Neural Network as an Opponent in Quoridor

Student: Michal Hoľa
Advisor: Peter Gergeľ

1. Brief overview of existing approaches to the production of agents playing board games
2. Program intelligent agent built based on neural networks, which will learn to play Quoridor
3. Test end evaluate the behaviour of the agent

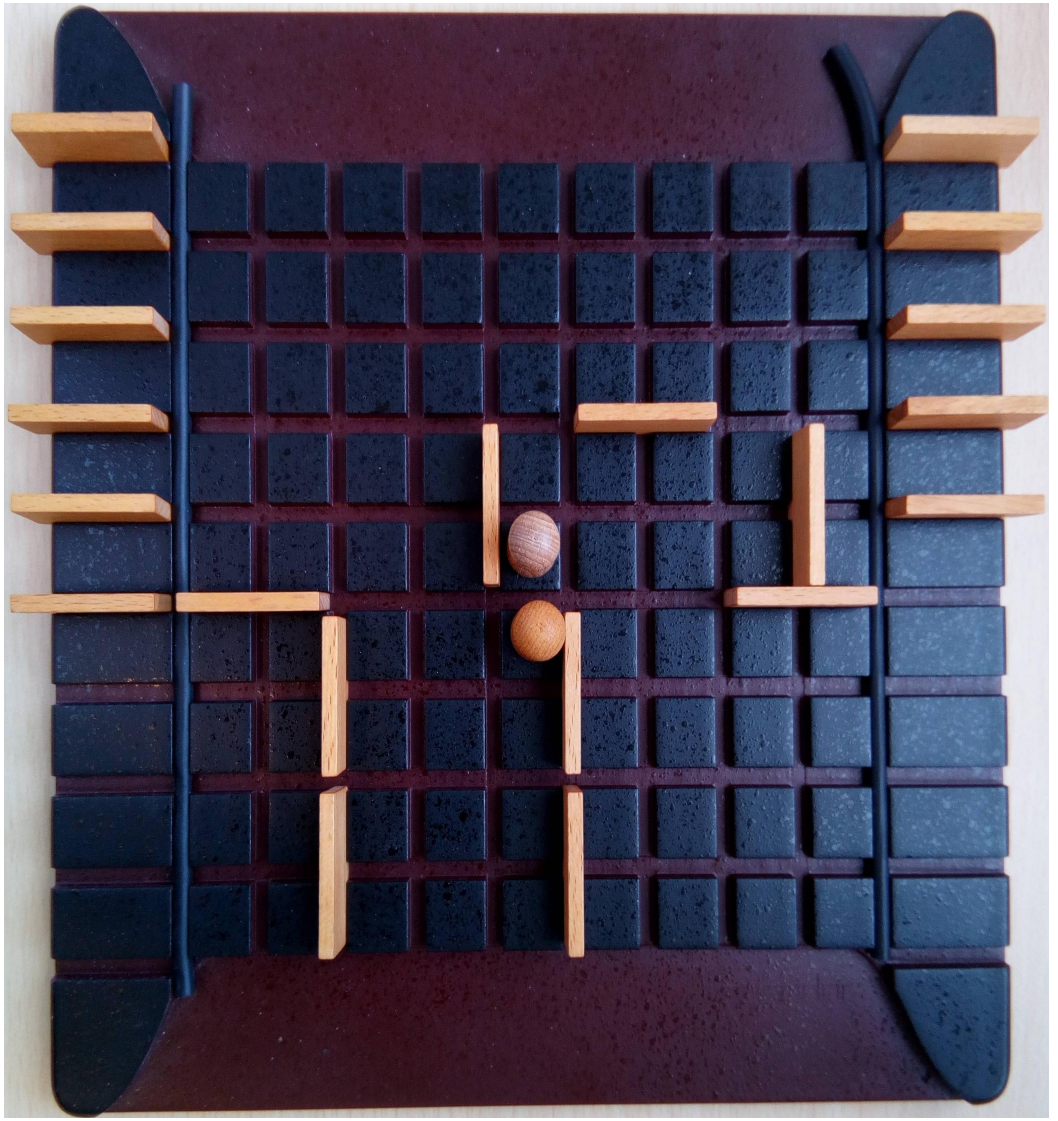
Neural Network as an Opponent in Quoridor

- Mirko Marchesi 1995

- 9X9
- 4 or 2 player version
- 10 walls each

- single pawn, each starts in the middle of the opposite sides

- goal is opposite side



Neural Network as an Opponent in Quoridor

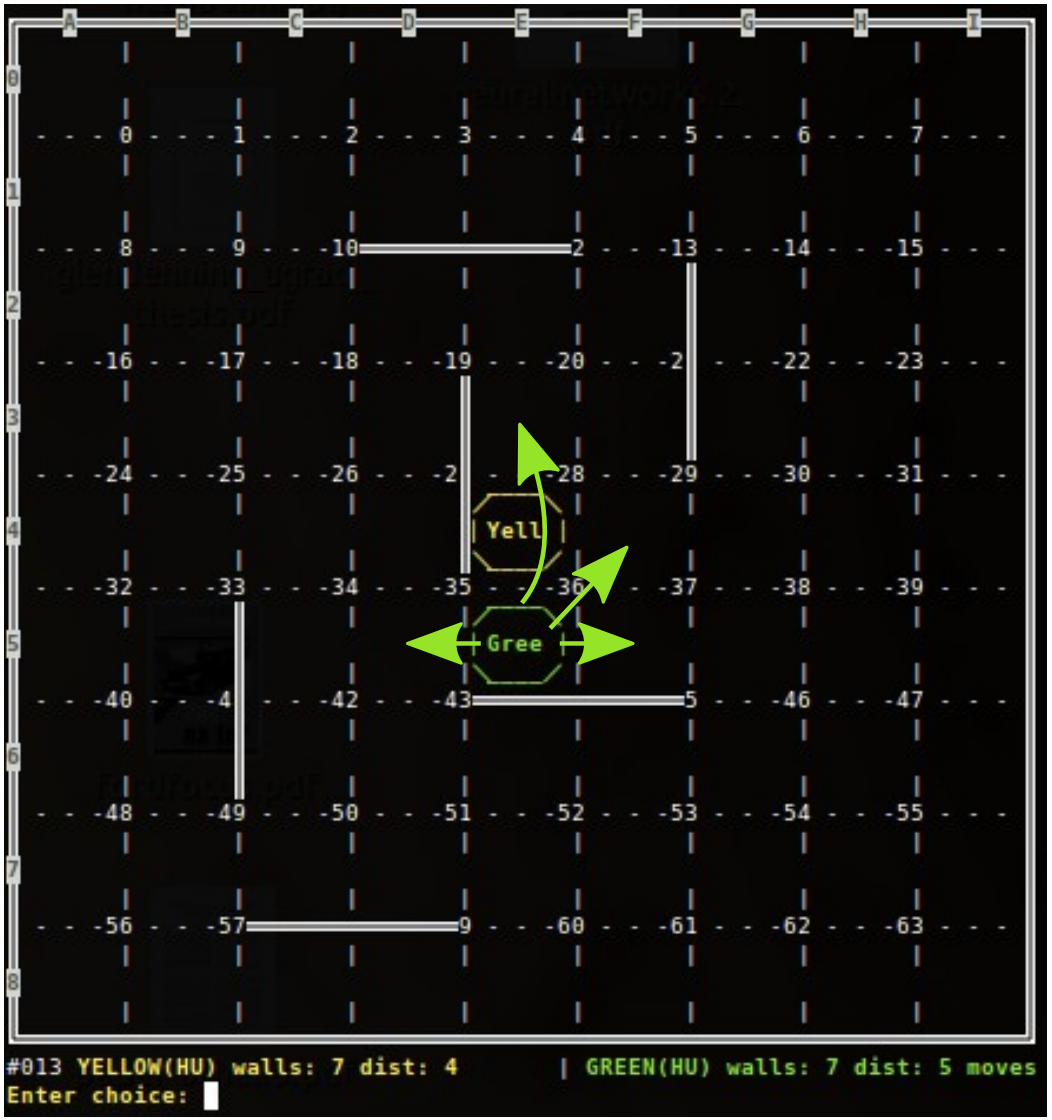
state complexity

P.J.Mertens 2006:

$$S_p = 81 \cdot 80$$

$$S_f = \sum_{i=0}^{20} \prod_{j=0}^{i} (128 - 4i)$$

$$S = S_f \cdot S_p = 3.9905 \cdot 10^{42}$$



Neural Network as an Opponent in Quoridor

branching factor

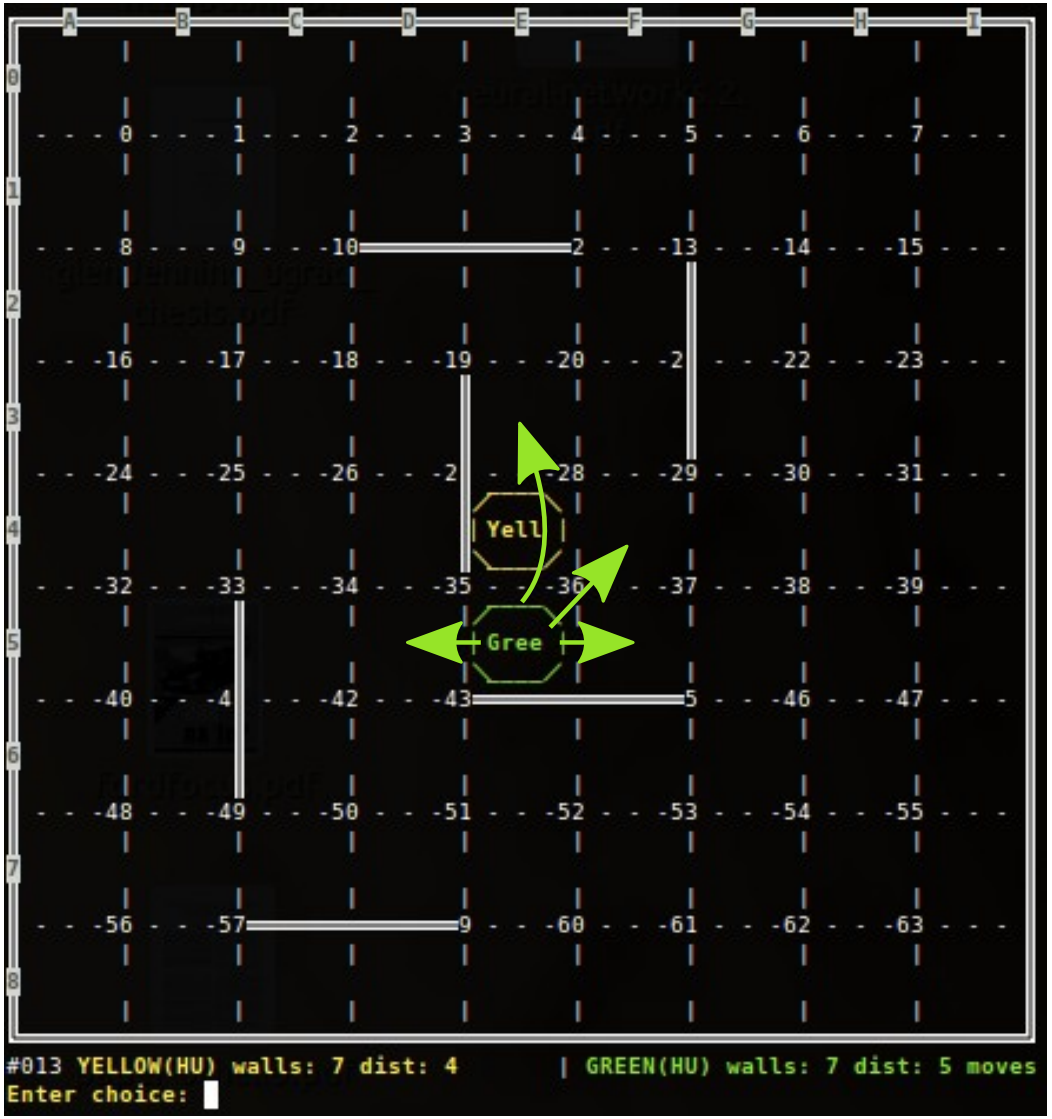
L. Glendennig 2005:

$$b = 60.4$$

game tree complexity

P.J.Mertens 2006:

$$G = 60.4^{91.1} = 1.7884^{10162}$$



Neural Network as an Opponent in Quoridor

State complexity

$$S = 2 \cdot S_p \cdot S_f = \mathbf{2.2775 \cdot 10^{27}} \neq 3.9905 \cdot 10^{42}$$

Branching factor

$$b = \frac{D}{S_d} = \frac{1.0766 \cdot 10^{28}}{5.6379 \cdot 10^{26}} = \mathbf{19.0974} \neq 60.4$$

Game tree complexity

$$C = 19.0974^{91.1} = \mathbf{4.9758 \cdot 10^{116}} \neq 1.7884 \cdot 10^{162}$$

Neural Network as an Opponent in Quoridor

Reinforcement Learning, Q-Learning

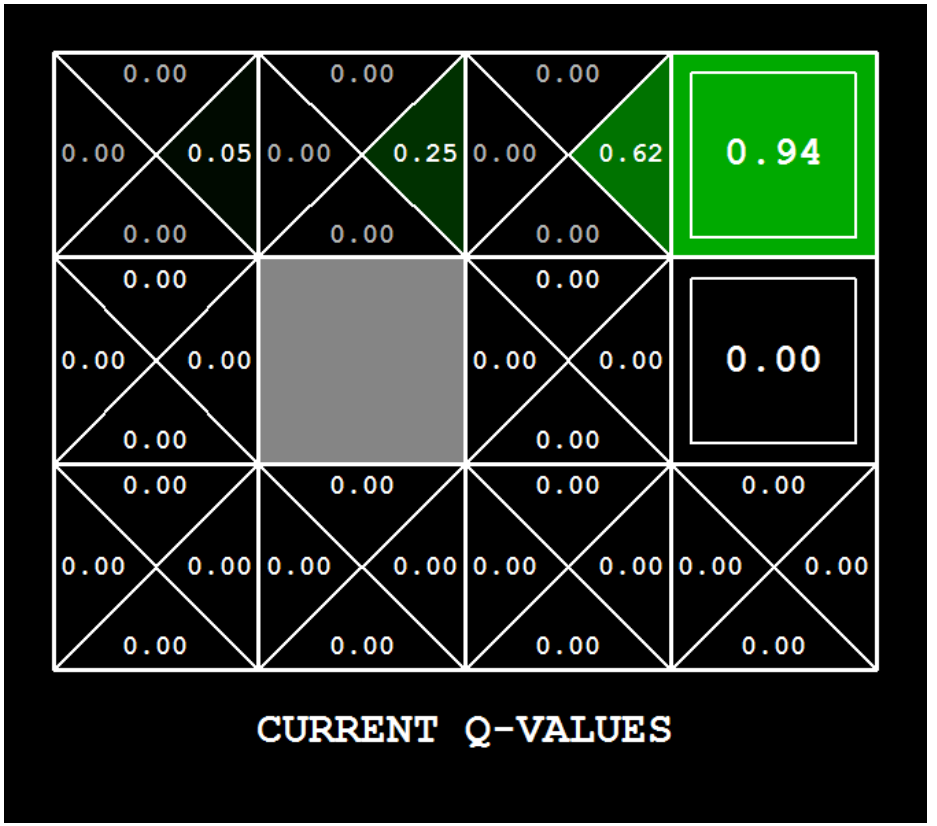
$$\forall t : Q_t(s,a) \rightarrow q$$

$$t \in N,\ q \in R$$

t … time in game (move)
q … q-value goodness of action

Too many evaluations for Quoridor

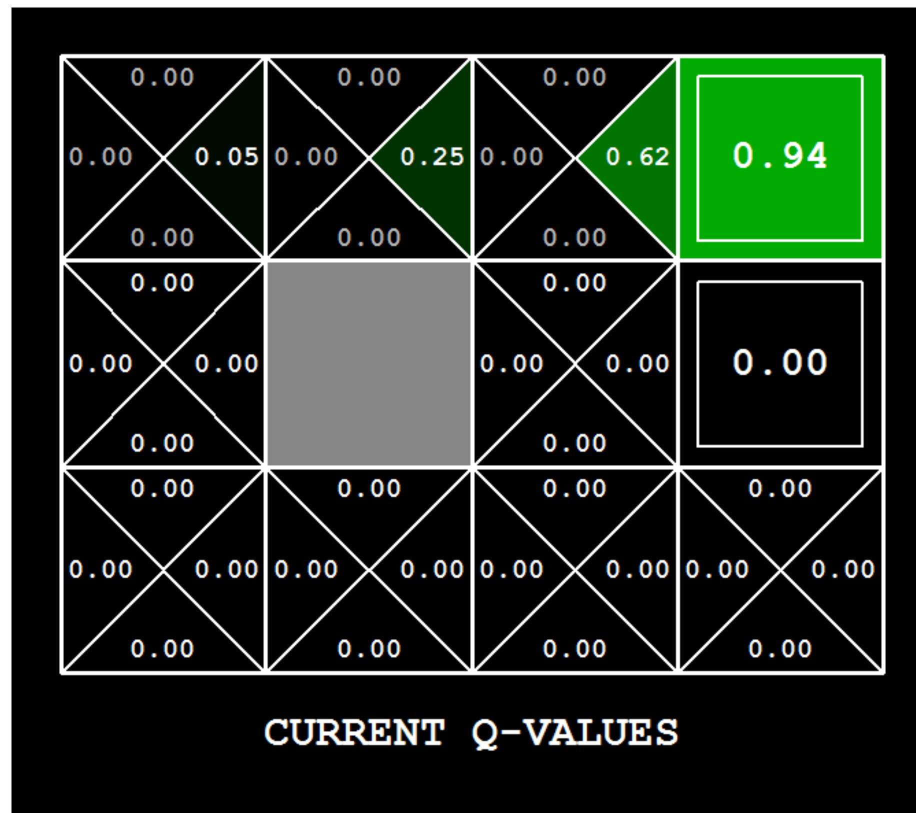
$$7.2172 \cdot 10^{13}\ years$$



Neural Network as an Opponent in Quoridor

Neural Network (NN) learns to estimate Q-value. NN's ability to generalize should help propagate Q-values faster to the early stage of the game.

Inspired by work Michiel van der Ree and col. 2013 for game othello, University of Groningen, The Netherlands



Alpha is 1, NN handles learning rate

$$\hat{Q}^{new}(s_t, a_t) \leftarrow r_t + \max_a \hat{Q}(s_{t+1}, a)$$

1. observe s_t
2. estimate $Q(s_t, a_i)\ \forall i$ using NN
3. choose min/max a_t
4. play a_t
5. estimate $Q(s_{t+1}, a_i)\ \forall i$ using NN
6. choose max/min a_{t+1}
7. back propagate $\Delta \mathbf{W}$ = (reward + $Q(s_{t+1}, a_{t+1})$)

Neural Network as an Opponent in Quoridor

To speed up the process, I have created 'Heuristic' player following shortest path to goal or placing wall on the shortest path of the opponent.

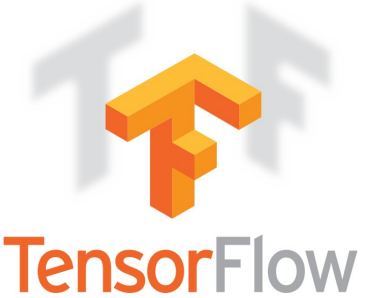
- State

$$S = (c, p_y, p_g, s_y, s_g, w)$$

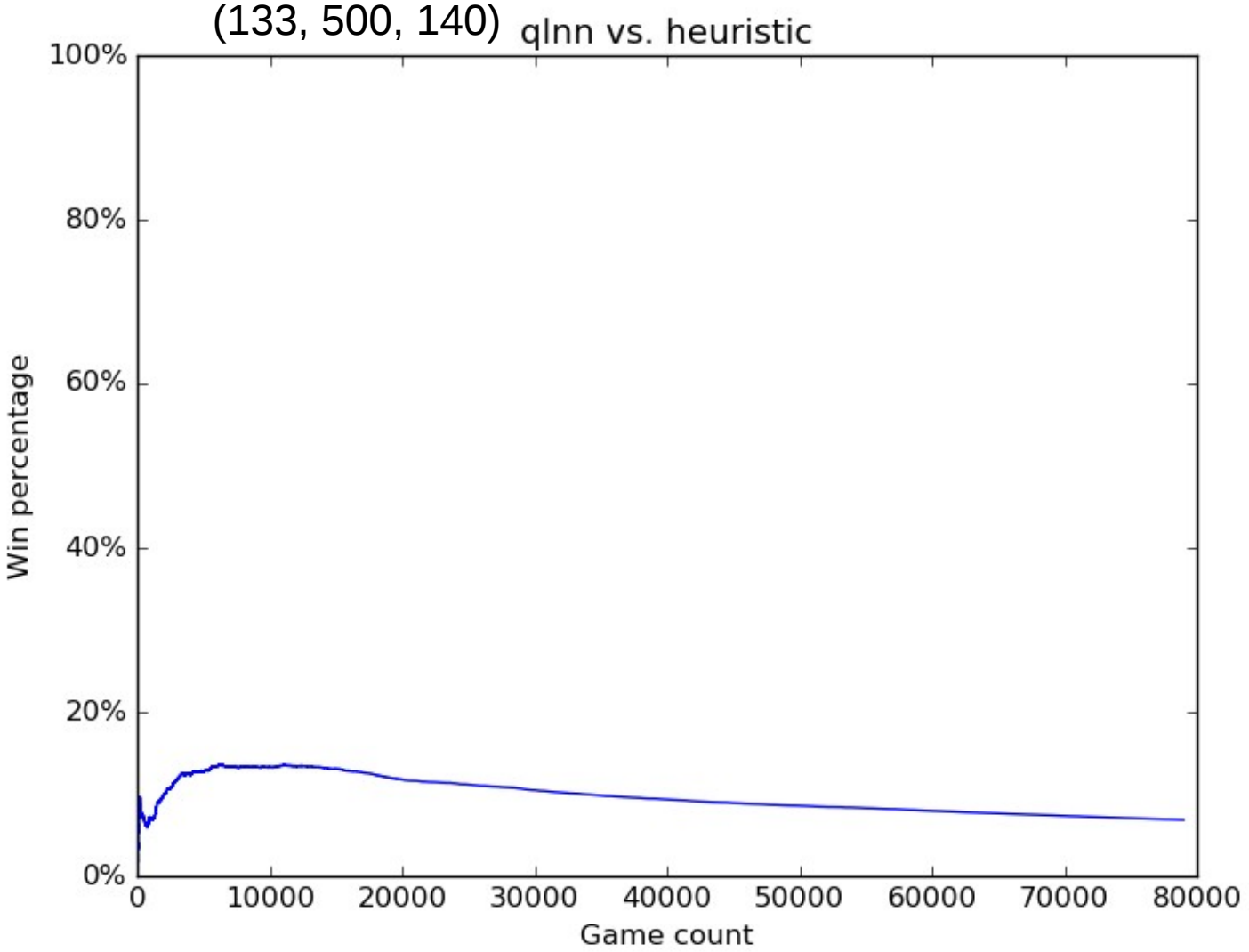
- Context

  - shortest path, blockers, crossers

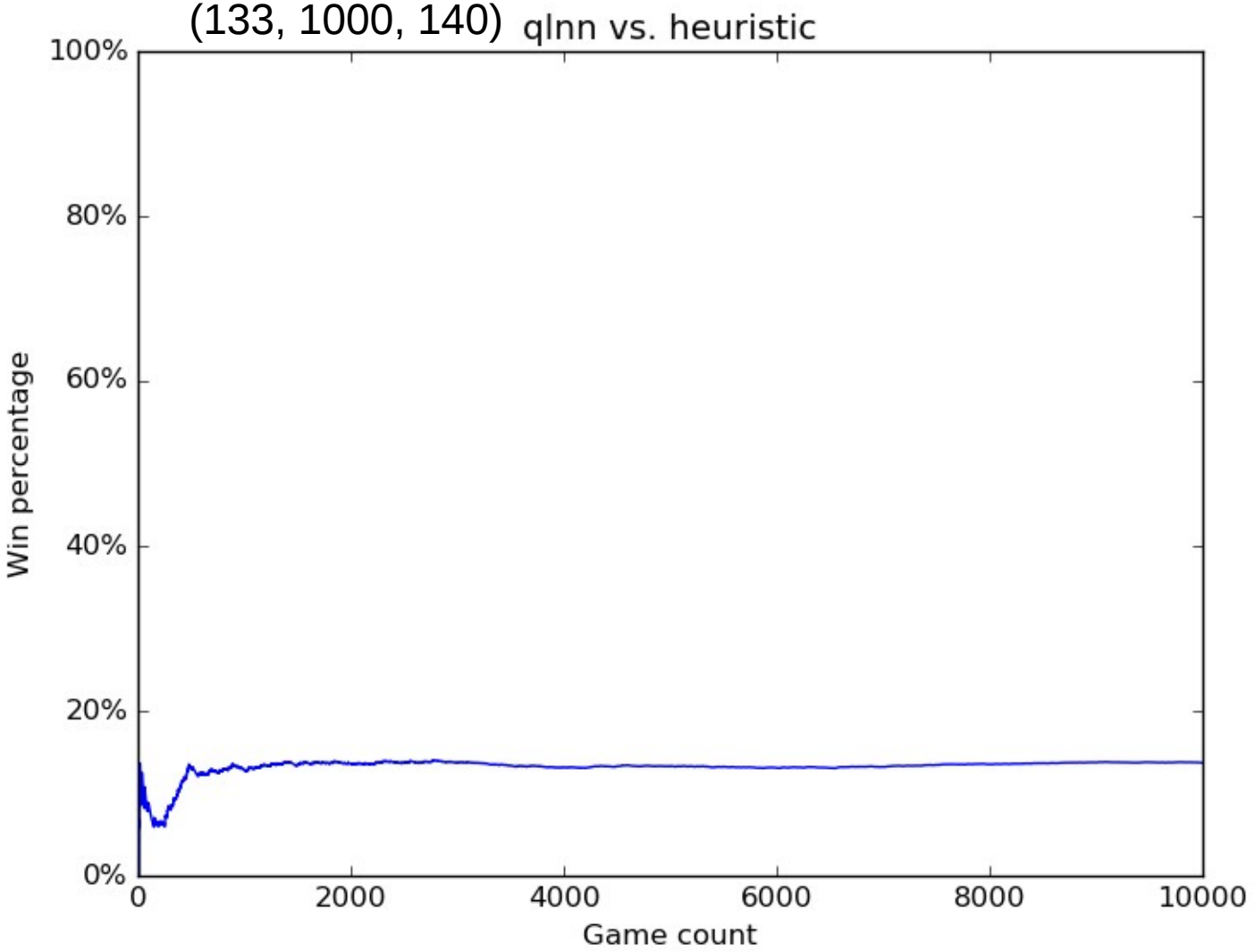
- Tensorflow – python GIL, GPU



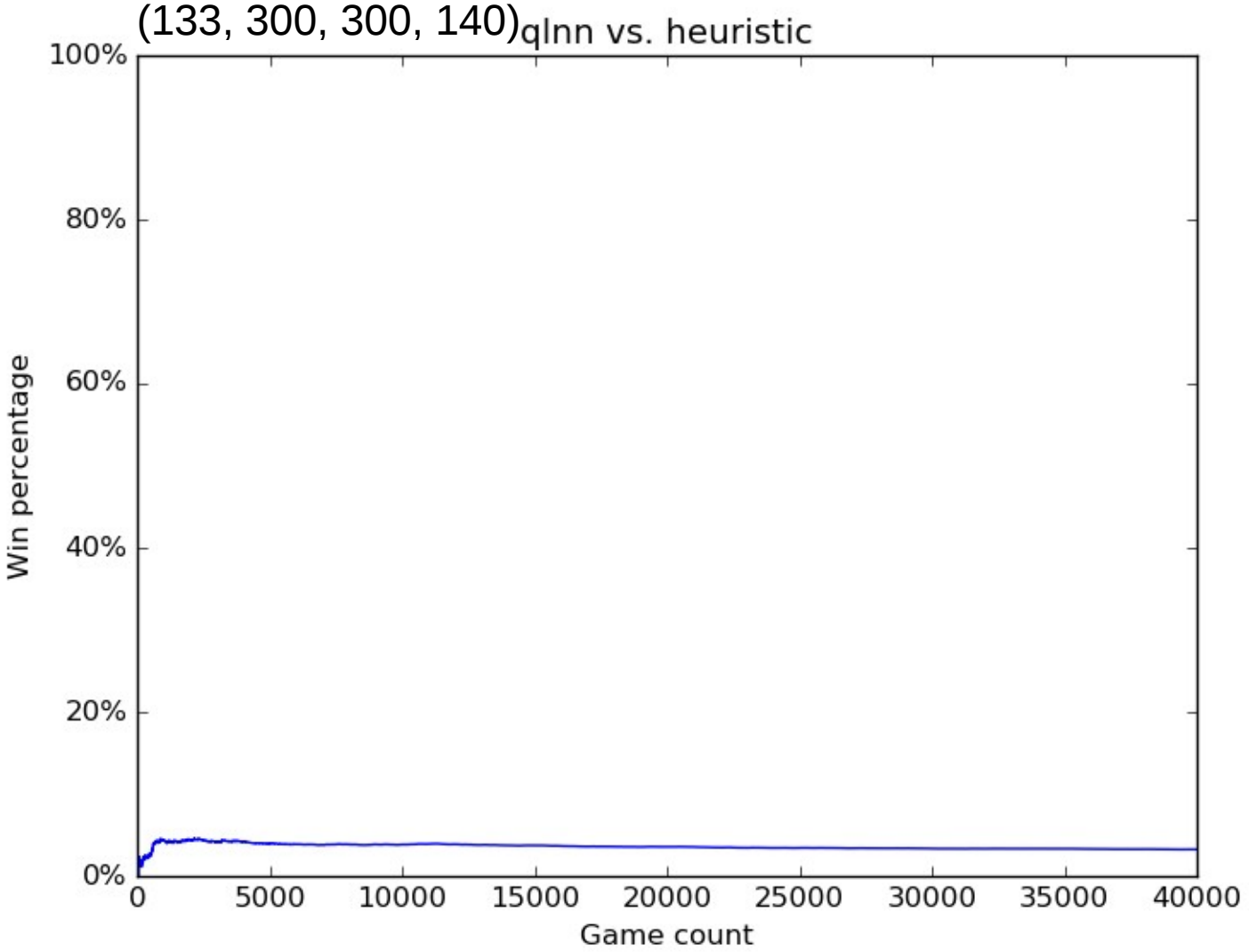
Neural Network as an Opponent in Quoridor



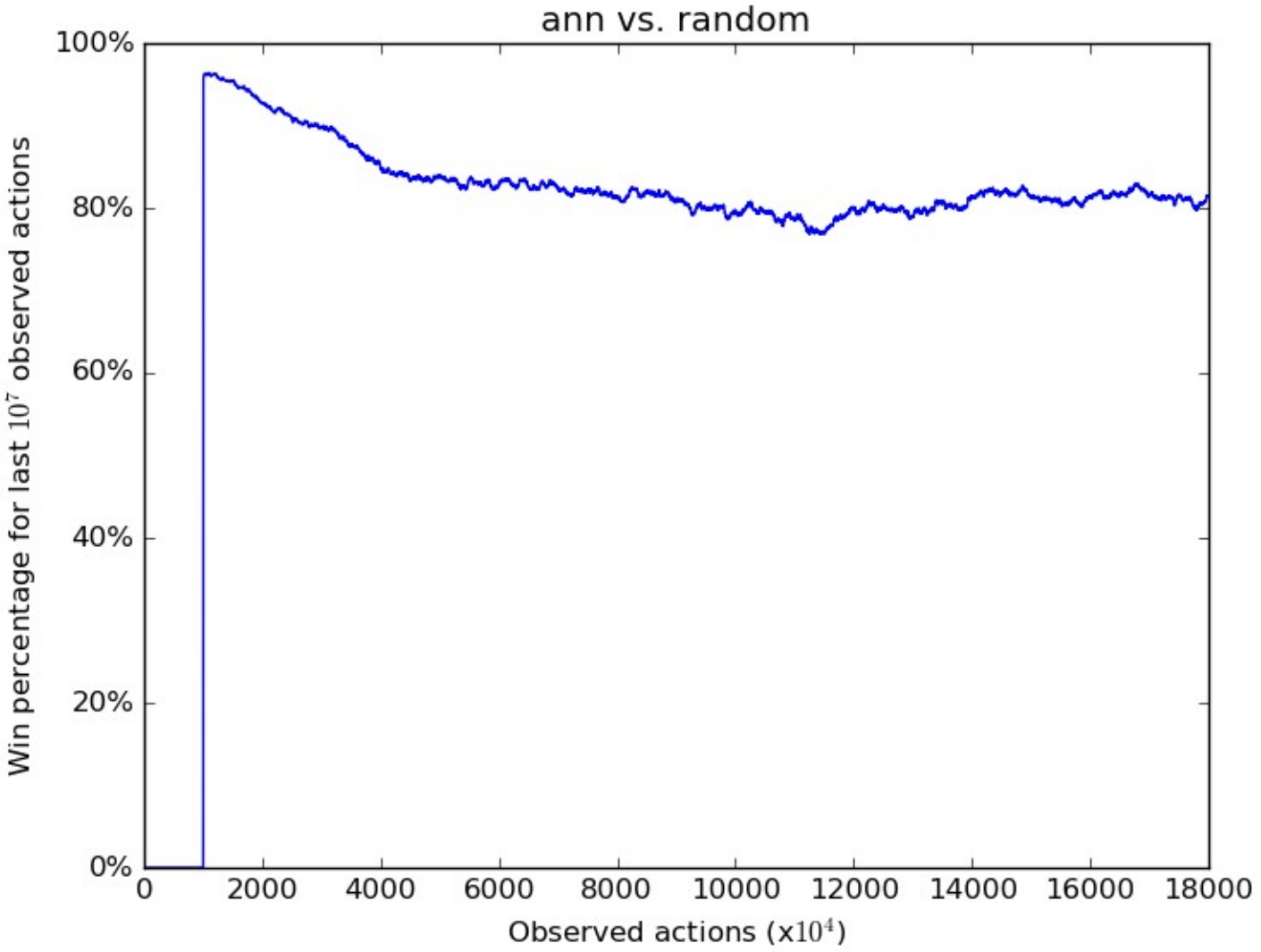
Neural Network as an Opponent in Quoridor



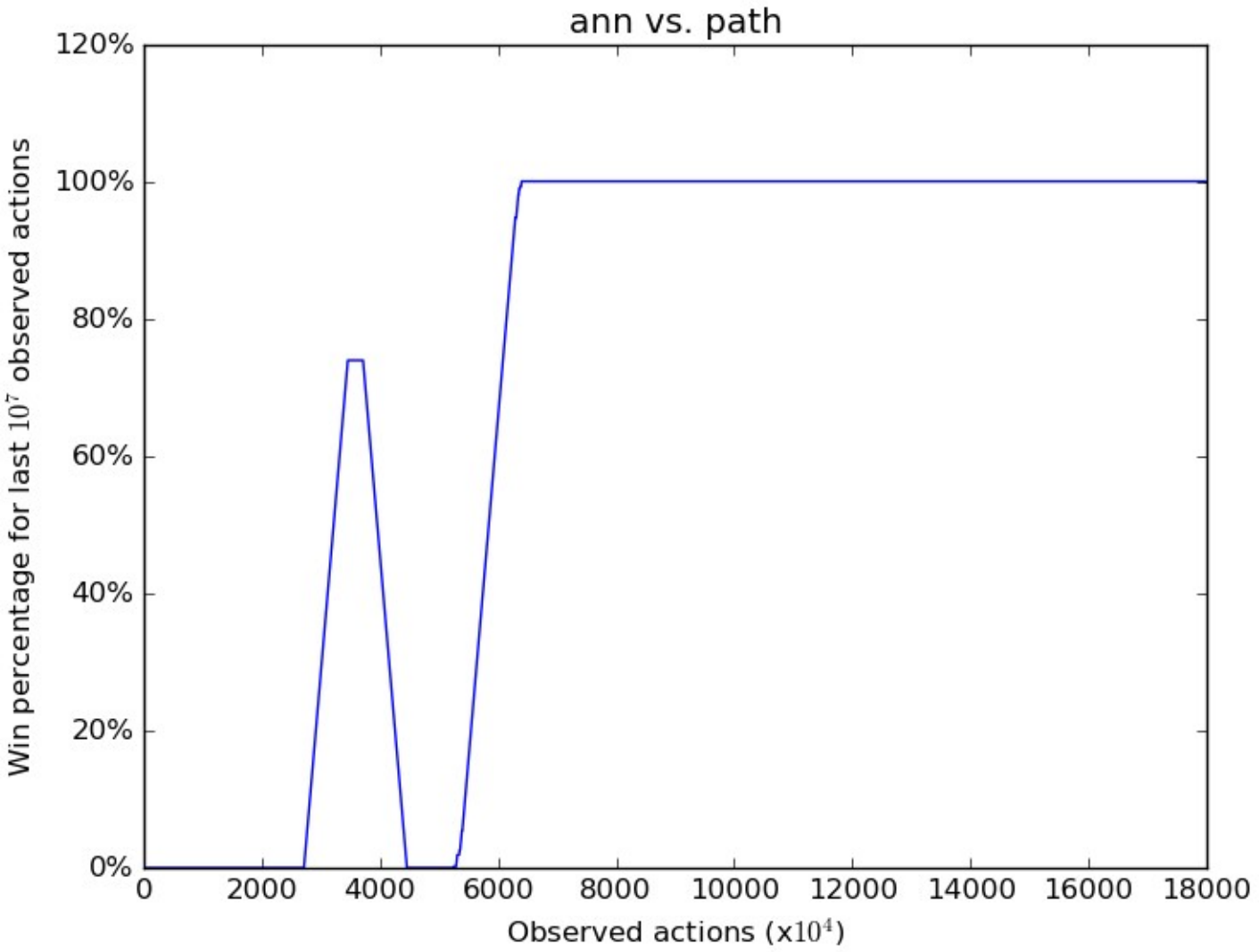
Neural Network as an Opponent in Quoridor



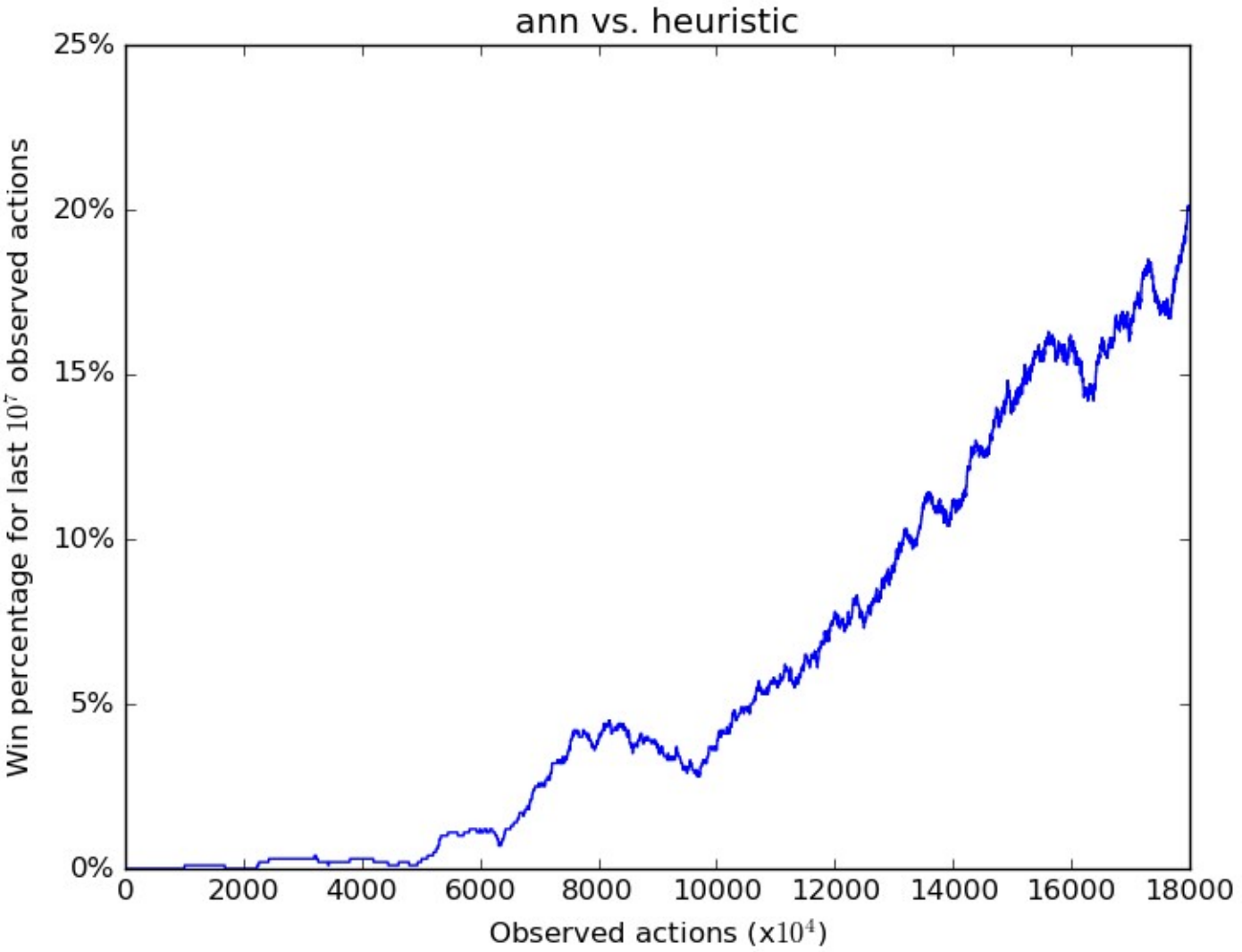
Neural Network as an Opponent in Quoridor



Neural Network as an Opponent in Quoridor



Neural Network as an Opponent in Quoridor



Neural Network as an Opponent in Quoridor

- New state complexity, branching factor and game tree compexity
- Heuristic player
- Q-values estimated by Artificial Neural Network
- Signs of improvement only by watching Heuristic player

Thank you for your attention