Focused Depth-first Proof Number Search using Convolutional Neural Networks for the Game of Hex

Chao Gao, Martin Müller, Ryan Hayward

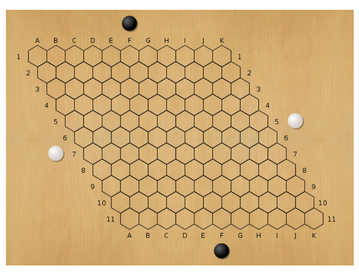
University of Alberta, Department of Computing Science, Canada

cgao3@ualberta.ca

Abstract

The game of Hex is a two-player, perfect information and zero-sum board games where no draws exist. Solving the game of Hex can be formulated as searching for a solution in an AND/OR graph. Proof Number Search is an effective algorithm for establishing theoretic values on games with non-uniform branching factor. We investigate how to use strength Convolutional Neural Networks (CNNs) to improve proof number search. We describe FDFPN-CNN to address the uniform-branching factor problem in Hex. FDFPN-CNN integrates two CNNs trained from games played by expert players. The value approximation CNN provides reliable information for defining the widening size by estimating the value of the expanding nodes, while the policy CNN selects promising children nodes to the search. We show that, on 8×8 Hex, FDFPN-CNN performs much better than the original FDFPN.

Game of Hex



The game of Hex is invented in 1942 by Piet Hein and in 1948 independently by John Nash. It has been an active domain of artificial intelligence since Shannon's work in 1950s.

The most popular board size is 11×11 (as shown left).

Facts we know about Hex:

- By strategy-stealing argument, there is a winning strategy for first player on initial board position.
- The explicit strategy is unknown.
- Solving arbitrary Hex position is PSPACE-Complete.

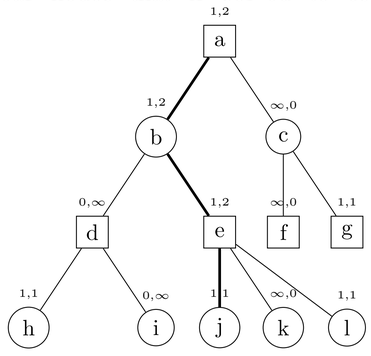
Automatic computer solver requires efficient tree search. Solving all openings is of particular interest since Hex is usually played with *swap* rule.

Board sizes solved:

- 7×7 Hex is solved in 2004 by depth-first search + a number of knowledge pruning.
- 8×8 first solved in 2008 by improved H-search + improved knowledge pruning + depth-first search. In 2011, it is shown that Focused depth-first proof number search is twice faster than DFS.
- 9×9 solved in 2014 by parallel FDFPN, took > 17 months.

Focused depth-first proof number search with CNNs

Proof/disproof number of a node is the minimum number of leaf nodes that if solved to establish the value of this node.



In trees, those numbers can be calculated from bottom up.

- $\phi(s) = \min_{s' \in children(s)} \delta(s')$
- $\delta(s) = \sum_{s' \in children(s)} \phi(s')$

$\phi(s)$ and $\delta(s)$ are respectively the minimum number of leaf nodes to prove or disprove s. Following ϕ, δ, there exists most proving node, which potentially would lead to the least effort to solve the root.

Proof number search works in an iterative fashion:

1. Select an Most Proving Node (MPN)
2. Expand MPN, set children's proof and disproof number
3. Backup

Depth-first proof number search (DFPN) reformulates on PNS, employing bounds to avoid unnecessary traversal in the tree. It is more efficient than PNS.

PNS or DFPN does not work well when the game has near uniform branching factor. Focused DFPN with CNNs is to address the near-uniform branching factor in Hex. FDFPN-CNN is redesigned upon the original FDFPN, at each expanding node:

- Resistance for move sorting $\Rightarrow$ A policy net for sorting moves;
- Fixed widening factor $\mu \Rightarrow$ A value neural net is applied to dynamically define a widening factor.

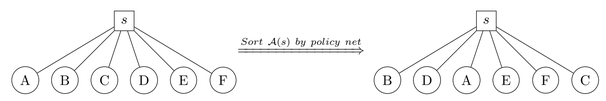


Figure 1: A policy neural net is used to order the children of an expanding node.

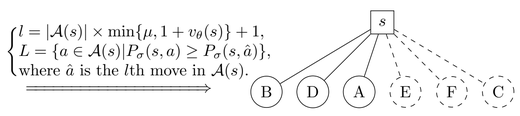


Figure 2: A value neural net is used to decide window size.

The value net helps create an "artificial" non-uniform branching factor, making proof number search prefers nodes with small value estimation.

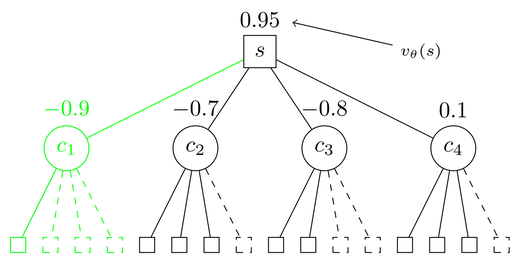


Figure 3: The smaller value estimation, the smaller expanding window.

Policy and value net

DATA: 6.5×10^5 state-action or state-value pairs produced by MoHex vs MoHex or MoHex vs Wolve play.

Architectures

Input feature planes consist of black, white, empty, black-bridge endpoints, white-bridge endpoints.

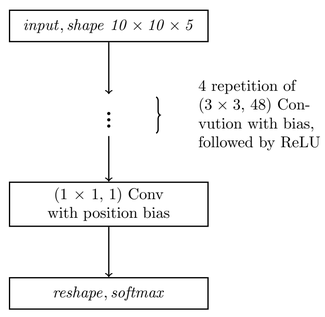


Figure 4: Policy neural net architecture

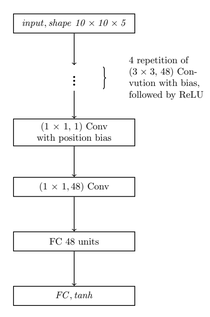


Figure 5: Value net architecture.

Accuracies of the neural nets

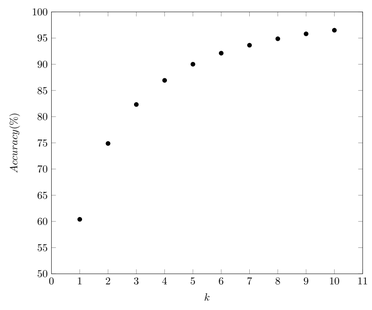


Figure 6: Prediction accuracies of the policy net.

MSE of value net:

- Train: 0.067
- Test: 0.083

Empirical performance

Using fixed widening factor for all expanding nodes, could FDFPN perform better by replacing its move ordering function from Resistance to policy net?

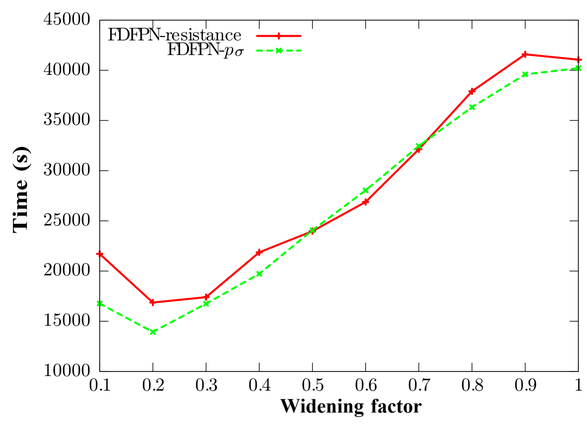


Figure 7: Policy neural net is better at ranking strong moves than Resistance.

- Policy net is better at small winder factor.
- Widening factor is close to 1, both recede to normal DFPN.

Solving all 8×8 openings

Improvement of FDFPN with both policy and value nets (compared to original FDFPN):

- #expansion: 46.7%
- #time: 40% [1]

Further investigation of the value net

In the discovered solution graph, for each node, compare the estimated value with ground truth.

A simple classifier can be defined:

$$f(s) = \begin{cases} 1, & \text{if } v_\theta(s) > 0 \\ 0, & \text{if } v_\theta(s) \leq 0 \end{cases}$$

Error rate of f: 14.3%.

Conclusions

The better performance of FDFPN-CNN is due to two reasons:

1. Better move selection using policy net,
2. Creating "artificial" non-uniform branching factor with value net.

FDFPN-CNN shows that incorporating learnt knowledge to proof number search is possible. It is a promising direction since it is easy to generate **training data** from strong players.

Forthcoming Research

Further improvement is quite possible.

1. Better neural network design. Perhaps residual neural net. More regularization...
2. Dealing with the imperfectness of the training data.
3. Exploiting AND/OR structure to improve the value estimation accuracy.
4. Modify the calculation/initialization of proof and disproof number with value net.
5. Or define a new search paradigm other than proof number search.

[1] Due to the update of Tensorflow, the program gets a bit faster