*: The code is an adaption from the LunarLander code from Block4. For most functions copy+paste and some minor changes will do the job.
  Hint: Take care of naming the actionspace, statespace and goal file as required in this file.

Subtask 2:

Scenario:

(Proceed with 3 WPs) This subtask executes subtask 1 and afterwards sorts the WP. There are three ramps are used as output storages. Two of the ramps are filled using respective pushing cylinders (pusher) which are internal parts with in conveyor. Ramp 3 (no.5) mounted to the end of the conveyor is filled by translationally moving WP using the conveyor. The ramps shall be filled one by one – starting with the one at the end of the conveyor (ramp 3), then the mid-ramp (ramp 2 no.6)) and finally the ramp at the beginning of the conveyor(ramp 1 no.7)). Each ramp only allow one WP to be stored.

Task:

You are provided a new template for the DQL-code. The other templates can be used for this task as well. Try to find a suitable reward structure for the goal function. Think about small positive rewards if a WP was stored in the right way although the task is not completed yet.

As you can see in the DQL template the DQL algorithm has to execute the Subtask 1. For training purposes you can hardcode Subtask 1 and then call this function. Be careful when calling functions, since there are functions which have to be activated for running Subtask1, but which are not necessary for Subtask2. To simplify and speed up the training the DQL agent should only be able to execute the hardcoded function which runs Subtask1 and all other functions that are necessary for Subtask2, but not the functions which are only necessary for Subtask1.

The simulation code has no function to check if the execution of an action leads to the desired output. E.g.: Assume a WP is placed at Pos2 on the conveyor. If lsc_moveWPForwardToPos1 is executed the conveyor will start to move until a WP reached Pos1 or a timer expires. This will move the WP to the last ramp since it was placed beyond Pos1. In this subtask you are allowed to check, whether an action is suitable for the current workpiece position. Maybe you can succeed without using this trick, but the training process may take long time.

Subtask 3:

Scenario:

(Proceed with 1 WP) This subtask executes subtask 1 and afterwards sorts the WP which was provided on the conveyor to ramp 1 (white or metallic WP) or to ramp 2 (black WP).

Task:

You are provided a new template for the DQL-code. The other templates can be used for this task as well. Try to find a suitable reward structure for the goal function.