Report LINFO2275: Assignment 1

Group N°4

Student1: DATI - Nicolas Jadoul - 21641800

Student2: Erasmus - Inês Ferreira - 20552101

Student3: INFO - Sophie Otlet - 12081700

April 3, 2022

---

1 Introduction

This project had the purpose to use the value iteration algorithm of Markov Decision Processes to solve the game "Snakes and Ladders". A Markov Decision Process is a mathematical process that is commonly used to solve decision-making problems when the possible outcomes are partly random and controllable.

In this case we have a board with 15 squares as shown in figure 1 and we want to reach the final square (15) with the least amount of turns possible. We therefore have to choose the best dice for every square. There are 3 different dices that can be used in the various turns: the "security", the "normal" and the "risky". When playing with the "security" one it is ignored the presence of traps and bonus on the board, with the "normal" there is 50 % of probability of triggering the trap/bonus square and finally, with the "risky" dice it is triggered any trap/bonus square. The output of the program should indicate the expected cost associated to the 14 squares of the board as well as the choice of the dice to be used in each of them.

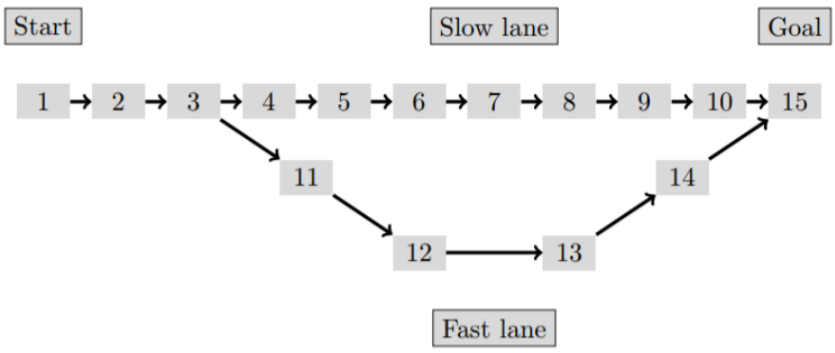


Figure 1: Schematic representation of the game.

In this report is first presented the theoretical methods used to determine the optimal strategy, followed by a brief explanation of the implementation of these methods. Afterwards, it will be compared the theoretical and experimental results for different possible layouts.

2 Theoretical methods

The model we built is based on the Bellman's recurrence equations of a Markov decision process. This is normally used to define the "value" of a certain decision in terms of the payoff based on some initial

choices. In our case, the states are the different squares of the game, the possible actions on a square are the different dices and the cost is the number of turns to reach the goal. The following equations are therefore obtained:

$$\begin{cases} V^*(k) & = min\{\sum_{k'=1}^{15} p(k'|k,a) \cdot (c(a,k'|k) + V^*(k'))\} \\ V^*(15) & = 0 \end{cases},$$

where $V^*(k)$ is the optimal expected cost on square k, $c(a,k'|k)$ is the cost of doing action a and reaching square k' from square k and $p(k'|k,a)$ is the probability to reach square k' from square k by doing action a.

3 Implementation

Our Markov Decision implementation is divided into three main parts.

Firstly, we compute the transition matrix (dimensions: $3 * 15 * 15$, respectively dices (actions), number of cells and number of cells). An element of this matrix is obtained by computing, for every cell relations and dice, the probability to go from one cell to another with that dice. The probabilities are obtained with case by case scenario depending on the cells and the dice.

Secondly, we compute the expected costs recursively, using the value iteration process using the equation in section 2, until a sufficient accuracy is reached (10^{-5}). The expected costs all start at zero and are updated at each iteration. In the equation from section 2, $p(k'|k,a)$ is an element of the transition matrix at index (a,k,k'), $c(a,k'|k)$ has to be computed[1] according to the dice and the two cells with case by case scenario and $V^*(k')$ is the current expected cost from the last iteration (or 0 at initialisation).

Finally, we get the optimal dice choices using the previously obtained expected costs by doing one final iteration and choosing the dice that minimises those costs.

4 Experimentation

To validate our model, we used different interesting layouts, without the circling property :

- Normal : every square is set to 0 (ordinary square);
- Restart : every square, except the start and the goal, is a restart cell (1);
- Penalty : every square, except the start and the goal, is a penalty cell (2);
- Prison : every square, except the start and the goal, is a prison cell (3);
- Bonus : every square, except the start and the goal, is a bonus cell (4);
- Handmade : An arbitrary board which favours the fast lane (every square on the slow lane is a restart (1) and a bonus (4) on the fast lane) ;
- Random : A randomly generated board;

[1] We could have computed the costs alongside the transition matrix to avoid re-computations but the cost function has a $\mathcal{O}(1)$ time complexity and therefore does not slow down the overall computation.

4.1 Strategy accuracy

The first criterion to validate our model is the accuracy of the expected number of turns computed by the Markov value iteration. We therefore computed the Markov Decision Process on the different layouts and applied the obtained strategy 10 000 times to be able to compare the expected optimal cost to the empirical average cost. Our results are presented on Figure 2.

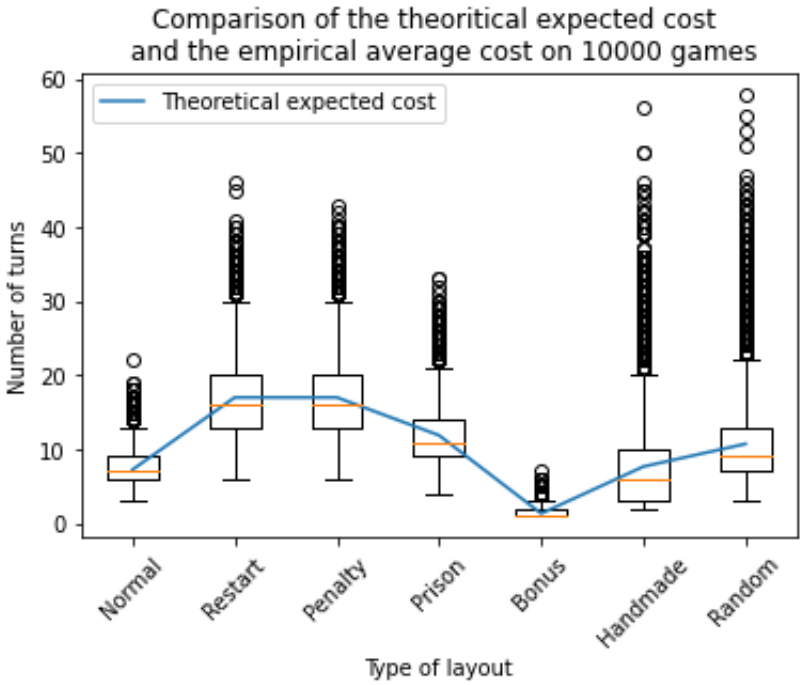


Figure 2

Those results are consistent with the initial expectations. In this graph, we observe that the theoretical expected cost and the empirical average cost during 10 0000 games are really close to each other. It may seems like a lot of outliers are produced but it actually is lower than 5% for every strategy, the worst case being the random strategy.

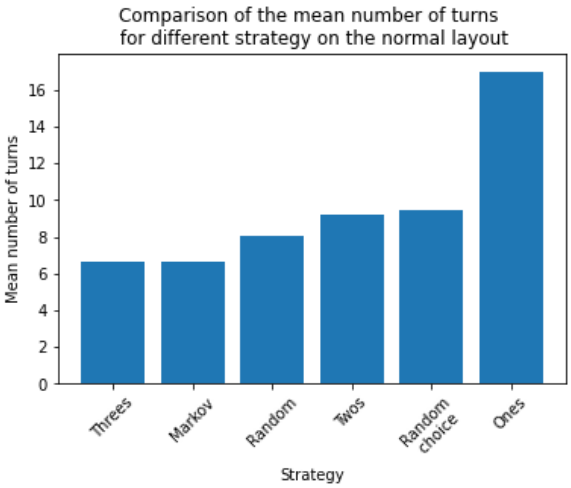
As expected, the cost is lower for the Bonus case (where it is possible to play again in all squares without waiting for the next turn), comparing to all the other possible boards. It is interesting to note that for the "Restart" and "Penalty" cases the expected and empirical costs are exactly the same, there is no difference between having a board with all "restart" squares or "penalty" squares. For the "handmade" and "random" cases we can observe that the expected value is between the highest ("restart" and "penalty" cases) and the lowest ("bonus" case) values. This will always happen for all the possible boards generated as they are a combination of those layouts. Having this in mind, we can say that the empirical average or expected cost will be always less than 20 and higher than 1 with the optimal strategy.

In this graph, it is possible to observe that the theoretical expected cost is always equal or higher than the empirical average cost for all the 10 000 games. Note that the difference between the two results obtained in every case is not substantial.

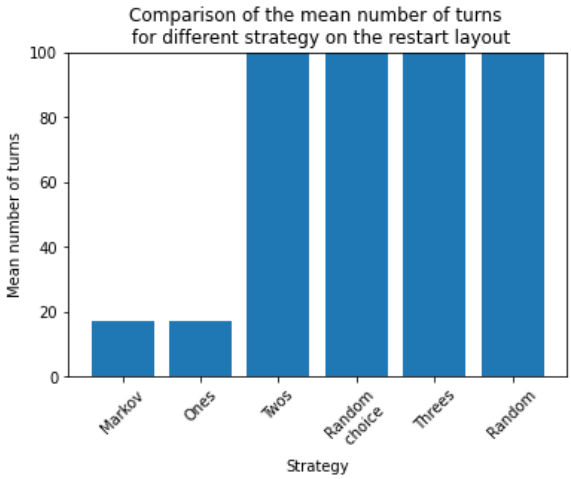
4.2 Strategy comparison

Secondly, we analysed the results using different strategies. It is relevant to compare the Markov strategy with all the other possibilities (for example, using only dice 1, dice 2 or dice 3) and also using a random and a random mixed strategy to ensure that the computed strategy is indeed the optimal one.

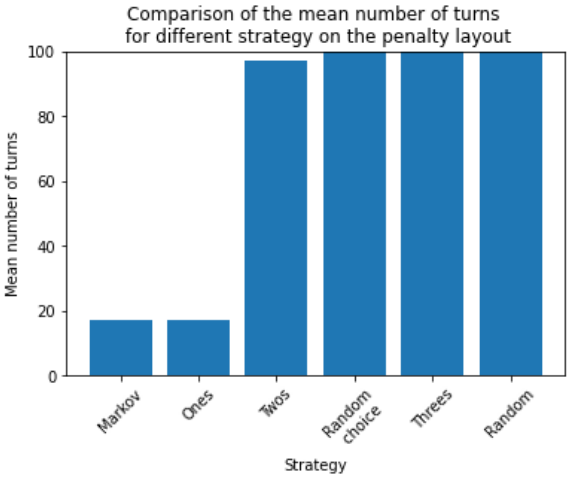
The random mixed strategy is computed by choosing a random dice for every square, but landing two times on the same square leads to using two times the same dice. On the contrary, the random choice strategy choose a random dice at every turn, no matter the square it is on. To be able to see more easily wich startegy was the optimal one on the graphs, they were sorted by mean number of turns and limited to a 100 turns. The resulting graphs are shown in figure 3.



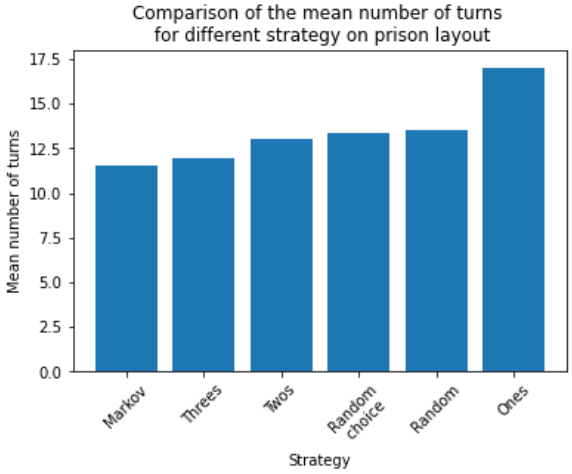
(a) Normal layout



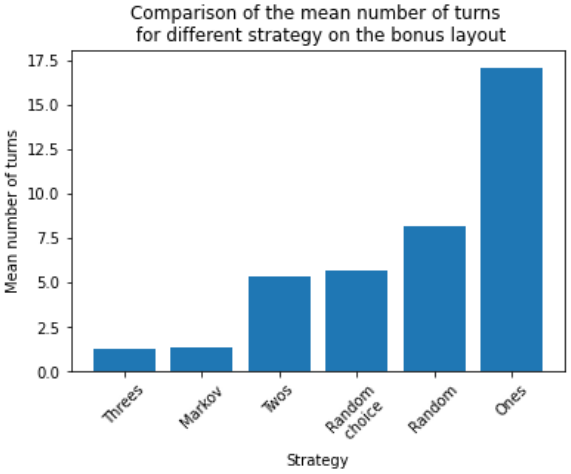
(b) Restart layout



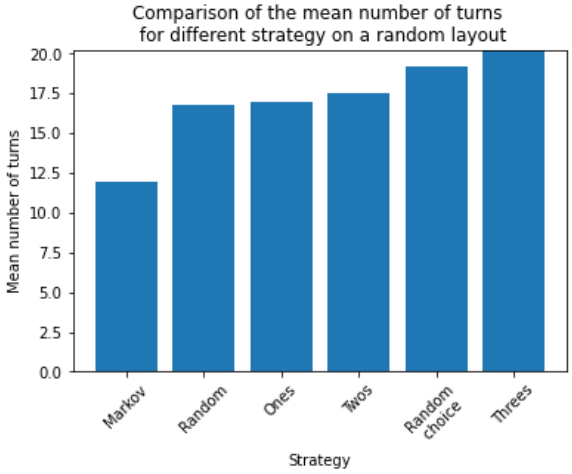
(c) Penalty layout



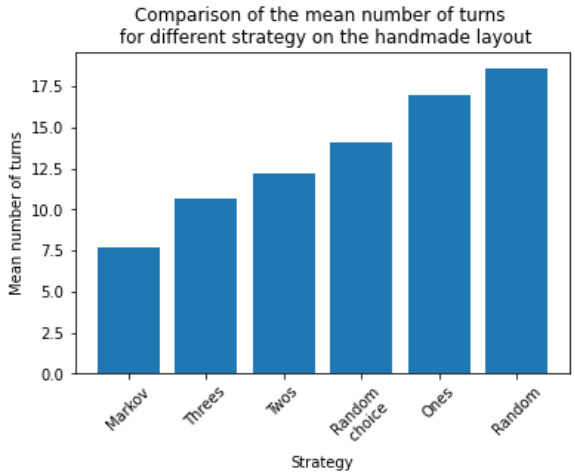
(d) Prison layout



(e) Bonus layout



(f) Random layout



(g) Handmade layout

Figure 3: Comparison of the empirical average cost for different strategies on different layouts

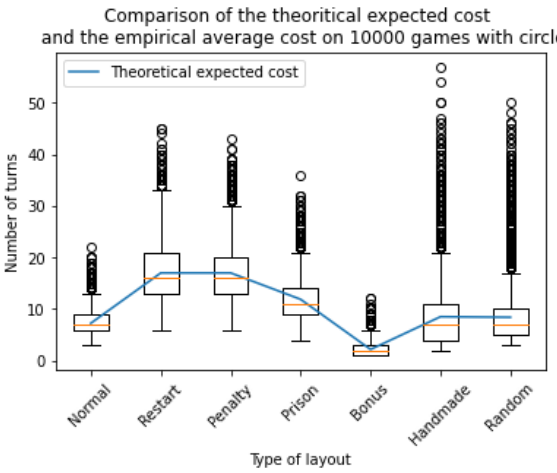
We can observe that for every layout, the strategy computed by Markov's value iteration algorithm is indeed the optimal one. It sometimes is equivalent to another tested strategy, as for the normal and bonus layout (figures 3a and 3e where Markov's strategy is to always choose the third dice. Indeed, as there is no trap to trigger but only advantages, the third dice allows to go further faster. We also see that, when on a layout with only restart (figure 3b) or penalty (figure 3c), the computed strategy is to always use the first dice and using another strategy needs a lot more turns to reach the goal, sometimes an infinity of turns as we would always be brought back to the start when using the dice 3.

A result that may seem surprising is that the computed strategy for a layout with only prison cells (figure 3d) is closer to a strategy using only dice 3 than the other dices. This is explained by the fact that the prison cell just add a turn and that this delay is compensated by the fact that the third dice allows to move further at once. The obtained gain is not substantial, but it proves that the Markov strategy indeed is the optimal one. It is furthermore confirmed by the computation on the random and handmade layouts (figures 3f and 3g) where the Markov strategy gains at least 5 turns on the other tested strategies.

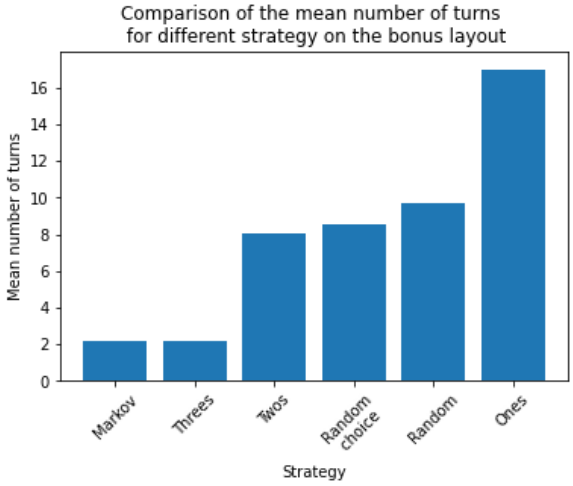
4.2.1 Comparison between Circle vs. Not Circle Board

All previous results were computed on layouts without the circle property, so one may wonder if the Markov decision process is as efficient on a layout that makes the player circle back to the start when going further than the goal. The same tests were therefore computed on the same layouts, but setting the Circle parameter to True. The obtained results are shown on figure 4. We can observe that the expected number of turn is as close to the empirical mean as without circle. The number of outliers are also lower than 5%. The comparison of strategies also shows that the computed strategy is indeed the most efficient.

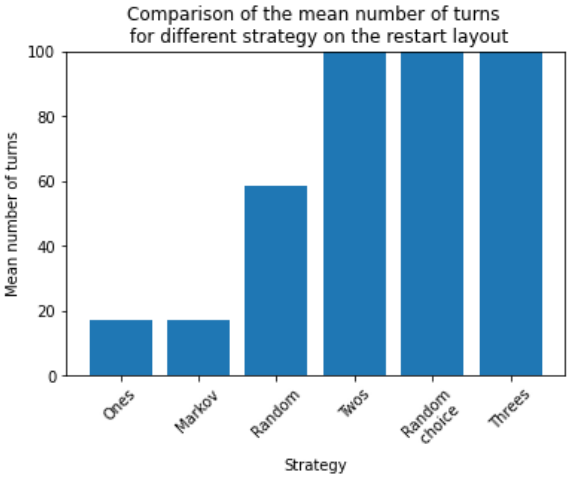
Those results were expected as the strategy on a layout with circle differs form one without circle only on the lasts few squares. Indeed, when using dice 2, the only squares that may lead to circling back to the start are the tenth square and fourteenth and this situation only happens one third of the time. With the dice 3, it can happen on the squares 9, 10, 13 and 14 but it has only 1 to 2 chances on 4 to happen. With the first dice, it will never happen, as we can move one square forward at most. So the strategy on a layout with circle usually changes the chosen dice for squares 9, 10, 13 and/or 14 from 2 or 3 to 1. It is confirmed by the graphs on figures 4b and 4c that are very similar to those obtained in section 4.2. However, for the other layouts, we can see some changes. The prison layout (figure 4d) and handmade layout (figure 4f) show that the Markov strategy is closer to choosing dice one at every turn, which is explained by the fact that the last cells choose the dice one to not circle back to the start.



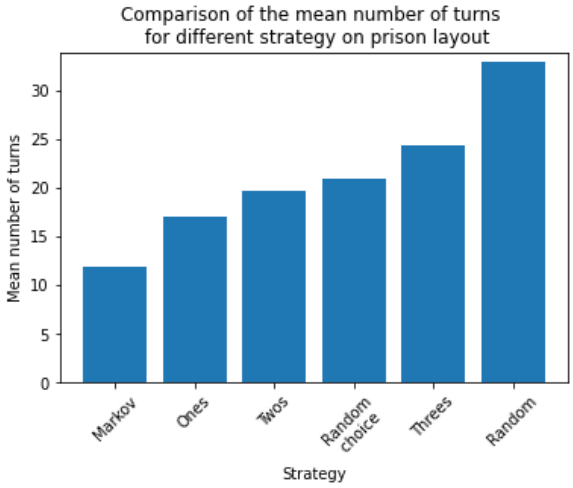
(a) Expected vs Empirical average



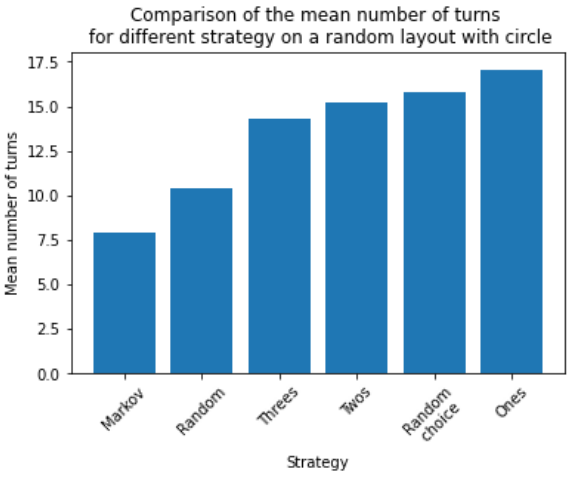
(b) Strategy comparison on the bonus layout



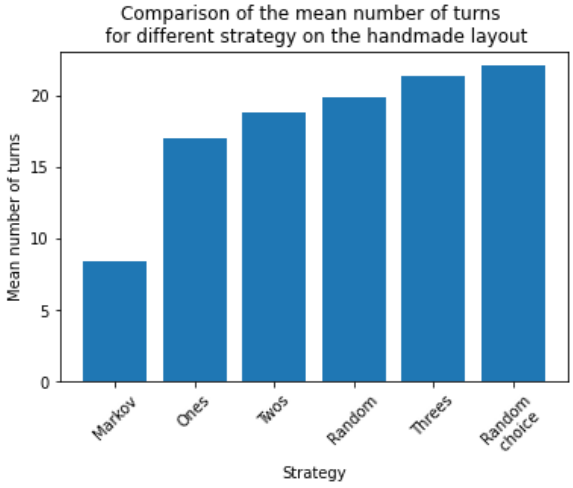
(c) Strategy comparison on the restart layout



(d) Strategy comparison on the prison layout



(e) Strategy comparison on a random layout



(f) Strategy comparison on the handmade layout

Figure 4: Performances of the Markov decision process on layouts with circle

5 Conclusion

Having in mind the results presented in the previous section, we can say that the model built is working as expected. It is easy to conclude that the developed model presents better results than all the other possible strategies to solve the problem presented. In the graphs presented in figures 3 and 4 we can observe that the Markov strategy is always the best one. This proves that the model is working as supposed and also that the Markov Process is useful to solve problems like this one.
The model applied in this project is considered to be very important to solve other problems within the framework of reinforcement learning. It is possible to understand, based on the model built, that in the Markov process it is only important to have information about the current state and based on that it is possible to determine the next actions and the correspondent costs. The project developed shows that this type of processes is independent of the past actions, and all the estimations are based on the present action.