Project 2023 for the cours on MDP and RL: Solving Pickomino

Nicolas Gast　　Bruno Gaujal

October 13, 2023

The goal of the project is to write a program that plays as best as possible to a two-player version of the board game *Pickomino*. There are two parts:

- Part 1: exact solving of the one-turn version of the game.
- Part 2: two player version and learning.

Work to be done

Project can be done by group of *maximum* 2 *persons*. The grading will be based on two things:

1. A report of your work (in PDF form). Commented code like jupyter-notebook will be appreciated. **Deadline: Thursday, Nov 9, 2023**(midnight).
2. An oral exam on Tuesday, Nov 14, 2023.

In the report and the oral exam, you are asked to provide details about the algorithms that you implemented; what works, what does not work; and what difficulties you encountered. Please also provide simulation results illustrating the performance of your algorithms.

Rules of the game

The game is composed of 8 six-face dices and sixteen tiles (numbered from 21 to 36). The rules are detailed on `https://cdn.1j1ju.com/medias/85/5e/7c-pickomino-rulebook.pdf`. They will be explained during the course.

1 Part I: MDP

We consider a player who plays one turn and who wants to maximize their expected revenue for this turn. The revenue is modeled as follows:

- There is a reward vector $r_{21} \dots r_{36}$ (if your dices sum to i, and contain at least a worm, you get a score r_i).
- If your turn is over, you loose c (This can happen is your total is less than 21, or if you do not have a worm or if you cannot pick any values).

We do not assume that the reward vector r is increasing.

You are asked to:

- Write a program that computes the optimal strategy.
- Illustrate the optimal solution by showing the decision that it takes in a few cases. For instance, if your initial throw is $(1, 3, 3, 3, 3, 4, 4, 5, W)$, $r = (1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 3, 4, 4, 4, 4)$ and $c = 0$: which dices should I pick? Is the strategy the same if $c = -3$?
- Use to model to compute:
  - What is the (maximal[1]) probability of obtaining a Pickomino with i or more pickominos (for all $i \in \{1, 2, 3, 4\}$).
  - What is the maximal probability of obtaining exactly Tile 24? To obtain a tile 27 or higher?

If you program is not fast enough to compute the optimal policy, you can reduce the number of dices and work with a smaller model (*i.e.*, use 4 dices and tiles numbered from 11 to 18).

2 Part II: learning

We consider an adversary, parametrized by two parameters $\alpha, \beta \in [0, 2]$, that you do not know. This adversary implements the one-turn solution that you solved in part I with key modifications:

- The cost c is equal to α times the number of worms on the top tile that this player has (or 0 if the adversary has no tile).
- The reward vector r is such that $r_i = -c$ if obtaining i does not provide any reward, $r_i = 2\beta W$ if obtaining exactly i allows the adversary to steal a tile with W worms from you, and $r_i = W$ if obtaining i allows it to obtain a tile with W worms.

You are asked to:

- Assume that the adversary plays $(\alpha, \beta) = (1, 1)$. Compute the best parameters (α, β) that you can choose to beat this policy (indication: use stochastic gradient descent or a bandit algorithm with discrete parameter values).
- Assume that the parameters (α, β) of the algorithm is unknown. Implement an algorithm that maximizes the probability of winning against such an adversary (you can play as many games as possible against this adversary and its parameters remain fixed. You can use policy gradient, deep Q-learning, contextual bandits...).
- Test your algorithm against parameters like $(\alpha, \beta) = (0, 0)$ or $(1, 1)$. Can you bit the bot? (If relevant) Plot the performance of your algorithm as a function of the number of games played.

As before, if your program is not fast enough to compute the optimal policy, you are allowed to reduce the number of dices (*i.e.*, use 4 dices and tiles numbered from 11 to 18), or even fewer dices.

[1] *Maximal* means using a strategy that tries to maximize that