Reinforcement Learning for the King-Rook vs King Chess Endgame

Work done by Belsem JEMAI & Mohamed Ramez MABROUK 3ATEL1

1. Introduction

Reinforcement learning (RL) in games offers a controlled setting in which an agent must learn long-horizon decision-making from reward alone. We study a classical chess sub-problem: the **King-Rook versus King (KRK) endgame**, in which the side holding the rook must force checkmate against a lone king. KRK is small enough to be solved with tabular methods, yet it already exhibits the two features that make control hard: a **large state space** and a **sparse, delayed reward** (the only decisive event, mate, occurs at the very end of a possibly long game).

Our objective is threefold :

- Formulate KRK as a Markov Decision Process (MDP)
- Solve it with two tabular control families studied in the course, **Monte-Carlo (MC)** and **Temporal-Difference (TD, Q-learning)**, on the *same* environment and reward
- Argue, with theory and experiments, which family is better suited to this problem.

2. MDP Formulation

The agent plays White (King + Rook) and is always the side to move when it acts. The lone Black king is part of the environment and follows a fixed defensive policy, so the agent faces a stationary environment.

Element	Definition
State s	(white_king_square, white_rook_square, black_king_square), White to move
Action a	a legal White move (king or rook); legality, check, mate and stalemate are handled by python-chess
Opponent	lone Black king: stays as central as possible and captures a hanging rook (draw). A 'random' defender is also available
Reward	−1 per agent move; −100 on any draw (stalemate, rook captured, move limit); mate ends the episode
Termination	checkmate (win), draw, or a 60-move cap, so every episode terminates and the task is episodic

The reward design is the classical **shortest-path-to-mate** scheme: with a discount factor γ = 1 the return of a won game equals −(number of moves), so maximising return is equivalent to **mating in as few moves as possible** while strongly avoiding draws. Formally the agent seeks the optimal action-value function

$$Q^*(s,a) = E[\, R_{t+1} + \gamma \cdot \max_{a'} Q^*(S_{t+1}, a') \mid S_t=s, A_t=a \,]$$

3. Chronological Design Process

The project was built incrementally across the lab sessions; this section documents our progress and the main design decisions.

- **Session 2: Problem & MDP.** We chose RL-in-gaming and scoped it to the KRK endgame. We defined the state, the action set and the reward, and settled the roles (agent = White, environment = defending Black king).
- **Session 4: Monte-Carlo solution.** We implemented first-visit on-policy MC control with an ε-greedy policy and sample-average updates. This was our first working learner and our baseline.
- **Session 6: Temporal-Difference solution.** We implemented Q-learning (off-policy TD) on the identical environment, so that the only difference from the MC agent is the update rule, keeping the comparison fair.

- **Session 7: Execution & evaluation.** We trained both agents, evaluated their greedy policies over 2000 games, and produced the learning curves. A key empirical finding emerged here (Section 7).
- **Design pivot: reward shaping.** Early runs showed that with the pure sparse reward neither agent learned in a feasible budget: over ~160 000 states the greedy policy was effectively random and kept hanging the rook. We therefore added **potential-based reward shaping** (Section 6), which is policy-invariant and made the task learnable.

4. Monte-Carlo Control

MC control plays a **complete episode** with the current ε-greedy policy, then computes the return G from the end backwards (γ = 1) and, for the **first** occurrence of each (s,a), moves its estimate toward the observed return with a sample-average step:

$$G \leftarrow r + \gamma \cdot G \;\; ; \;\; N(s,a) \leftarrow N(s,a)+1 \;\; ; \;\; Q(s,a) \leftarrow Q(s,a) + (1/N(s,a)) \cdot (G - Q(s,a))$$

MC is **unbiased** and needs no model, but it **does not bootstrap**: a state-action value is updated only once the whole episode is over, and it receives the full, high-variance Monte-Carlo return of that trajectory.

5. Temporal-Difference Control (Q-learning)

Q-learning updates after **every single move**, bootstrapping on the current estimate of the next state:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \cdot [\, r + \gamma \cdot \max_{a'} Q(s',a') - Q(s,a) \,]$$

The target uses only r on terminal transitions. Because the update propagates value one step backwards at each move, the learned value function behaves like a **distance-to-mate**: the information contained in the terminal reward spreads through the state space far faster, and with much lower variance, than in MC.

6. Potential-Based Reward Shaping

To make the sparse signal learnable we added a shaping term $F = \gamma \cdot \Phi(s') - \Phi(s)$, with $\Phi(\text{terminal}) = 0$. By the policy-invariance theorem (Ng, Harada & Russell, 1999) this **does not change the optimal policy**: it only guides exploration. Our potential rewards the two ingredients of a KRK mate (push the Black king to the edge, bring the White king in to support):

$$\Phi(s) = -2 \cdot d_{\text{edge}}(\text{black_king}) \; - \; 1 \cdot d_{\text{Chebyshev}}(\text{white_king}, \text{black_king})$$

The same shaping is applied to both agents, so the MC-vs-TD comparison remains fair.

7. Experiments and Results

Setup :

- Both agents use exploring starts (random legal positions)
- ε-greedy exploration decayed from 1.0 to 0.1
- γ = 1
- Q-learning α = 0.4
- A 60-move cap and the heuristic defender

Policies are evaluated greedily (ε = 0, shaping off) over 2000 games, that is, on the **true** mate objective.

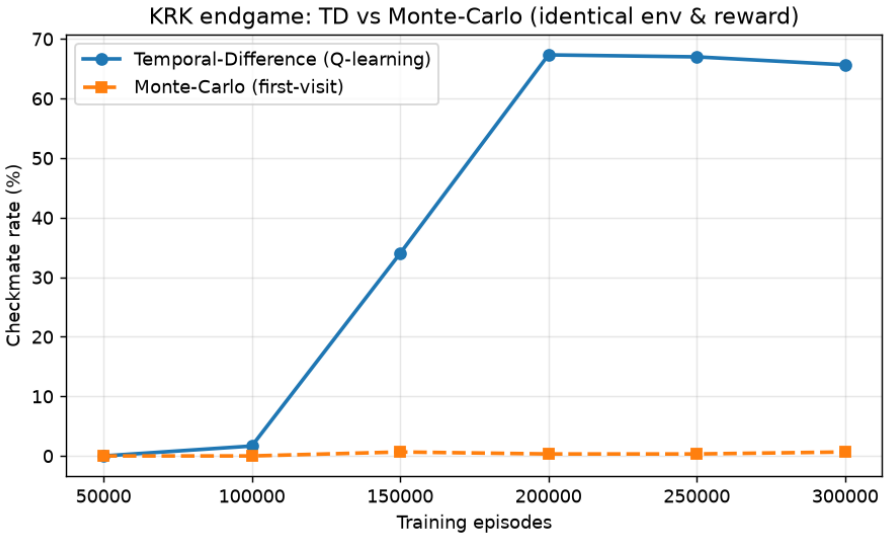


Figure 1. Checkmate rate of the greedy policy during training (300 000 episodes each, identical environment and reward). TD rises to ~65%, MC stays near 0%.

Method	Mate rate	Draw rate	Avg. moves
Temporal-Difference (Q-learning)	61.7 %	38.3 %	24.7
Monte-Carlo (first-visit)	0.5 %	99.5 %	3.1

Table 1. Final greedy-policy performance (2000 games) after 300 000 training episodes.

8. Why KRK is Better Suited to Temporal-Difference

Both methods are valid for this episodic task, yet KRK matches the strengths of TD and the weaknesses of MC almost point for point:

- **Long episodes + sparse, delayed reward.** A learning KRK game can last dozens of moves before the single decisive reward at mate. MC must wait for the end and then assigns the same noisy return to every (s,a) on the trajectory, giving high variance and very slow credit assignment. TD updates each move and bootstraps, so the terminal signal propagates efficiently backwards.
- **Credit assignment as 'distance to mate'.** An endgame is about making monotone progress (shrink the box, drive the king to the edge). TD naturally learns a value that reflects proximity to mate; MC only sees the final outcome and struggles to rank intermediate progress states.
- **Deterministic environment + online learning.** Chess transitions are deterministic; off-policy Q-learning can learn the optimal policy while still exploring, updating continuously, which is stable and sample-efficient here.

Our experiment confirms this. Under an identical environment, reward and budget, TD reaches a 60 to 67% mate rate while MC never gets off the ground (≈0.5%, essentially the handful of positions that were already one move from mate). This is exactly the behaviour the theory predicts: on long-horizon, sparse-reward tasks, bootstrapping beats full-episode averaging.

9. Self-Critique and Improvements

Limitations :

- The value function is **tabular**, so nothing is shared between similar positions and ~160 000 states must each be visited many times.
- TD still draws ~38% of games (coverage gaps and timeouts remain).
- The Black defender is a heuristic, not optimal.

- The shaping potential was hand-designed.

Improvements :

- Exploit the board's 8-fold symmetry (×8 smaller state space)
- Replace the table with **function approximation** (hand-crafted features or a DQN) for generalisation
- Train against an **optimal (tablebase) defender**
- Use a decaying learning rate and variants such as **Expected-SARSA** or **n-step / TD(λ)** (a continuum between MC and TD)
- Add prioritised sweeping
- Simply train longer, since the TD curve is still rising.

10. Conclusion

We modelled the KRK endgame as an MDP and solved it with Monte-Carlo and Temporal-Difference control on an identical environment. Theory and experiment agree that **TD is the better-suited method** for this long-horizon, sparse-reward problem: Q-learning learns a mating policy (≈62% mate rate) while MC, under the same budget, does not. The project also showed, in passing, the practical importance of policy-invariant reward shaping for making sparse RL tasks learnable.