Vamshi Jandhyala

Knights on a Chessboard

✠

November 2020

A white knight and a black knight are situated on diagonally opposite corners of a 3×3 square. In turn, starting with White, they move randomly until (inevitably) Black captures White. What is the expected number of Black moves to achieve capture?

Solution

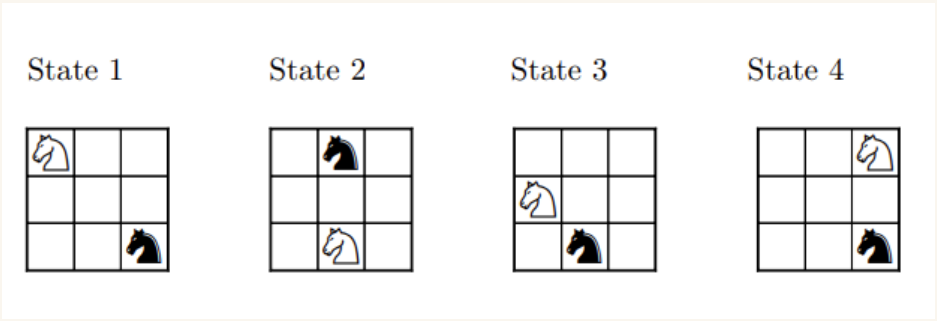
Key observations

1. The complete *state* of the game is determined by the position of the two knights on the board.
2. The next state of the game depends only on the current state.
3. Configurations related by rotation or reflection of the board collapse into a single equivalence class.

This is a *Markov chain* problem.

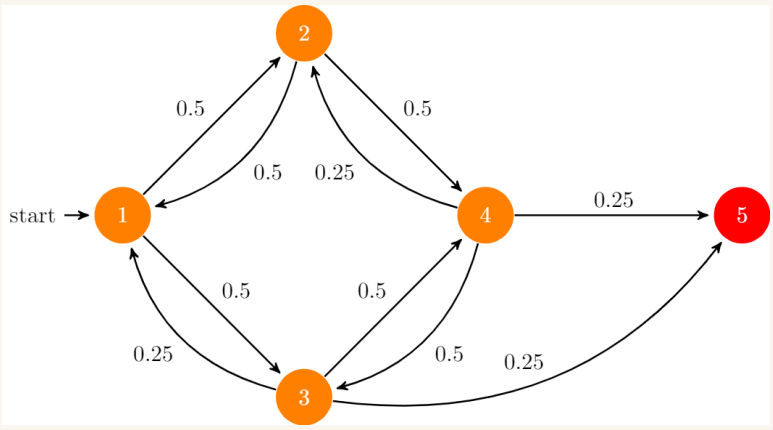
Board states

The diagram below shows all reachable states apart from the absorbing state in which Black captures White. State 1 is the initial state.



State transition matrix

The transition diagram for the game's Markov chain is below. State 5, where the Black knight captures the White, is the absorbing state.



The transition matrix $\mathbf{Q}$ on the four transient states is

$$\mathbf{Q} = \begin{pmatrix} 0 & 0.5 & 0.5 & 0 \\ 0.5 & 0 & 0 & 0.5 \\ 0.25 & 0 & 0 & 0.5 \\ 0 & 0.25 & 0.5 & 0 \end{pmatrix}.$$

Fundamental matrix and expected hitting time

For an absorbing Markov chain with transient block $\mathbf{Q}$, the matrix $\mathbf{N} = (\mathbf{I} - \mathbf{Q})^{-1}$ is the *fundamental matrix*. Let t_i be the expected number of steps before absorption, given that the chain starts in state s_i, and let $\mathbf{t}$ be the column vector whose i-th entry is t_i. Then

$$\mathbf{t} = \mathbf{N}\,\mathbf{c},$$

where $\mathbf{c} = (1,1,1,1)^{\top}$.

Computing,

$$\mathbf{N}\,\mathbf{c} = \begin{pmatrix} 1 & -0.5 & -0.5 & 0 \\ -0.5 & 1 & 0 & -0.5 \\ -0.25 & 0 & 1 & -0.5 \\ 0 & -0.25 & -0.5 & 1 \end{pmatrix}^{-1} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 8 \\ 8 \\ 6 \\ 6 \end{pmatrix}.$$

The expected number of steps to capture, starting from the initial configuration (state 1), is therefore **8**.

Computational Verification

```
from random import choice

transitions = {
    (0,0): [(2,1), (1,2)],
    (0,1): [(2,0), (2,2)],
    (0,2): [(1,0), (2,1)],
    (1,0): [(0,2), (2,2)],
    (1,2): [(0,0), (2,0)],
    (2,0): [(0,1), (1,2)],
    (2,1): [(0,0), (0,2)],
```

```
    (2,2): [(1,0), (0,1)],
}

runs = 100_000
total = 0
for _ in range(runs):
    b, w = (0, 0), (2, 2)
    steps = 0
    while True:
        steps += 1
        w = choice(transitions[w])
        b = choice(transitions[b])
        if b == w:
            break
    total += steps
print(total / runs)
```

The Monte Carlo estimate converges to 8.0, matching the closed form.