RL homework

A.Y. 2023/2024

Nicola Rossi - 289927

Contents

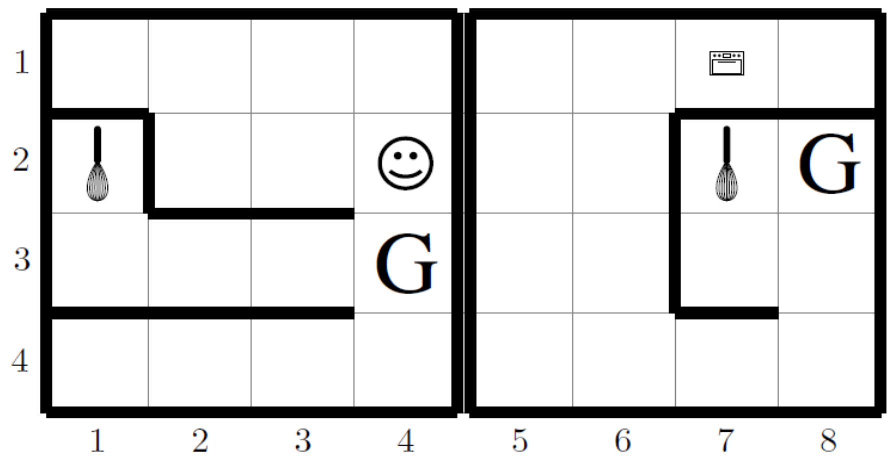
1 Introduction

The objective of this report is to describe the RL [DT0171] project. In particular:

1. briefly describe state representation and dynamics of the system (transition tensor)
2. describe the rewards and the choice of gamma (for optimal policy)
3. show the optimal policy obtained by the Policy Improvement algorithm

2 Environment

I want to start this section by introducing a slightly modified version of the environment (a matter of code convenience and matrix-like indexing):



As discussed in the written test, states can be modeled as triples

$$(i, j, tool)$$

where $i \in \{1, 2, 3, 4\}$ represents the row-position and $j \in \{1, 2, 3, 4, 5, 6, 7, 8\}$ the column-position of the agent, while $tool \in \{0, 1\}$ is a flag set to 1 if the agent has collected **one** tool; 0 otherwise. So, $|S| = (4 \cdot 8) \cdot 2 = 64$.

It's assumed that whenever the agent reaches a cell containing a tool ($(2, 1)$ or $(2, 7)$), it will 'automatically' collect the tool, without any other action needed. This means also that if the agent spawns in $(2, 1)$ or $(2, 7)$, on spawning, it has already collected the tool. So, states $(2, 1, false)$ and $(2, 7, false)$ are practically non-existent (in the tabular representation of the dynamics, their entries are grey-colored to stress this consideration and their columns/rows are empty (all zeros)).

The agent has one objective: reach the oven in $(1, 7)$. But, in order to do so, it must have previously collected one tool. It's therefore natural to partition the set of states in two subsets:

$$S_{tool} = \{(i, j, tool) \mid tool = true\}$$
$$S_{\bar{tool}} = \{(i, j, tool) \mid tool = false\}$$

Indeed, given a position (i, j) the action needed in the case $tool = true$ is, in general, different from the action needed in the case $tool = false$.

On the other hand, there are at most 4 possible actions in each position (actions are not constrained by the value of $tool$): $up, down, left, right$. Since the environment is deterministic, a couple (s, a) exactly determines the next state s'. This implies, with respect to the system's dynamics that, given a couple (s, a), in the transition matrix P_a associated with action a, each row will contain a single 1, since $p(s'|s, a) = 1$ for some $s' \in S$ and $p(s''|s, a) = 0\ \forall s'' \neq s'$.

Finally, a note on the gates G in $(3, 4)$ and $(2, 8)$: it was chosen, as action to cross the gates, to go $right$ both in $(3, 4)$ and in $(2, 8)$. So, if the agent moves to the right in $(3, 4)$ it will reach $(2, 8)$ and vice versa. Another way could have been to add a specific action to cross the gates, but, practically, this would not change the optimal policy. There is only one thing to point out: this choice could be considered not formally correct, since $p((3, 4, tool)|(3, 3, tool), right) = p((3, 4, tool)|(2, 8, tool), right) = 1$, so in matrix P_{right}, both the columns associated with states $(3, 4, true)$ and $(3, 4, false)$ contains 2 ones. The same happens for $(2, 8)$, since it can be reached, going $right$, by $(3, 4)$ and by $(2, 7)$.

3 Reward function

As discussed in the written test, since it's required that the agent does not 'waste time' in collecting the tool and reaching the oven, it's possible to define R as

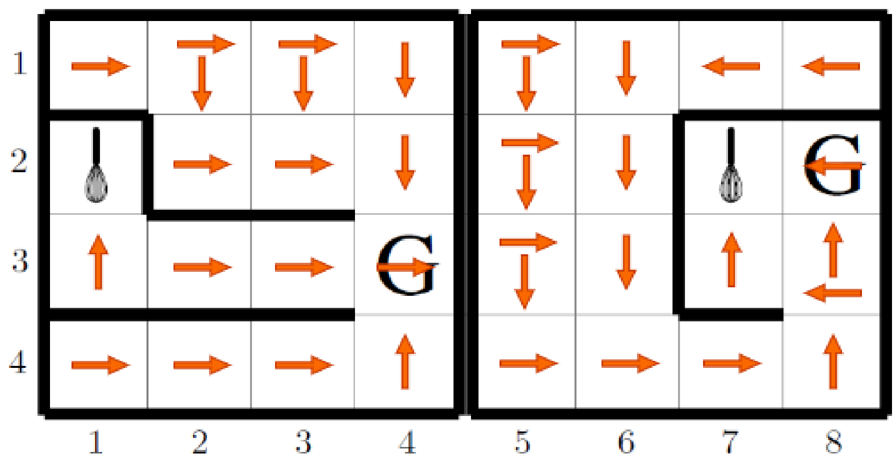
1. $R(s,a,s') = -1 \; \forall \; s, s' \in S_{\bar{tool}}$
2. $R(s,a,s') = 0$ if $s' = (2,1,true)$ or $s' = (2,7,true)$
3. $R(s,a,s') = -1$ if $s \in S_{tool}$ and $s' \neq (1,7,true)$
4. $R(s,a,s') = 0$ if $s \in S_{tool}$ and $s' = (1,7,true)$

Notice that:
1. penalizes every step made without having collected the tool, so the agent will be encouraged to reach a cell containing a tool in the least number of steps.
2. is the reward for a step that makes the agent collect a tool.
3. once a tool is collected, every step made without reaching the oven is penalized. As before, this encourages the agent to choice the shortest tool-oven path.
Finally, 4. is the reward obtained when (having collected the tool) the agent reaches the oven (and achieves its objective).

In this setting we don't have clear guarantees about episodes' length. Although very unlikely, it can happen that (especially in the first iterations of the policy improvement algorithm) the agent get stuck in a loop. In general, an episode, can be of arbitrarily length. A trivial upper bound (at least, for this environment's configuration) on the episode length is 32, since even the longest shortest-path from the initial position of the agent to a cell where a tool is stored and then to the oven cannot be longer than the total 'surface' of the grid, which is equal to 32 (in other words, if an episode's length is greater than 32, we can assume the agent will probably reach the goal in the future, but surely, with more steps than necessary, even for the longest shortest-path). In order to let the agent explore more in the first iterations of the policy improvement algorithm, a maximum length of 1000 was chosen. At this point, being certain that episodes' length is finite, $\gamma = 1$.

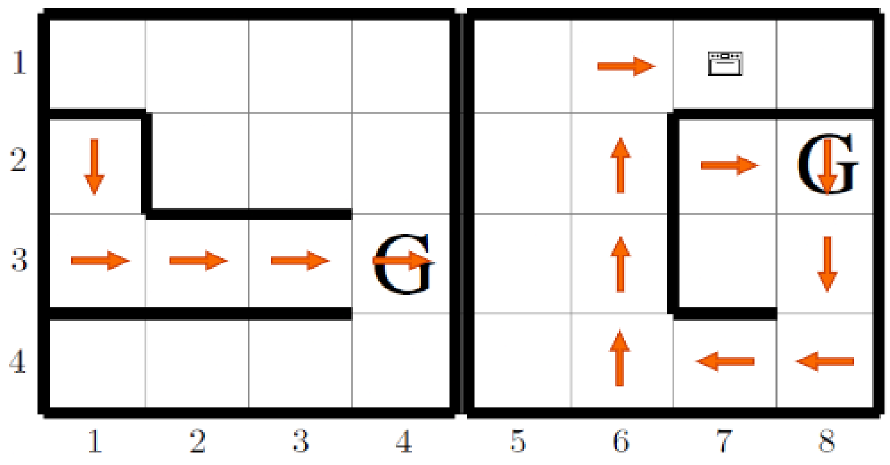
4 Results

First, the computed policy for states in $S_{\bar{tool}}$



A couple of arrows in a cell means the actions are equivalent (in terms of optimality). These equivalent actions were obtained executing the policy improvement algorithm a few times. It's reasonable to assume both actions, in these cases, have probability to be chosen equal to $\frac{1}{2}$.

Then, the computed policy for states in S_{tool}



Actions for blank cells are not showed since they are meaningless.