Professor Eleni Vasilaki & Dr Luca Manneschi

Assignment

February 17, 2022

1. Reinforcement Learning

1.1. Overview

You may tackle this assignment in groups of (up to) three students or individually. You will need to submit a report and a GitHub link containing your code. When you work in a group, you will need to submit a description of how you divided the work and a log of your meetings (up to 1 page long). Otherwise, the submission will be considered incomplete. In general, we expect to see commits in the code from all group members. If you prefer to work individually, to balance out the effort, you will **not** have to address all the questions of this assignment. These will be marked in the "Assignment description" as [**Group Only**] tasks.

1.2. Assignment description

The task consists of a chessboard 4x4. Three pieces, **1x King (1)**, **1x Queen (2)** and **1x Opponent's King (3)**, are placed in a random location of the board. In the initial positions, the pieces are not causing any threats.
Assuming the role of player 1 (white), you aim to checkmate the Opponent's King (player 2, black). The game progresses as follows:

1. Player 1 moves the King or the Queen aiming checkmate.
2. Player 2 randomly moves the Opponent's King to a safe position (i.e. where Player 1's King or Queen doesn't threaten it).
3. Repeat the above sequence until:
   - **Checkmate**. Player 2 plays. Its King is threatened. There are no squares that the King can move.
   - **Draw**. Player 2 plays. Its King is not threatened. There are no squares that the King can move.

Please note that the King is not allowed to move to a square where he is threatened.
Your task is to implement a deep neural network capable of learning to perform checkmate. The features representing the states are denoted by a vector $\mathbf{s}(t)$, whose dimensionality is $3N^2 + 10$, where $N \times N$ is the size of the board and 3 is the number of the pieces involved in this task. Thus, the position of each piece is decoded by a binary matrix C_{ij} (the chessboard), where $c_{ij} = 1$ if that particular piece is in the position ij and $c_{ij} = 0$ otherwise. The $+10$ term in the dimension of $\mathbf{s}(t)$ contains: (i) the degree of freedom of the opponent King (8 options in 1-hot-encoding), i.e. the number of available

squares where the Opponent's King is allowed to move, and (ii) whether the Opponent's King is threatened or not (2 options in 1-hot-encoding).
We provide you with documented code implementing the environment (in Python). You only need to implement the updated rules for the backdrop and TD algorithms (please see appendix A). You are allowed to use (modified where appropriate) code from earlier Labs you have developed on your own or from the Lab solutions we have provided to you.

Your specific tasks for this assignment are:

1. Describe the algorithms Q-learning and SARSA. Explain the differences and the possible advantages/disadvantages between the two methods. No coding is needed to reply to this question.

2. **[Group Only]** Describe the experience replay technique. Cite relevant papers.

3. We provide you with a template code of the chess game. Fill the gaps in the program file *chess* implementing either Q Learning or SARSA; detailed instructions are available inside the file. We provide indicative parameter values. Produce two plots that show the reward per game and the number of moves per game vs training time. The plots will be noisy. Use an exponential moving average.

4. Change the discount factor γ and the speed β of the decaying trend of ϵ (for a definition of β please see code). Analyse the results.

5. Implement another Deep RL algorithm (it could be SARSA if you chose before Q Learning and vice versa). Compare the new results with your previous results.

6. **[Group Only]** Change the state representation or the administration of reward. Interpret your results.

7. **[Group Only]** The values of the weights could explode during learning. This issue is called *exploding gradients*. Explain one possible way to fix this issue and implement it. For example, search and study *RMSprop*). Show in a plot how your solution fixed this issue.

2. Report

Your report should adhere to scientific standards, i.e. a journal paper-like format with sections: introduction, methods, results, conclusions. We recommend that you search the web for relevant journal papers and mimic their structure. For instance, you can use the IEEE templates `https://www.ieee.org/conferences/publishing/templates.html`. Explain and justify your results. Figures should have captions and legends. The individual reports should **NOT exceed four pages** and the group reports should **NOT exceed six pages** (excluding Appendix and Cover Page). We recommend a two-column format. The minimum font size is 11pt. Margins should be reasonable. Kindly note that we mark taking into account the readability and clarity of your document. Your report should include:

1. Reply to the questions.

2. An Appendix with snippets of your code referring to the algorithm implementations, with comments/explanations, where you would like to highlight any novelties you have introduced (i.e. your ideas that go beyond the Lab code).

3. A description on how we can reproduce your results, see also “Important Note”.

Important Note. Please make sure that your results are reproducible. You can, for instance, initialise a random seed to make sure you always get the same results. Together with the assignment, please provide code well commented. Provide sufficient information so that we can easily test your results. If your results are not reproducible by your code, the assessment cannot receive full points. If you do not provide your code, the submission is incomplete.

3. Suggested language

The recommended programming language is Python.

4. Marking

To maximise your mark, please make sure you explain your model. Please evidence your results (e.g., plots with captions, scientific arguments). Figures should be combined wherever appropriate to show a clear message. Any original additions you have employed should be highlighted and well explained.

5. Submission

The **deadline** for emailing the assignment to vasilaki@ifi.uzh.ch is **the 31 of March 2022, 23:59** (in PDF format). Your report should include a link to your GitHub project of the assignment. If you submit a group project, you will also need to submit a short description of how you divided the workload. Also, a log of the group meetings (see also section "Overview"). Only one member of the team needs to submit.
Please note that your GitHub repository needs to be "public". If for any reason you wish to keep it private, add "eleni-vasilaki" and "LucaManneschi" as collaborators so we can assess your work.

6. Plagiarism, and Collusion

You are permitted to use Python/Matlab code developed for the lab as a basis for the code you produce for this assignment or other code that inspired you with **appropriate acknowledgement**. It will not affect your mark. There is no credit to material copied (either unchanged or minimally modified) from published sources, including websites. Please cite any sources of information that you use.

A. Chess

Chess is a two-player strategy game played on a 64 square board (chessboard) arranged in an 8x8 grid. Each player is given 16 pieces in total falling under 6 types and each type has a unique set of moves. The objective of the game is to **checkmate** the opponent's king by placing it into a position where it is threatened by a piece and cannot perform any action to escape the threat. Another way that a chess game can end is by **draw** which happens if the remaining pieces in the chessboard are unable to perform a checkmate or the king *(a)* is not threatened, *(b)* is the only piece that can perform an action and *(c)* any action will cause him a threat. More information on chess and how each piece moves can be found online (e.g. here), for your assignment you will use a simplified version of the chess with only 3 total pieces in the chessboard board.

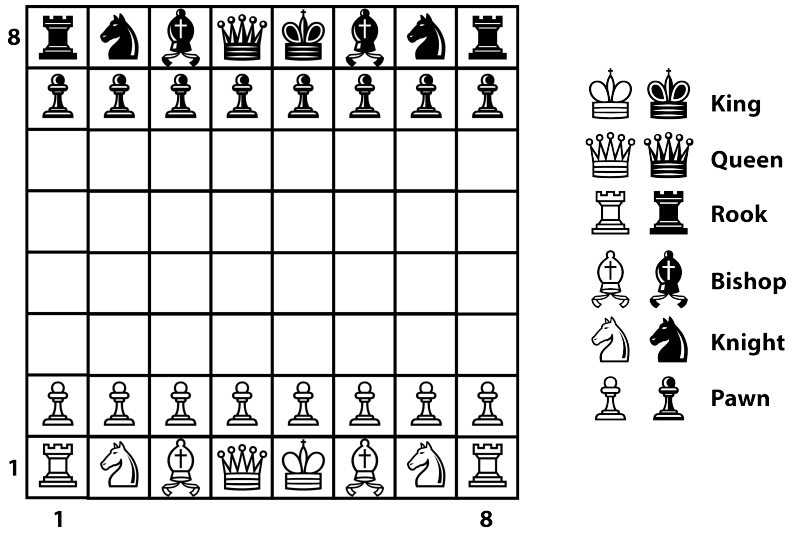

Figure 1: A game of chess and the name of each piece.

Learning to checkmate only with King and Queen

For your assignment you will be given a reduced chessboard with a **4x4 grid** and three pieces that are generated in random locations: **1x King (1)**, **1x Queen (2)** and **1x Opponent's King (3)**. Your task is to use reinforcement learning to teach your pieces (White, King and Queen) to perform checkmate to the Opponent's King (Black). *For illustrating purposes in these examples we will use an 8x8 grid.*

General movement rules

- The King can move 1 block at a time.
- The Queen can move in any one straight direction: forward, backward, right, left, or diagonally, as far as possible as long as she does not move through any of the other pieces on the board.
- None of the pieces is allowed to exit the board.

Refer also to figure 3.

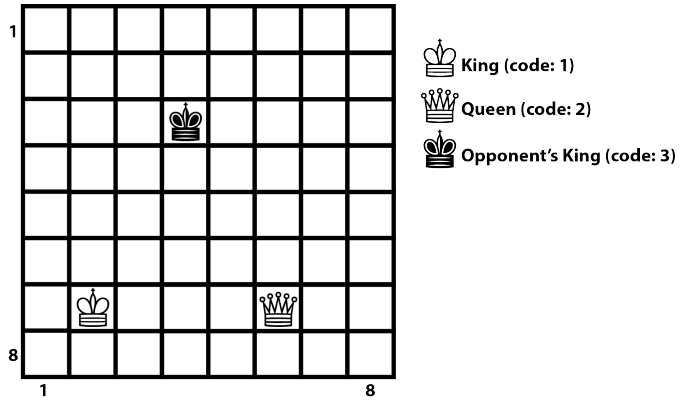

Figure 2: Simplified chess.

Hard-coded movement rules

To simplify the game even more and make the learning process faster some constraints have been added:

- The King will never move to a location that will cause it to be threatened by the Opponent's King.
- The Queen will never move to a location that will cause it to be threatened by the Opponent's King except if the King is nearby to protect.
- The Opponent's King will always pick a random action that will not cause him to be threatened. If such action is not available then:
  - If it is already threatened then it's a **checkmate**.
  - If it is not already threatened then it's a **draw**.

Refer also to figure 4.

End of game conditions and example of a full game

The game can end either in a **checkmate**, where the Opponent's King cannot move and is threatened (refer to figures 5(a) and 6), or in a **draw**, where the Opponent's King cannot move but it is not threatened (refer to figures 5(b) and 7).

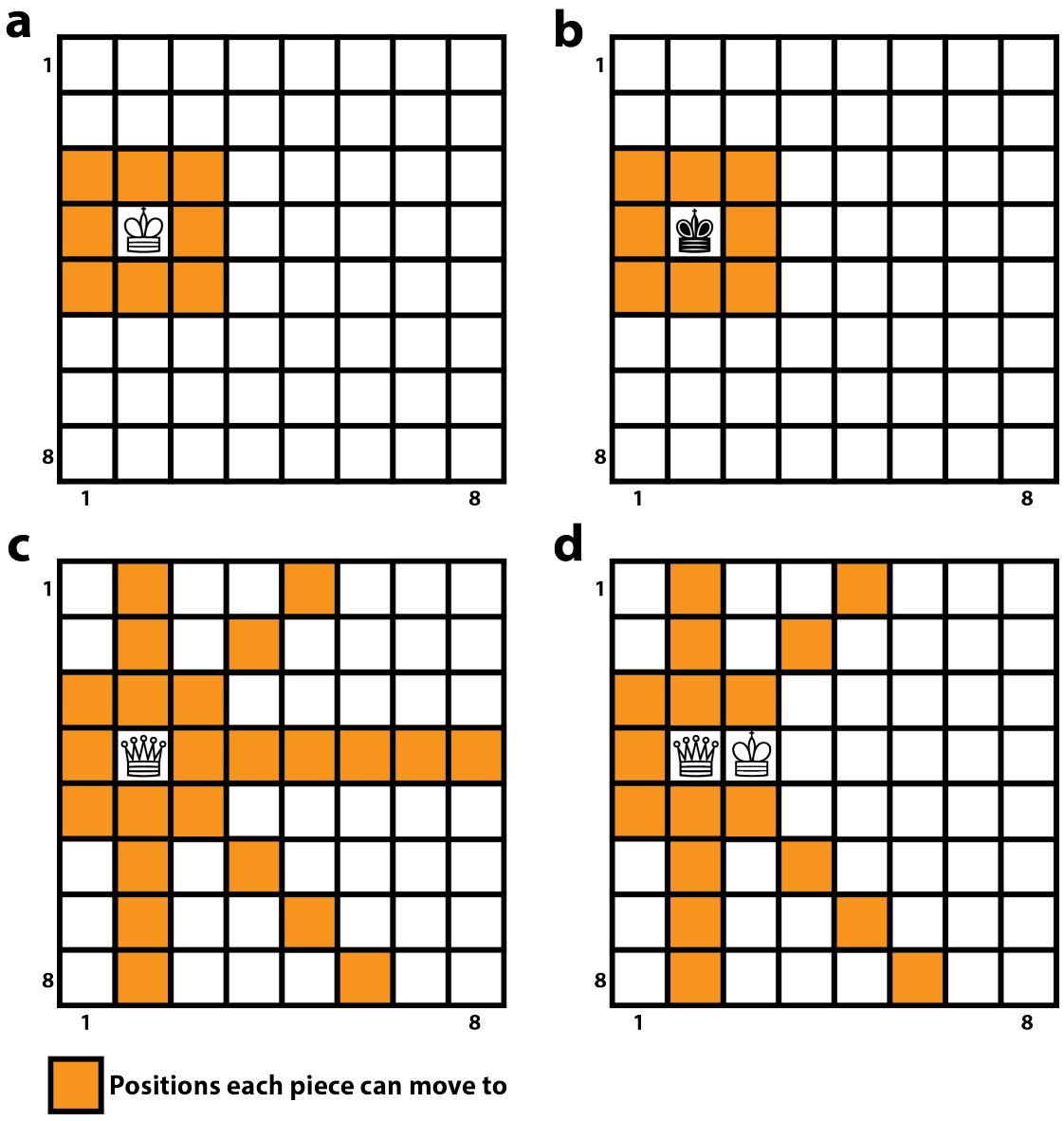

Figure 3: Movement pattern of each piece. **(a)** King, **(b)** Opponent's King, **(c)** Queen, **(d)** Queen blocked by King.

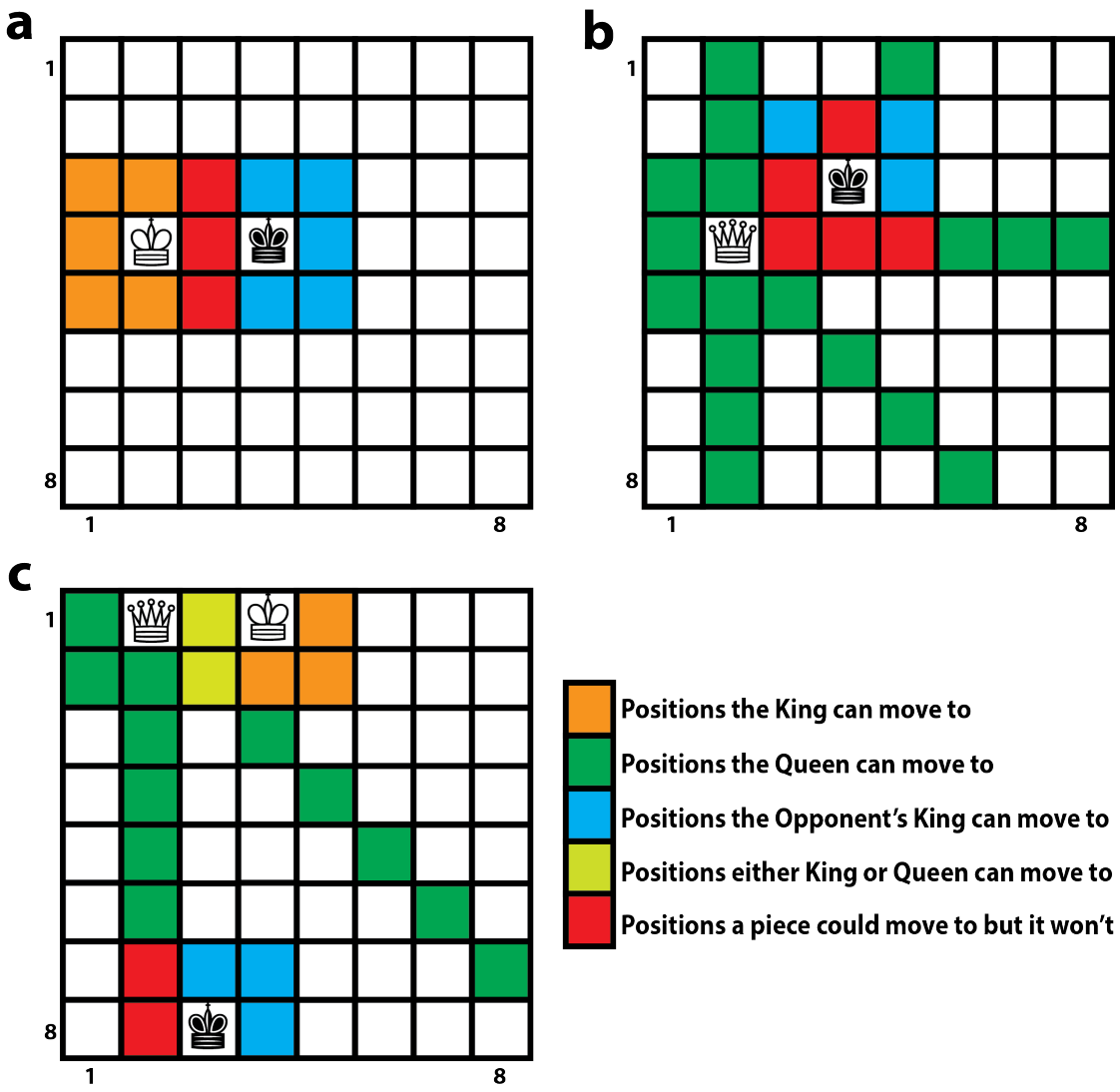

Figure 4: Constrained movement pattern of each piece. **(a)** The King or the Opponent's King will not select to move in the blocks marked with red because on the next turn one of them will be captured. **(b)** The Queen or the Opponent's King will not select to move in the blocks marked with red because on the next turn one of them will be captured. **(c)** The Queen or the Opponent's King will not select to move in the blocks marked with red because on the next turn one of them will be captured.

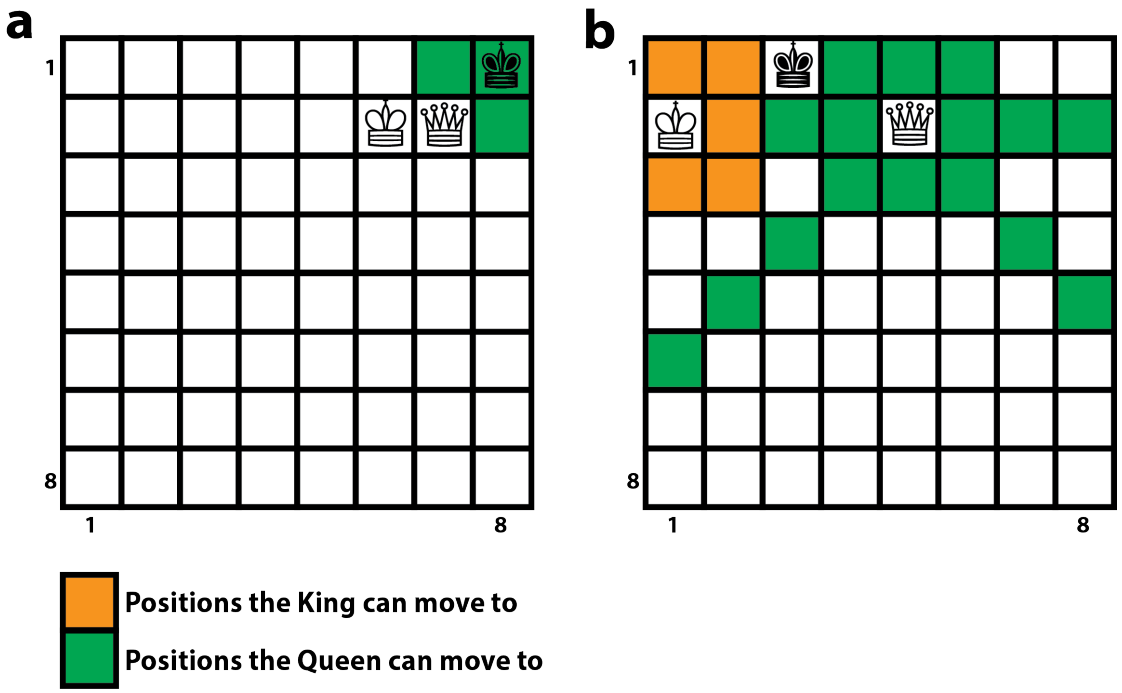

Figure 5: Ending of a game. **(a)** It is the Opponent's King turn, it is threatened by the Queen and it cannot perform any action to escape the threat (it also cannot capture the Queen because then it will be captured by the King). It is a **checkmate**. **(b)** It is the Opponent's King turn, it is not threatened but it cannot move anywhere since all the available positions will cause it to be threatened. It is a **draw**.

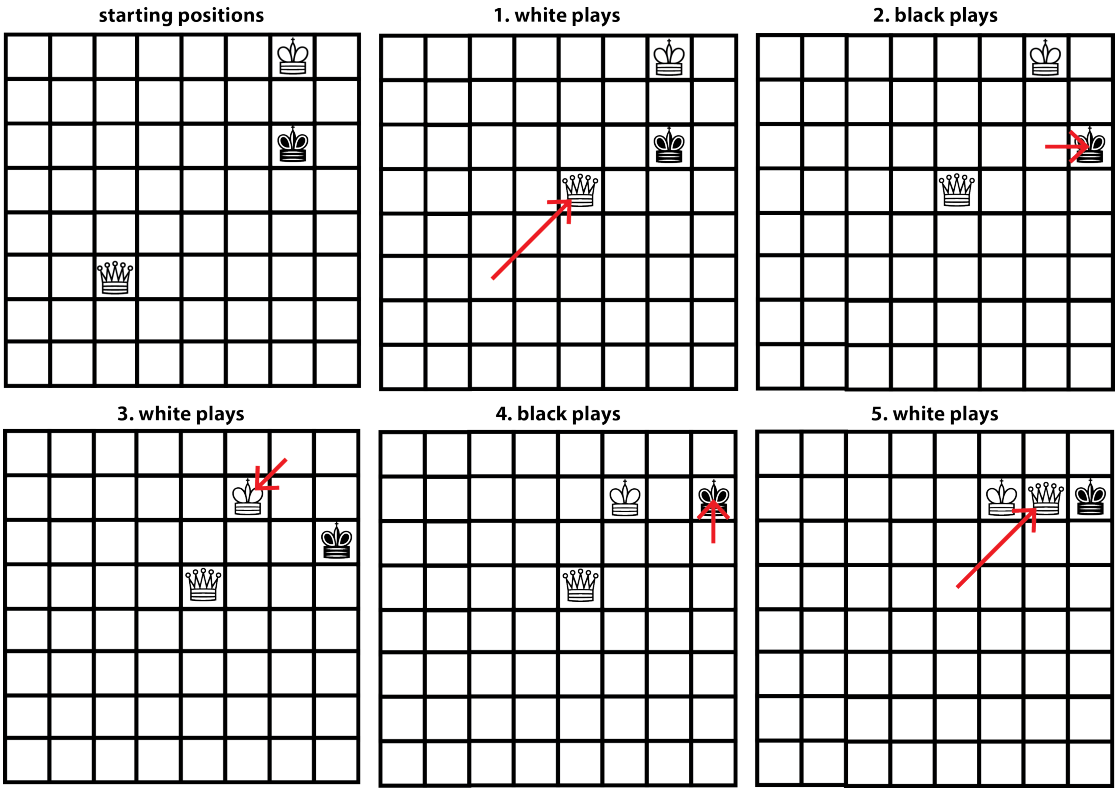

Figure 6: Game ends with a checkmate. The pieces are randomly placed in the chessboard without causing any threats. **(1)** Player 1 (white) plays first and moves the Queen. **(2)** Player 2 (black) moves the Opponent's King to a random position given that it will not be threatened. **(3)** Player 1 (white) moves the King. **(4)** Player 2 (black) moves the Opponent's King. **(5)** Player 1 (white) moves the Queen. The game ends in a **checkmate** because the Opponent's King is threatened by the Queen and it cannot perform any action to escape the threat.

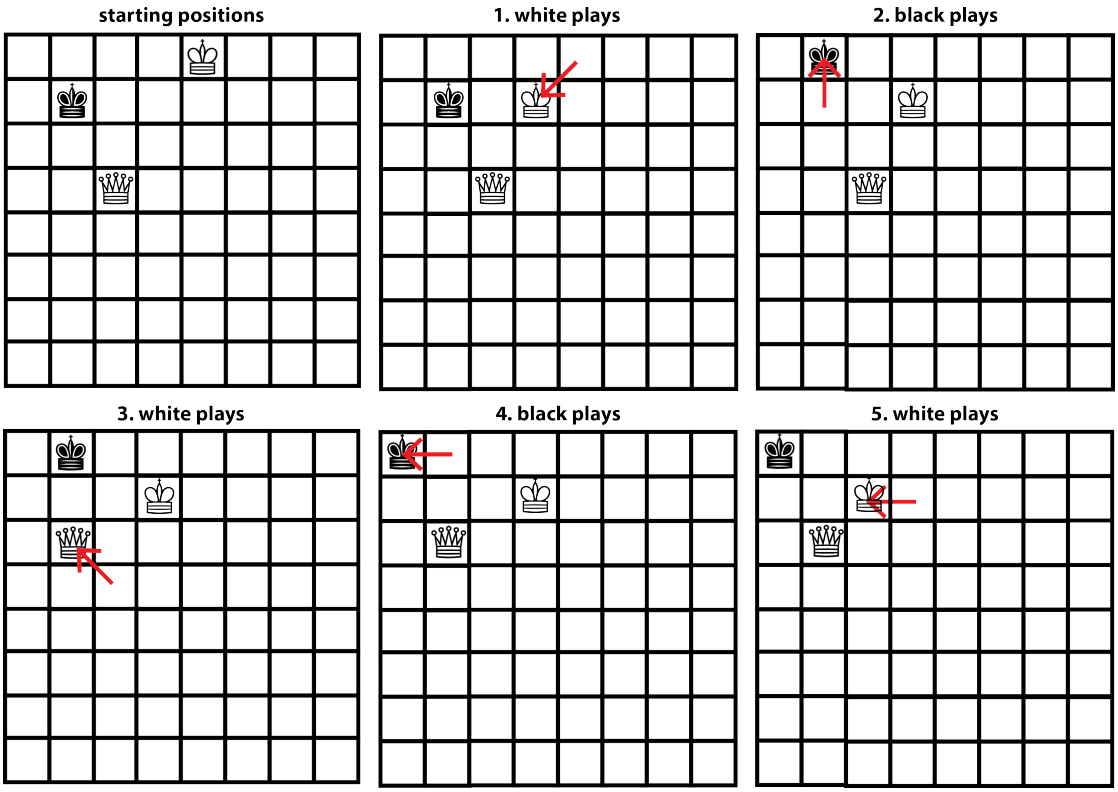

Figure 7: Game ends with a draw. The pieces are randomly placed in the chessboard without causing any threats. **(1)** Player 1 (white) plays first and moves the King. **(2)** Player 2 (black) moves the Opponent's King to a random position given that it will not be threatened. **(3)** Player 1 (white) moves the Queen. **(4)** Player 2 (black) moves the Opponent's King. **(5)** Player 1 (white) moves the King. The game ends in a **draw** because the Opponent's King is not threatened and it cannot perform any action since every available action will cause it to be threatened.