LINFO2275 - Data Mining & Decision Making

Markov Decision Processes in the framework of a Snake and Ladders game

Academic Year: 2024 - 2025

Nathan Devaux
Louis-David Piron
Michela Selva

1 Introduction

In this project, we consider a simplified version of a one-player Snakes and Ladders game where, instead of rolling a predefined die, the player can choose between 3 different dice, offering distinct movement properties and levels of risk. In this setting, our goal is to use some concepts seen in class in order to determine an optimal dice-rolling strategy to reach the final square.

2 Theoretical explanation

A natural way to model this problem is through a **Markov Decision Process** (MDP). Indeed, the game satisfies the so-called Markov property, meaning that the optimal choice at any given state depends only on the current state and the actions available in it, not on the sequence of past moves (see [1], p.44). Moreover, since the board configuration and dice probabilities are fully known, this setting falls under model-based reinforcement learning, where solutions can be computed using dynamic programming techniques such as **Value Iteration**.

More formally, MDPs provide a mathematical framework for modeling decision problems where the transition to a **successor state** (k') depends only on the agent's current **state** (k), the chosen **action** (a), and the associated **transition probability** ($p(k'|k,a)$).

Solving an MDP consists of finding an optimal policy π^*, which defines the best action to take in each state to minimize the total expected cost. In our case, the policy corresponds to the optimal dice-rolling strategy, where the Agent must decide which die to roll at each square to minimize the number of moves required to reach the final square. Value Iteration provides an efficient way to compute this optimal policy by iteratively updating the value function :

$$\begin{cases} \hat{V}(k) \leftarrow \min_{a \in U(k)} \left\{ c(a|k) + \sum_{k'} p(k'|k,a) \hat{V}(k') \right\} & k \neq d \\ \hat{V}(d) \leftarrow 0 \end{cases} \tag{1}$$

where in our context:

- $\hat{V} : \mathcal{K} \to \mathbb{R}^+$ is the **value function**, where $\hat{V}(k)$ represents the expected number of turns required to reach the final state $d = 15$ from state k;
- $\mathcal{K} = \{1, \ldots, 15\}$ is the finite set of **states**, where each state represents a square on the board;
- $U(k) \subseteq \{a_1, a_2, a_3\}$ is the set of **available actions** at state k, where each action corresponds to choosing a die (a_1: secure, a_2: normal, a_3: risky);
- $c(a|k)$ denotes the expected cost associated with choosing die a in state k;
- $p(k'|k,a)$ represents the probability of reaching state k' given that action $a \in U(k)$ was taken in state k.

Some formulations of the Value Iteration Algorithm adopt a reward-based perspective, where the agent seeks to maximize cumulative reward. In our setting, we instead define a cost-based objective, aiming to minimize the total number of turns until reaching the goal.

Moreover, in some versions of the algorithm, a discount factor $\gamma \in [0, 1]$ is introduced to weigh future outcomes relative to immediate ones. Smaller values of γ emphasize short-term outcomes, while values close to 1 prioritize long-term planning. However, in episodic problems with guaranteed termination - as in our case - the use of a discount factor is not strictly necessary (see [2], p.54).

Finally, note that starting from an arbitrary initial value V_0, the sequence $\{V_k\}$ converges to V^*, as long as the destination state d is guaranteed from all states under the policy π (see [2], p.74 and 83). The impact of initialization in the context of our Snakes and Ladder game can be seen in **Appendix 1**.

3 Implementation

We have adopted an object-oriented approach to structure our implementation, encapsulating the game logic within the `BoardGame` class. This class defines the game environment, manages state transitions, and implements the Value Iteration algorithm. All functions, except `markovDecision()`, are methods of this class, ensuring that game mechanics and decision-making processes are neatly encapsulated (see **Appendix 2**).

Class structure and attributes

The `BoardGame` class is initialized with the argument of the `markovDecision()` function (`layout` and `circle`) as well as some dictionaries (defining the dice options, their probabilities, trap activation chances, *etc.*), and lists (tracking the fast and slow paths on the board).

Core Methods

- `ValueIteration()`: this is our main function, it returns a list `[Expec,Dice]`, containing the expected number of turns and optimal dice for each state k. Its implementation is based on the pseudo-code presented in **Appendix 2**.
- `BellmanOptimum()`: evaluates the expected cost-to-go from a given state for each possible die roll, and returns the action that minimizes this expected value, along with its associated cost `(best_action, min_value)`.
- `Move()`: returns the next position based on the player's current state and die roll, accounting for movement along fast and slow lanes.
- `TransitionProbabilities()`: returns a dictionary $\{k' : p(k'|k, a)\}$, which maps each possible successor state k' to the probability of transitioning from the current state k to k', given that action a was selected.
- `ApplyTraps()`: adjusts the transition probabilities from state k to successor states k' by incorporating the effects of traps. If a trap is present in k', the function modifies the transition probability based on the trap type and the probability of triggering it, depending on the chosen action a.

4 Validation of Theoretical Results

To evaluate the validity of the outputs produced by our `markovDecision()` function, we selected five distinct trap layouts and examined the results from three complementary perspectives:

1. Empirical (**Section 4.1**) : Do our theoretical results regarding the expected cost align with empirical outcomes obtained through simulation ?
2. Logical (**Section 4.2**) : Are the selected policies and their associated costs coherent (with respect to `Layout` and `Circle`) ?
3. Benchmarking (**Section 4.3**) : How does our optimal policy perform relative to various suboptimal strategies ?

4.1 Comparison between theoretical and empirical results

To assess the reliability of our theoretical cost estimates, we compared them with empirical outcomes across the five layouts that represent increasing levels of complexity and trap density. More information about the selected layouts and the simulation process are available in **Appendix 4** and **Appendix 5**.

For all layouts, the empirical and theoretical cost align almost perfectly. As shown in **Appendix 8**, deviations are symmetrically distributed around zero, indicating no systematic bias. The left panel demonstrates that deviations fluctuate randomly across different states and layouts, without any clear systematic pattern. This suggests that our theoretical model provides reliable predictions, with only minor, randomly distributed discrepancies.

The following figure depicts the empirical distributions at the initial state for Layout 1 and 5 :

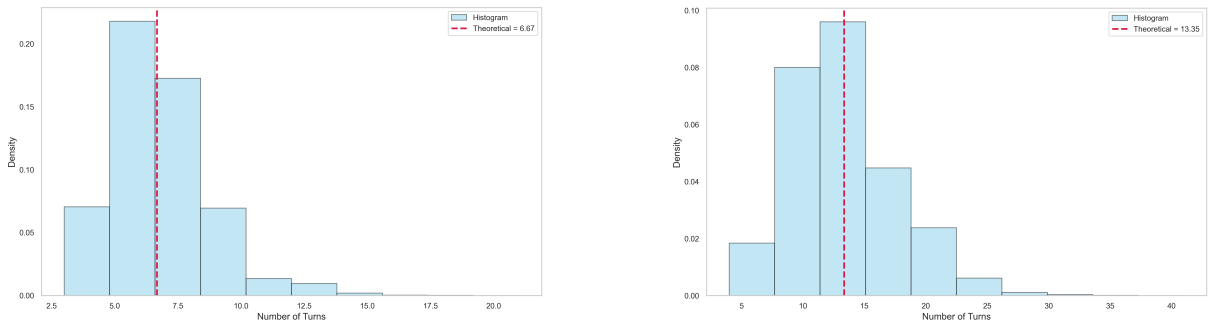


Figure 1: *Empirical distribution of the number of turns over 100,000 simulations for Layout 1 and No Circle (left panel) as well as for Layout 5 and No Circle (right panel). The theoretical value (corresponding to the expected cost to reach the destination state when applying the optimal policy) is represented in red.*

We notice that for both layouts, the mode of the empirical distribution matches our theoretical value, indicating once again that our theoretical estimates are consistent.

4.2 Policy Behavior Across Layouts

In this section, for each layout described earlier, we analyzed the optimal policy computed by `markovDecision()`, observing how dice choices adapt to risk, reward placement, and distance to the goal.

One general observation can already be made: in the cases where `circle = True`, the agent never takes risks with the possibility of restarting the game. In fact, even when the game has no traps, the agent will choose die 2 two square before the end, and will choose die 1 for the second-to-last square. It is also worth noting that the average number of turns for a given layout with `circle = False` will always be less than or equal to the same layout with `circle = True`.

Based on **Appendix 5** and **Appendix 6**, we can also observe that :

1. In **Layout 1**, the optimal policy almost exclusively selects die 3, which offers the fastest progression. The agent wants to go as fast as possible to the end.

2. In **Layout 2**, where the slow lane is filled with traps and the fast lane offers many bonuses, the agent favors strategies that enable it to enter the fast lane. After the branching point on square 3, a significant divergence in both theoretical and empirical costs is observed between the slow lane (square 4) and the fast lane (square 11), with the slow lane's expected cost being 6 turns higher. As a result, the agent selects die 2 on square 1 and die 1 on square 2 to maximize its chances of reaching the fast lane.

3. For **Layout 3**, we explore the inverse setup : traps are now placed on the fast lane. The design discourages taking the branching point at square 3, which leads to a 50-50 chance between the fast and slow lanes. Unlike in Layout 2, where the agent aimed to reach square 3, in Layout 3, the agent seeks to avoid it.

4. In **Layout 4**, we aimed to highlight how the agent reacts to the first and third types of traps. The first type is quite severe, while the third type is more forgiving, causing the agent to simply lose a turn. Interestingly, even when there are multiple type 3 traps along the path, the agent still prefers to use die 3. Conversely, when there is a chance of triggering a type 1 trap, the agent consistently chooses die 1. The agent doesn't want to take a risk against traps of type 1.

5. **Layout 5** incorporates all types of traps and reinforces the conclusions drawn earlier. As seen in Layout 4, the agent avoids risks when confronted with type 1 traps. Additionally, the agent faces a dilemma with the fast lane, weighing the potential rewards against the risks of a shorter path, which may have more traps. This decision-making process is evident when the agent selects die 1 at square 2 and die 2 at square 1, indicating a strong desire to take the fast lane.

4.3 Comparison with Suboptimal Strategies

To further assess the quality of our optimal policy, we compare it against several suboptimal strategies: fixed dice policies (always using die 1, 2 or 3), and random dice rolls [1].

We evaluate these policies along two complementary dimensions. First, *variability*: a good policy should not only perform well on average, but also deliver consistent outcomes across episodes. Second, *performance*: how much better, on average, is the optimal strategy compared to its alternatives?

To quantify performance, we compute the **relative gain**, defined as the average reduction in expected number of turns compared to a given suboptimal strategy :

$$\text{Relative Gain} = \frac{\bar{V}_{\text{obtimal}} - \bar{V}_{\text{subobtimal}}}{\bar{V}_{\text{subobtimal}}} \tag{2}$$

where $\bar{V}$ denotes the average number of turns to reach the destination state d.

To assess variability, we consider the **interquartile range** (IQR), which captures the spread of the middle 50% of observed outcomes. The IQR is robust to outliers and provides a clear picture of stability across runs [3].

In Table 1 (see **Appendix 7**), we present a summary of the key metrics across the five previously described layouts, each analyzed under two conditions: `Circle = True` and `Circle = False`. This results in a total of 10 distinct scenarios. For each scenario, we report the metrics under five different policies: the optimal policy and four suboptimal ones. For each policy, we conducted 100,000 independent simulations of the game, resulting in a total of $5 \cdot 10^6$ simulations. We notice that several patterns emerge :

1. The **optimal policy** consistently achieves the lowest average costs across all environments. It also generally exhibits lower or comparable IQR values, indicating not only efficiency but also a high degree of stability in performance.

2. The **Die 1 policy** remains identical across all scenarios, since Die 1 has no chance of triggering traps and can at most advance by one square, which prevents overshooting the destination. Despite this predictability, it systematically results in significantly higher average costs and IQRs than the optimal policy.

3. The **Die 2 policy** produces highly variable outcomes. Due to its 50% probability of triggering traps, it performs reasonably well in simpler settings, but its performance deteriorates sharply in more complex environments (*e.g.*, Layout 4, mean = 43.4, IQR = 41.0).

4. The **Die 3 policy** occasionally approaches the optimal policy's performance in simple cases (e.g., Layout 3 + No Circle), but generally lacks consistency. Its 100% probability of triggering traps results in high costs and large variability in more difficult layouts, making it one of the least robust strategies.

[1] The same random dice strategy was used across all simulations. It was initially selected arbitrarily using `np.random.randint(1, 4, size=14)`.

5. The **Random policy** shows inconsistent performance across environments. While it remains clearly suboptimal compared to the optimal policy, it often performs better than the `Die 3` policy and is frequently comparable to Die in terms of average cost and variability.

Visually, if we look at how the 5 policies behave with 2 extreme settings (layout 1 - empty board, and layout 5 - various traps), we notice those patterns again :

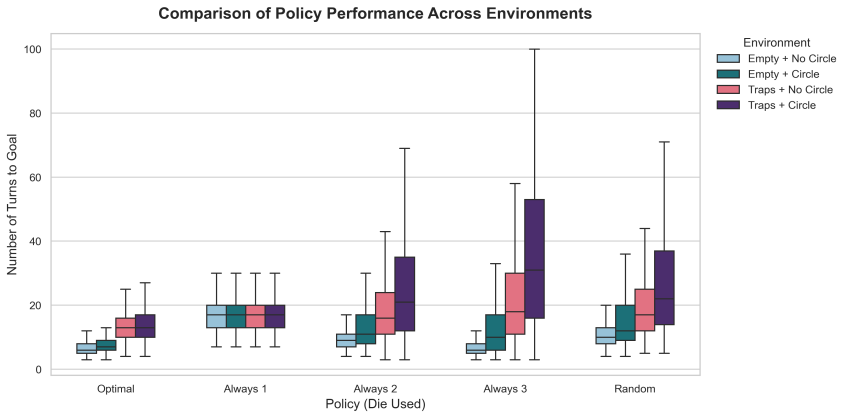


Figure 2: *Empirical distribution of the number of turns for 4 environments and 5 policies (100,000 simulations each)*

In the following table, we see that the optimal policy consistently outperforms the sub-optimal strategies :

Policy	**Best Relative Gain (%)**	**Worst Relative Gain (%)**
Die 1	-66.20	-6.60
Die 2	-63.40	-27.70
Die 3	-90.4	0.00
Random	-63.50	-34.60

Table 1: *Relative performance of the optimal policy across different environments. Best (Worst) Relative Gain means the highest (smallest) performance improvement*

5 Conclusion

Throughout this project, we demonstrated how MDPs provide a structured framework for modeling sequential decision-making. Our analysis confirmed that theoretical cost estimates align closely with empirical results, with deviations symmetrically distributed around zero, indicating no systematic bias. In addition, our logical analysis confirms that the agent's choices are intuitive across various board layouts. In high-risk environments, it consistently opts for safer strategies, whereas in less perilous settings, it takes calculated risks to advance more quickly. Benchmarking further shows that the optimal policy outperforms suboptimal strategies in terms of both variability and performance. Ultimately, the agent's measured and prudent approach reflects the adage "*slow and steady wins the race*," underscoring the value of balancing caution with opportunity in decision-making.

Appendix 1 : Effect of initialization on convergence

While the final policy remains unchanged due to the guaranteed convergence of Value Iteration under standard conditions, the choice of initial values can significantly affect the number of iterations required to reach convergence. In particular, well-informed initializations - such as estimated costs based on the safest die - can accelerate convergence substantially.

The figures below illustrate the convergence behavior of the estimated value $\hat{V}(0)$—that is, the expected number of turns from the initial state toward the optimal value $V^*(0)$ for different initialization strategies. On the first two graphs, the behavior of convergence under Layout 1 is displayed, while on the second set of graphs, the one under Layout 5 is displayed.

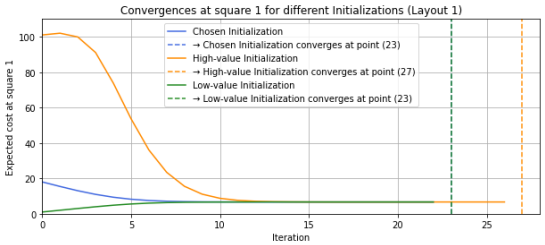


Figure 3: *Convergence of Layout 1*

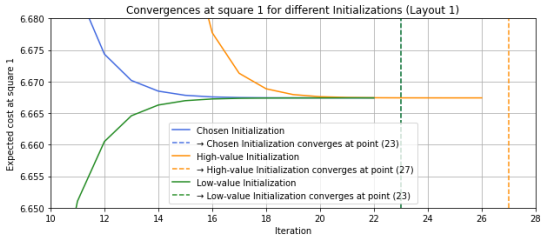


Figure 4: *Close-up Layout 1*

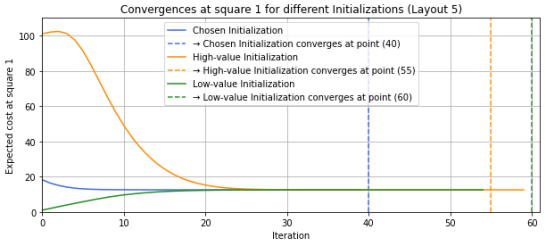


Figure 5: *Convergence of Layuout 5*

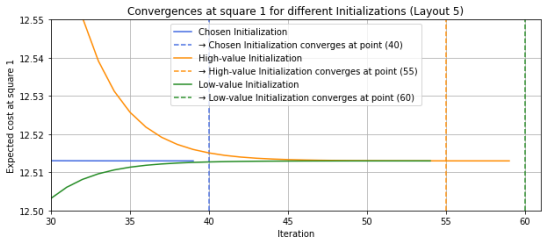


Figure 6: *Close-up Layout 5*

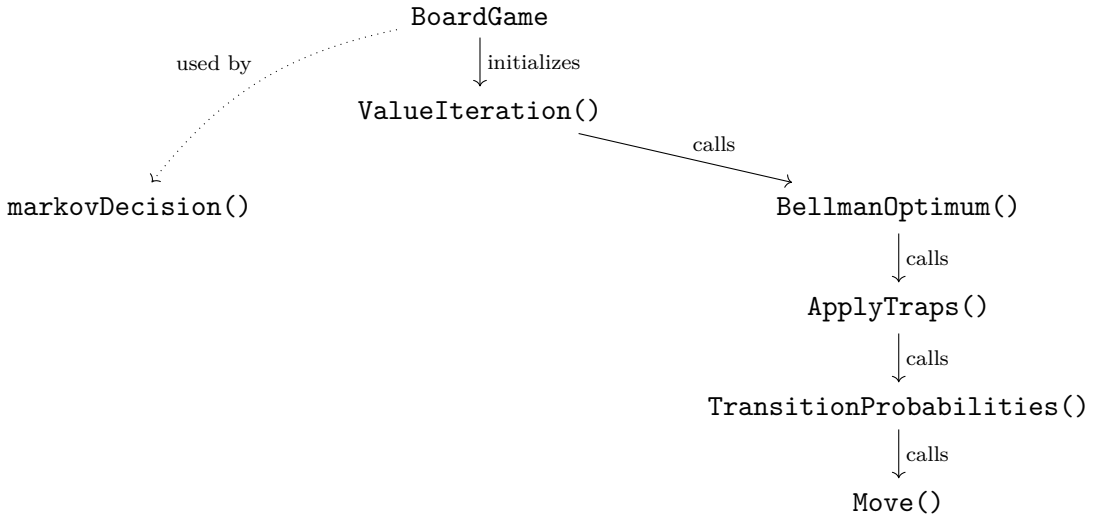
Three initialization schemes were tested:

- **High-value Initialization** (orange): all values initialized to 100, leading to a rapid decrease but slower convergence due to overshooting and oscillation.
- **Low-value Initialization** (green): all values initialized to 0, resulting in a slow but steady increase toward the optimal values.
- **Chosen Initialization** (blue): initialized using a decreasing pattern based on intuitive estimates from the structure of the game, e.g., `[20., 18., 16., 14., 12., 10., 8., 6., 4., 2., 8., 6., 4., 2.]`. This informed structure led to the fastest convergence.

Each curve represents the trajectory of $\hat{V}_k(0)$ across iterations. The vertical dashed lines indicate the iteration at which each initialization is considered to have **converged**, meaning that the maximum change in value across all states fell below a small tolerance (`epsilon = 1e-6`).

This experiment highlights the importance of initialization: while the final value function V^* is invariant, the rate at which it is approached depends heavily on the initial estimate.

Appendix 2 : Code architecture



Appendix 3 : Value Iteration Algorithm

Algorithm 1 Value Iteration Algorithm

$\Delta \leftarrow 0$
repeat
  $\Delta \leftarrow 0$
  for each $k \in \mathcal{K}$ **do**
    $V'(k) \leftarrow \min_{a \in U(k)} \left\{ c(a|k) + \sum_{k'} p(k'|k,a) \hat{V}(k') \right\}$
    $\Delta \leftarrow \max(\Delta, |V'(k) - V(k)|)$
  end for
  $V \leftarrow V'$
until $\Delta \leq \theta$

Appendix 4 : Simulation

For the simulations we implemented the function `simulate_strategy(layout, circle, game_strategy, simulations)`. This function takes into account 4 arguments : `layout` and `circle` (the same arguments as in the `markovDecision()`), `game_strategy` is the policy the Agent has to follow throughout the simulations, `simulations` is the number of simulations.

Appendix 5 : Optimal policies for different layouts

State (k)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Layout 1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Optimal policy (F)	3	3	3	3	3	3	3	3	3	3	3	3	3	3	
Optimal policy (T)	3	3	3	3	3	3	3	3	2	1	3	3	2	1	
Layout 2	**0**	**0**	**0**	**2**	**2**	**2**	**3**	**3**	**3**	**2**	**4**	**4**	**4**	**4**	**0**
Optimal policy (F)	2	1	3	3	3	3	2	1	1	1	3	3	3	3	
Optimal policy (T)	2	1	3	3	3	3	2	1	1	1	3	3	2	1	
Layout 3	**0**	**0**	**0**	**0**	**4**	**0**	**4**	**0**	**4**	**0**	**2**	**2**	**2**	**2**	**0**
Optimal policy (F)	3	3	3	3	3	3	3	3	3	3	2	3	3	1	
Optimal policy (T)	3	3	3	3	3	3	3	3	2	1	2	3	1	1	
Layout 4	**0**	**1**	**1**	**3**	**3**	**3**	**3**	**1**	**1**	**3**	**3**	**3**	**3**	**1**	**0**
Optimal policy (F)	1	1	1	3	2	1	1	1	1	3	2	1	1	1	
Optimal policy (T)	1	1	1	3	2	1	1	1	1	1	2	1	1	1	
Layout 5	**0**	**0**	**3**	**3**	**3**	**1**	**3**	**2**	**4**	**0**	**0**	**4**	**2**	**1**	**0**
Optimal policy (F)	2	1	1	1	1	1	1	1	3	3	1	1	1	1	
Optimal policy (T)	2	1	2	1	1	1	1	1	2	1	1	1	1	1	

Table 2: *Optimal policies for different layouts and states*

Appendix 6 : Theoretical/Empirical costs for 5 different layouts

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
Layout 1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Theoretical costs (F)	6.67	6.11	4.77	5.12	4.43	3.77	3.16	2.37	1.78	1.33	3.16	2.37	1.78	1.33
Empirical costs (F)	6.68	6.11	4.78	5.14	4.45	3.77	3.17	2.37	1.78	1.34	3.17	2.36	1.77	1.34
Theoretical costs (T)	7.26	6.69	5.37	5.71	4.99	4.37	3.78	2.83	2.5	2.00	3.78	2.83	2.50	2.00
Empirical costs (T)	7.27	6.69	5.37	5.72	4.92	4.40	3.80	2.83	2.51	2.00	3.78	2.86	2.52	2.00
Layout 2	**0**	**0**	**0**	**2**	**2**	**2**	**3**	**3**	**3**	**2**	**4**	**4**	**4**	**4**
Theoretical costs (F)	6.67	6.17	4.17	7.31	7.40	7.10	7.25	6.00	4.00	2.00	1.00	1.00	1.00	1.00
Empirical costs (F)	6.67	6.19	4.13	7.30	7.39	7.10	7.22	5.98	3.95	2.00	1.00	1.00	1.00	1.00
Theoretical costs (T)	7.52	7.02	5.02	7.95	7.82	7.31	7.25	6.00	4.00	2.00	1.89	1.67	2.00	2.00
Empirical costs (T)	7.41	7.02	5.06	8.01	7.86	7.29	7.27	6.00	4.00	2.00	1.87	1.67	2.00	2.02
Layout 3	**0**	**0**	**0**	**0**	**4**	**0**	**4**	**0**	**4**	**0**	**2**	**2**	**2**	**2**
Theoretical costs (F)	5.75	4.79	5.24	3.22	2.9	2.53	2.23	1.93	1.44	1.33	6.13	5.04	3.84	2.00
Empirical costs (F)	5.74	4.79	5.19	3.24	2.89	2.52	2.23	1.92	1.45	1.35	6.09	5.02	3.84	2.01
Theoretical costs (T)	6.38	5.41	5.86	3.86	3.5	3.19	2.89	2.42	2.25	2	6.62	5.47	4.00	2.00
Empirical costs (T)	6.36	5.37	5.87	3.89	3.48	3.20	2.89	2.43	2.28	1.99	6.62	5.47	3.98	2.00
Layout 4	**0**	**1**	**1**	**3**	**3**	**3**	**3**	**1**	**1**	**3**	**3**	**3**	**3**	**1**
Theoretical costs (F)	15.67	13.67	11.67	12.08	10.92	9.67	7.67	5.67	3.67	1.67	7.25	6.00	4.00	2.00
Empirical costs (F)	15.62	13.63	11.66	12.01	10.91	9.67	7.63	5.69	3.67	1.67	7.27	6.02	4.00	1.99
Theoretical costs (T)	15.83	13.83	11.83	12.42	11.25	10.00	8.00	6.00	4.00	2.00	7.25	6.00	4.00	2.00
Empirical costs (T)	15.77	13.89	11.75	12.41	11.21	10.00	7.96	5.99	3.97	2.00	7.19	5.99	3.99	2.00
Layout 5	**0**	**0**	**3**	**3**	**3**	**1**	**3**	**2**	**4**	**0**	**0**	**4**	**2**	**1**
Theoretical costs (F)	13.35	12.6	10.6	11.44	9.44	7.44	5.44	3.44	1.44	1.33	8.00	6.00	4.00	2.00
Empirical costs (F)	13.39	12.58	10.65	11.45	9.47	7.42	5.42	3.45	1.45	1.34	8.05	6.00	4.01	2.04
Theoretical costs (T)	13.75	13	11	12.25	10.25	8.25	6.25	4.25	2.25	2	8	6	4	2
Empirical costs (T)	13.77	13.03	10.98	12.25	10.24	8.26	6.23	4.24	2.26	1.99	7.95	5.96	4.01	1.99

Table 3: *Theoretical and empirical costs under different layouts and states*

Appendix 7 : Empirical performance and stability of each policy

Environment	Policy	Mean	IQR	Relative Gain (%)
Layout 1 + No Circle	Optimal	6.70	3.00	–
	Die 1	17.00	7.00	-60.80
	Die 2	9.20	4.00	-27.70
	Die 3	6.70	3.00	0.00
	Random	10.70	5.00	-37.60
Layout 1 + Circle	Optimal	7.30	3.00	–
	Die 1	17.00	7.00	-57.40
	Die 2	13.90	9.00	-47.80
	Die 3	13.40	11.00	-45.70
	Random	15.90	11.00	-54.20
Layout 2 + Circle	Optimal	7.50	6.00	–
	Die 1	17.00	7.00	-55.90
	Die 2	15.80	14.00	-52.70
	Die 3	18.70	18.00	-60.00
	Random	23.30	21.00	-67.80
Layout 2 + No Circle	Optimal	6.70	6.00	–
	Die 1	17.00	7.00	-60.60
	Die 2	10.80	8.00	-38.30
	Die 3	9.40	8.00	-28.50
	Random	16.80	14.00	-60.20
Layout 3 + Circle	Optimal	6.40	4.00	–
	Die 1	17.00	7.00	-62.40
	Die 2	15.40	11.00	-58.60
	Die 3	11.60	10.00	-44.70
	Random	15.80	11.00	-59.50
Layout 3 + No Circle	Optimal	5.70	3.00	–
	Die 1	17.00	7.00	-66.20
	Die 2	10.30	4.00	-44.30
	Die 3	5.70	3.00	0.00
	Random	10.50	5.00	-45.20
Layout 4 + Circle	Optimal	15.90	7.00	–
	Die 1	17.00	7.00	-6.60
	Die 2	43.40	41.00	-63.40
	Die 3	165.20	173.00	-90.40
	Random	43.50	40.00	-63.50
Layout 4 + No Circle	Optimal	15.70	7.00	–
	Die 1	17.00	7.00	-7.90
	Die 2	30.80	26.00	-49.10
	Die 3	55.10	52.00	-71.60
	Random	32.60	27.00	-51.90
Layout 5 + Circle	Optimal	13.80	7.00	–
	Die 1	17.00	7.00	-19.00
	Die 2	27.20	24.00	-49.40
	Die 3	49.30	49.00	-72.00
	Random	29.70	24.00	-53.60
Layout 5 + No Circle	Optimal	13.30	6.00	–
	Die 1	17.00	7.00	-21.60
	Die 2	19.10	13.00	-30.30
	Die 3	23.10	19.00	-42.30
	Random	20.40	13.00	-34.60

Table 4: *Average cost, variability (IQR), and relative gain of the optimal policy compared to suboptimal ones across different environments.*

Appendix 8 : deviations between theoretical and empirical values

The two following graphs depict the deviation between the theoretical and empirical values for our five layout (with `circle = False`) and for one game simulation :

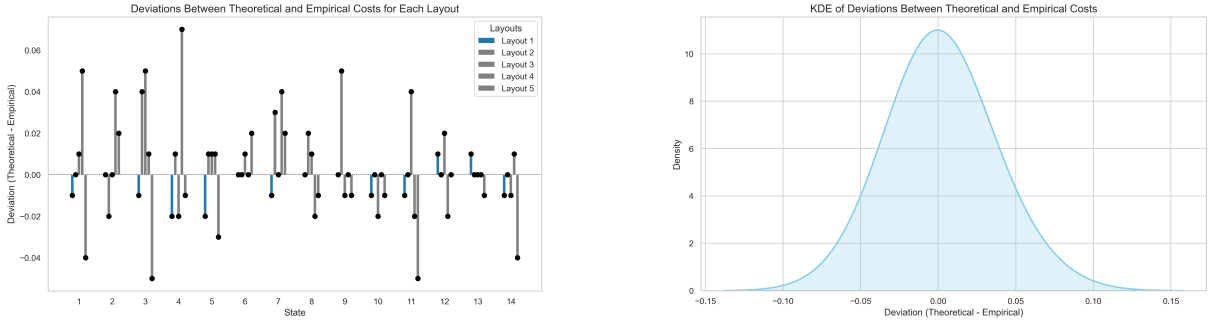


Figure 7: *Empirical distribution of the deviations accross states and layout (left panel) as wall as overall KDE of the deviations (right panel)*

References

[1] Plaat A., *Learning to Play, Reinforcement Learning and Games*, Springer, 2022.

[2] Richard S. Sutton & Andrew G. Barto. *Reinforcement Learning: An Introduction*. Second Edition, MIT Press, 2018.

[3] Wan X., Wang W., Liu J., & Tong T., *Estimating the sample mean and standard deviation from the sample size, median, range and/or interquartile range*, BMC Medical Research Methodology, vol. 14, p. 135, 2014.

[4] François B., & Dethier D., *Reinforcement Learning: Framework, Algorithms and Applications*. University of Liège, 2006