Project Proposal: Q-Learning Agent for Copenhagen Hnefatafl (11×11)

Author: Nick Pnevmatikatos

1) Problem to study

I will study how well tabular Q-learning (with careful state abstraction and reward shaping) can learn competent play in an asymmetric, perfect-information board game: Copenhagen Hnefatafl (11×11). The goal is not to reach superhuman strength, but to demonstrate a complete RL pipeline: environment modeling, state/action design, learning, and evaluation.

2) Game characteristics and difficulties

- Asymmetric objectives and piece counts: attackers aim to capture the king; defenders aim to escort the king to a corner (escape).
- Large branching factor: rook-like sliding moves on an 11×11 board yield many legal actions per position.
- Special rules increase tactical depth: restricted squares (throne + corners), king captured from 4 sides (or 3 near the throne), king cannot be captured on the edge, shieldwall captures, exit-fort win condition, and a no-perpetual-repetition rule.
- Sparse terminal rewards: wins/losses occur after long tactical sequences; naive rewards can lead to slow learning or degenerate play.

3) Algorithm and learning setup

Core method: off-policy Q-learning with ε-greedy exploration. Because the raw state space is enormous, I will use a compact, structured representation to keep learning feasible without deep networks.

- State representation (features): piece maps for attackers/defenders/king; king distance-to-corner metrics; mobility counts; local patterns near the king; occupancy of restricted squares; potential shieldwall patterns on edges; turn indicator.
- Action representation: (from_square, to_square) with legality constraints; actions generated on-the-fly to avoid enumerating all 121×121 moves.
- Reward shaping (kept consistent with optimal play): terminal ±1 for win/loss; small step penalty; defender shaping for king progress toward corners/edge safety; attacker shaping for reducing king mobility and forming nets; optional penalties for repetition attempts.
- Training regime: self-play, play against deterministic a-b pruning agents

4) Implementation details

- Programming language: Python + Rust (I found an existing implementation of the game & an a-b agent in rust).
- Environment: existing rules engine implementing Copenhagen Hnefatafl 11×11 (move generation, captures including shieldwall, restricted squares, win conditions, repetition rule).

5) Evaluation plan

- Metrics: win rate as each side, average game length, learning curves (win rate vs. episodes), and robustness vs. unseen opponents.

6) Related work to reference

- Classic temporal-difference learning and tabular RL (Watkins' Q-learning; ε-greedy exploration).
- RL for deterministic board games (general self-play methodology, stability issues, evaluation practices).

7) Connection to the course

This project directly applies reinforcement learning concepts from the course: modeling an MDP, designing a reward signal, balancing exploration/exploitation, and empirical evaluation. It also connects to search/decision-making themes through move generation and adversarial self-play.