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

Chao Gao, Martin Müller, Ryan Hayward

Department of Computing Science,
University of Alberta, Edmonton, Canada

IJCAI August 23, 2017

Contents

Introduction: Game of Hex

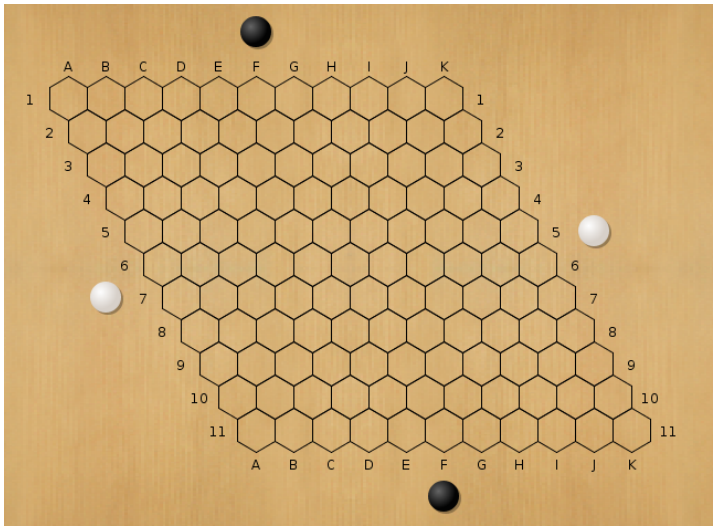


Figure: Hex is played on rhombus board.

- Invented in 1942, and in 1948 independently by John Nash.
- Played on rhombus board, 11×11 most popular.
- Two-player, perfect information, and zero-sum.
- No draws.

Introduction: Algorithmic Solving Hex

Facts we already know.

1. By strategy stealing, there is a wining strategy for first player from empty board.
2. The explicit strategy is unknown.
3. Solving arbitrary position is PSPACE-complete (Reisch 1981).

How to efficiently solve Hex positions by search?

Search Virtual Connections by Hierarchical Algebra (Anshelevich 2000), but

H-Search is incomplete.

Needs Tree Search.

Introduction: Algorithmic Solving Hex

Tree search to solve Hex,

- Inferior Cells Analysis can be helpful for pruning
- H-Search helps detect early win or lose

With these techniques + Depth-first search

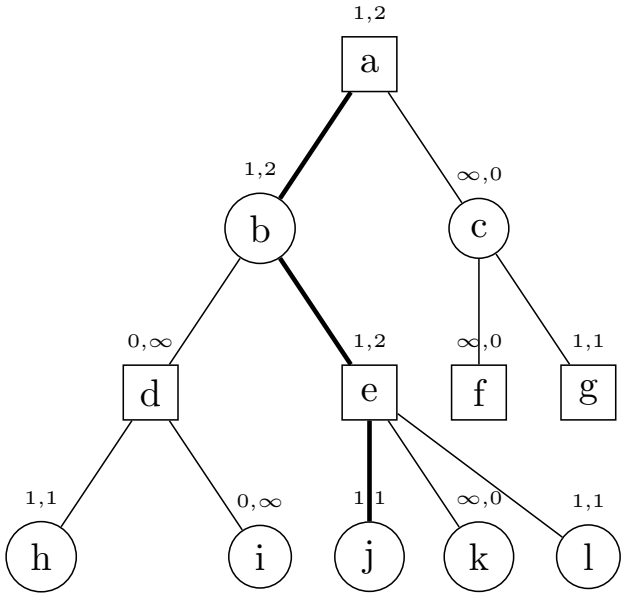
- 7x7 Hex is solved in 2004 (Hayward, Bjornsson, Johnson, Kan, Po and Rijswijck),
- 8x8 Hex is solved in 2008 (Henderson & Hayward).

Solving even larger board sizes is difficult,

Better tree search algorithms?

Review Proof Number Search

Proof number search (Allis et al. 1994).



$\phi(a) = \min\{\delta(b), \delta(c)\}$
$\delta(a) = \phi(b) + \phi(c)$
$\phi(s)/\delta(s)$ is the minimum number of leaf nodes to prove/disprove s.

Following ϕ, δ, there exists most promising node to expand. (requires the least effort to solve the root!)

PNS is best-first, works in an iterative fashion:

1. Select an MPN
2. Expand MPN, set children's proof and disproof number
3. Backup

Depth-first Proof number search

Problems of PNS

- Needs to save the whole tree
- Unnecessary traversal of the tree even MPN does not change

Alleviation: Depth-first PNS (Nagai, 2002):

- Use of bounds to avoid unnecessary traversal
- Use of fixed size transposition table (TT)

Is the algorithm still complete with fixed size TT?

No, but usually works fine if not #nodes $\gg$ TT.

Focused depth-first proof number search

Rethink of the definition of proof and disproof number:

- Shape of the tree matters, PNS exploits narrow and deep branches.

Hex has near "uniform branching factor".

Solution: FDFPN
focus on promising nodes (Henderson 2010).

- Resistance as move ordering
- Expands only $\mu \cdot \mathcal{A}(s)$ nodes.

Twice faster than DFPN in 8×8 Hex.

9×9 Hex openings are solved by parallel FDFPN (Pawlewicz & Hayward 2014), took $>$ 17 months.

Can we make FDFPN better?

Rethink the original FDFPN

Problems of the original FDFPN (Henderson 2010).

- Resistance often pathological
- "Artificial branching factor" is created in an uninformative manner.

Redefine Focused DFPN..
with the help of neural nets..

Redesign FDFPN

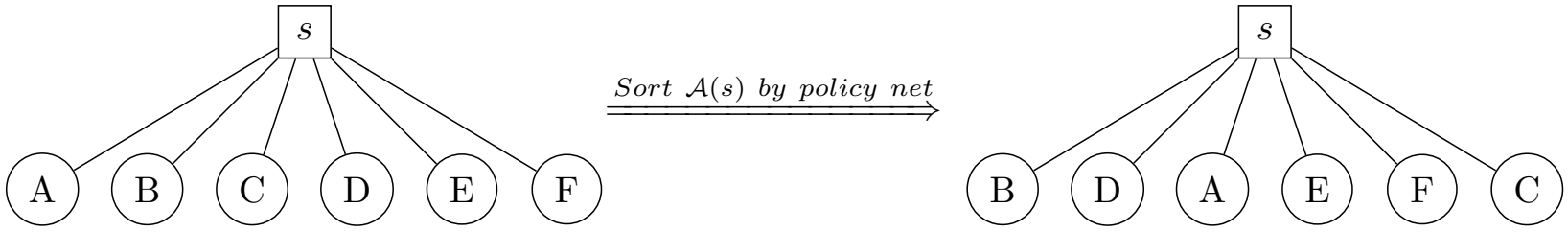
General idea:

- Use policy neural net for move ordering
- Use value neural net to create "artificial branching factor"

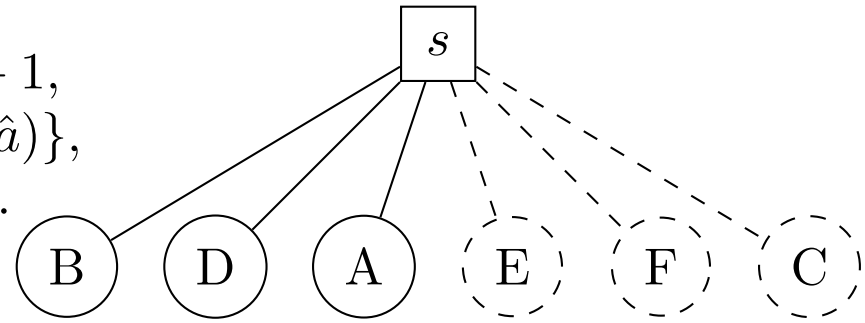
The motivation:

- policy neural net has high top-k prediction accuracy.
- value net provides reliable estimation of the optimal value of expanding node.

Redesign FDFPN with CNNs



$$\begin{cases} l = |\mathcal{A}(s)| \times \min\{\mu, 1 + v_\theta(s)\} + 1, \\ L = \{a \in \mathcal{A}(s) | P_\sigma(s,a) \geq P_\sigma(s,\hat{a})\}, \\ \text{where } \hat{a} \text{ is the } l\text{th move in } \mathcal{A}(s). \end{cases}$$



$v_\theta(s) \in [-1.0, 1.0]$ is the output of the value net.

μ is parameter.

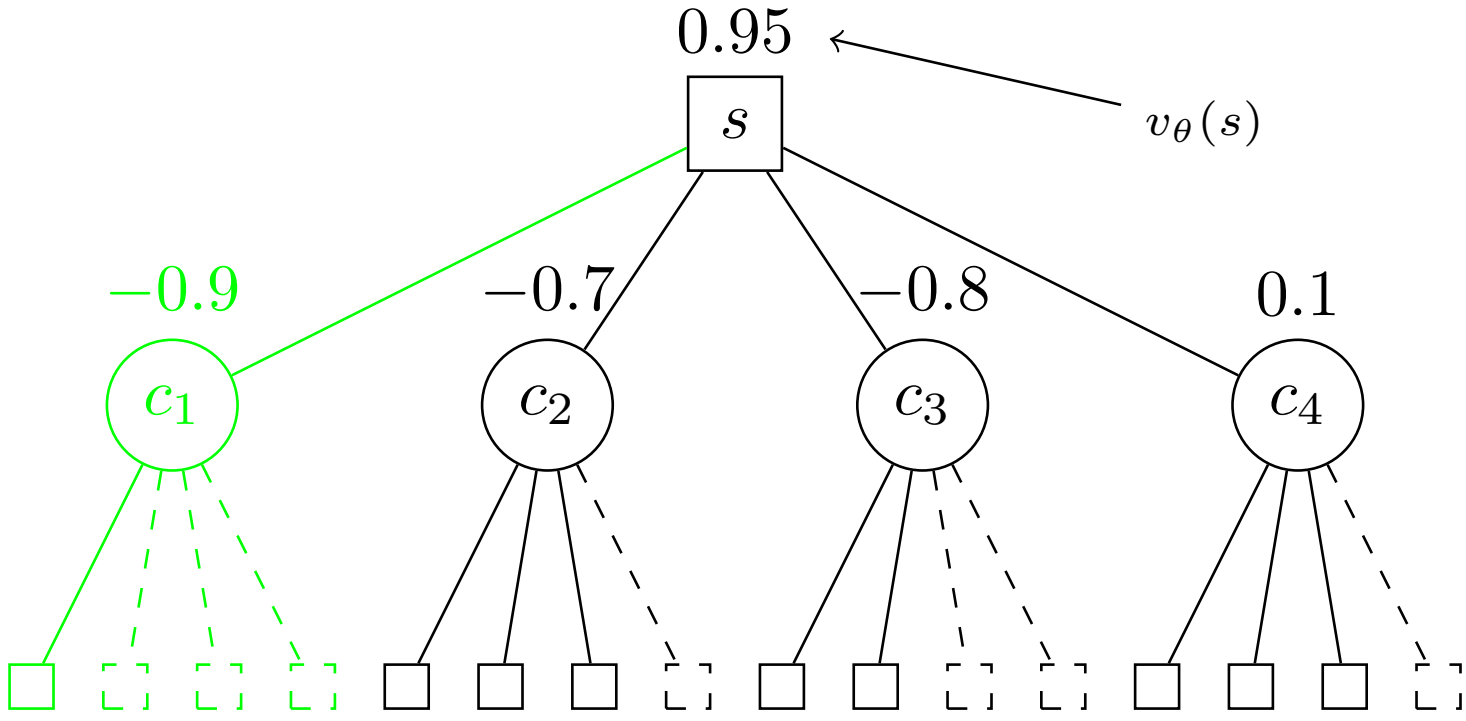
$P_\sigma(s,a)$ is the prior probability from the policy net for state-action (s,a).

Smaller $v_\theta(s)$, smaller search window!

Redesign FDFPN with CNNs

Why smaller window when v_θ is small?

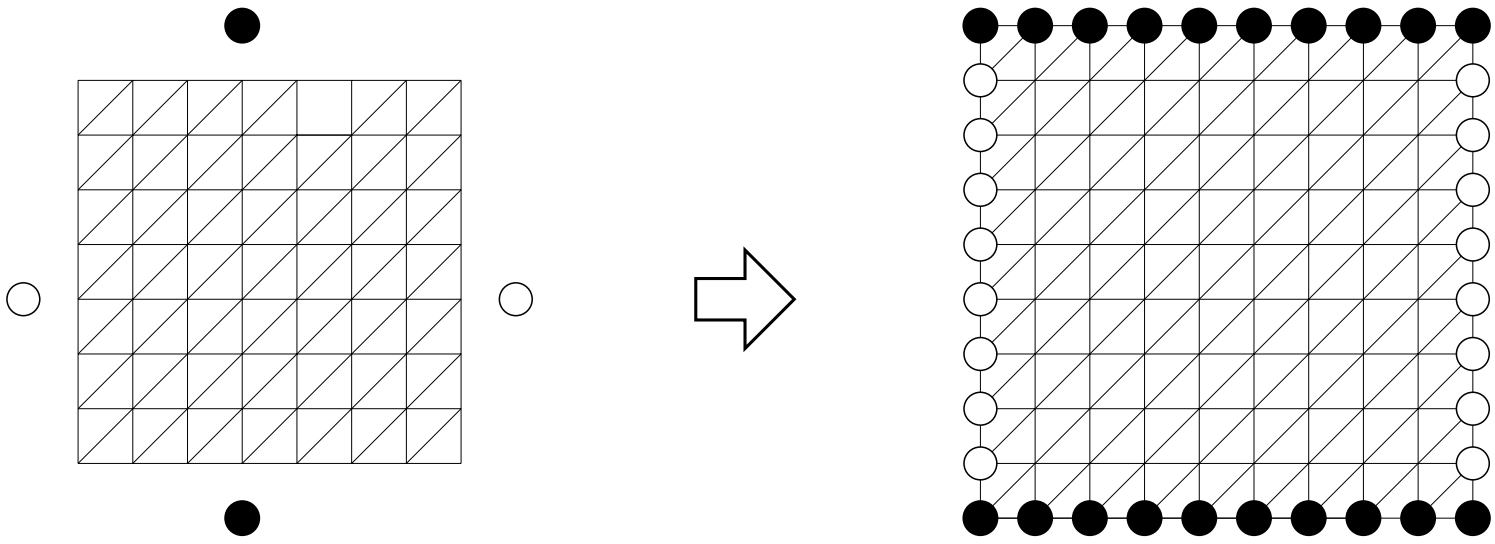
PNS favours "narrow" branches.



Smaller window forces PNS prefer node c_1 (Smaller proof/disproof number).

Experiment: Game studied

8×8 Hex openings.
Tractable but non-trivial.
A Hex-board can be turn into an square board.



Experiments: Neural net architecture

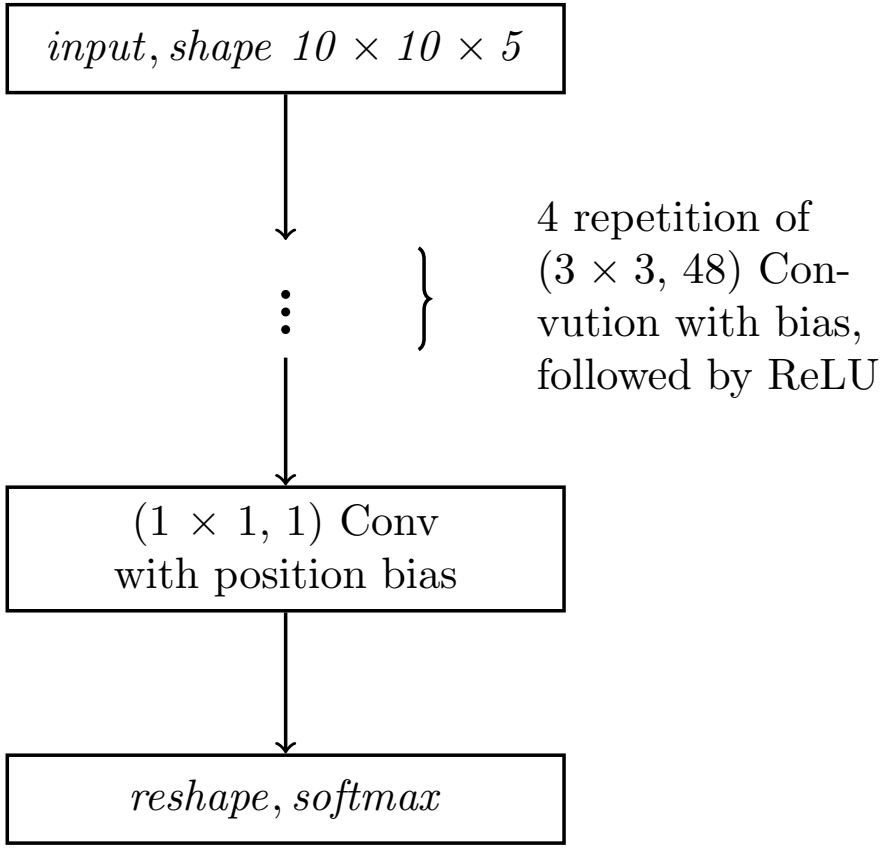


Figure: Architecture for policy net

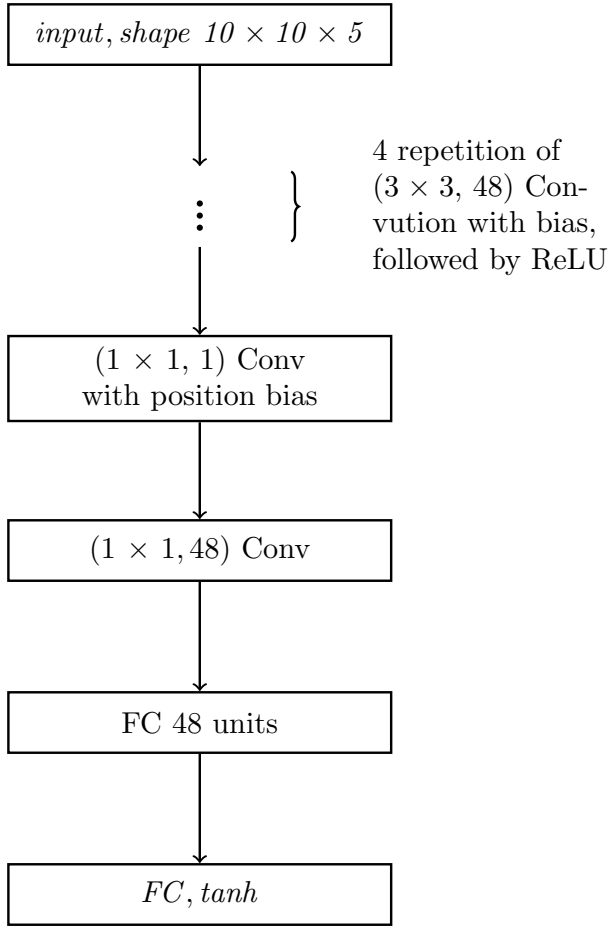


Figure: Architecture for value net

Experiments: policy and value net

Data: 6.5×10^5 state-action or state-value pairs.

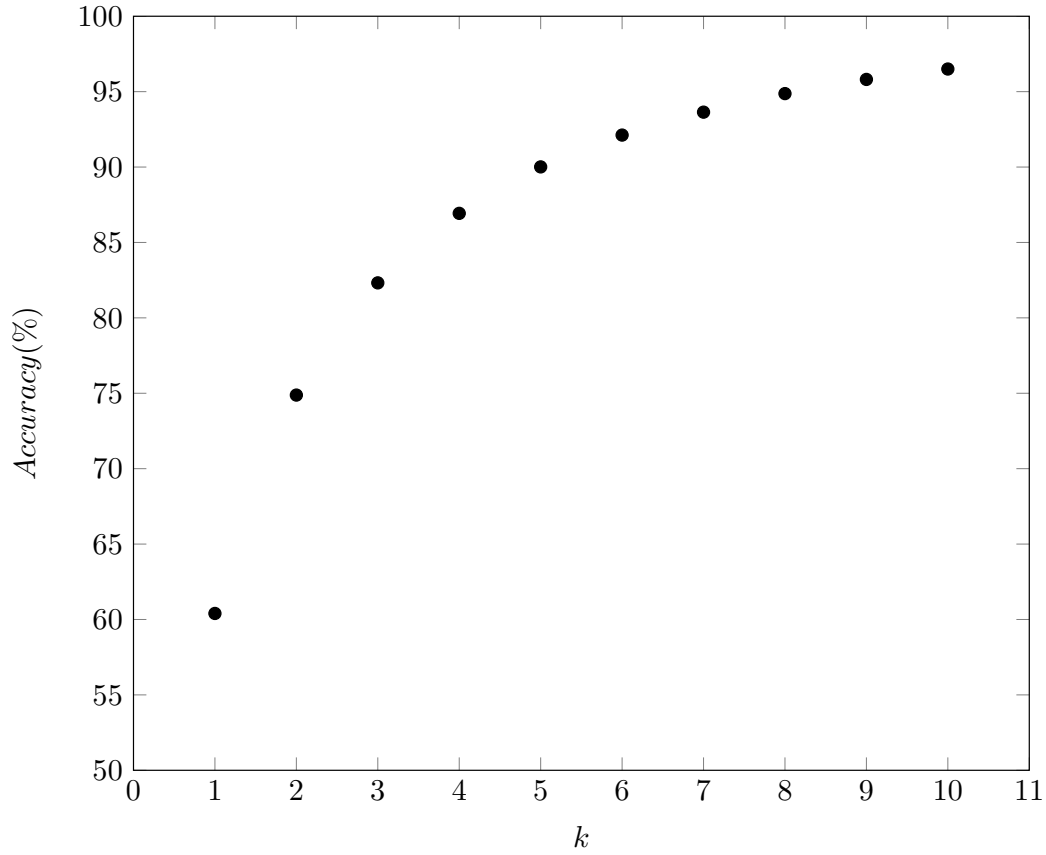
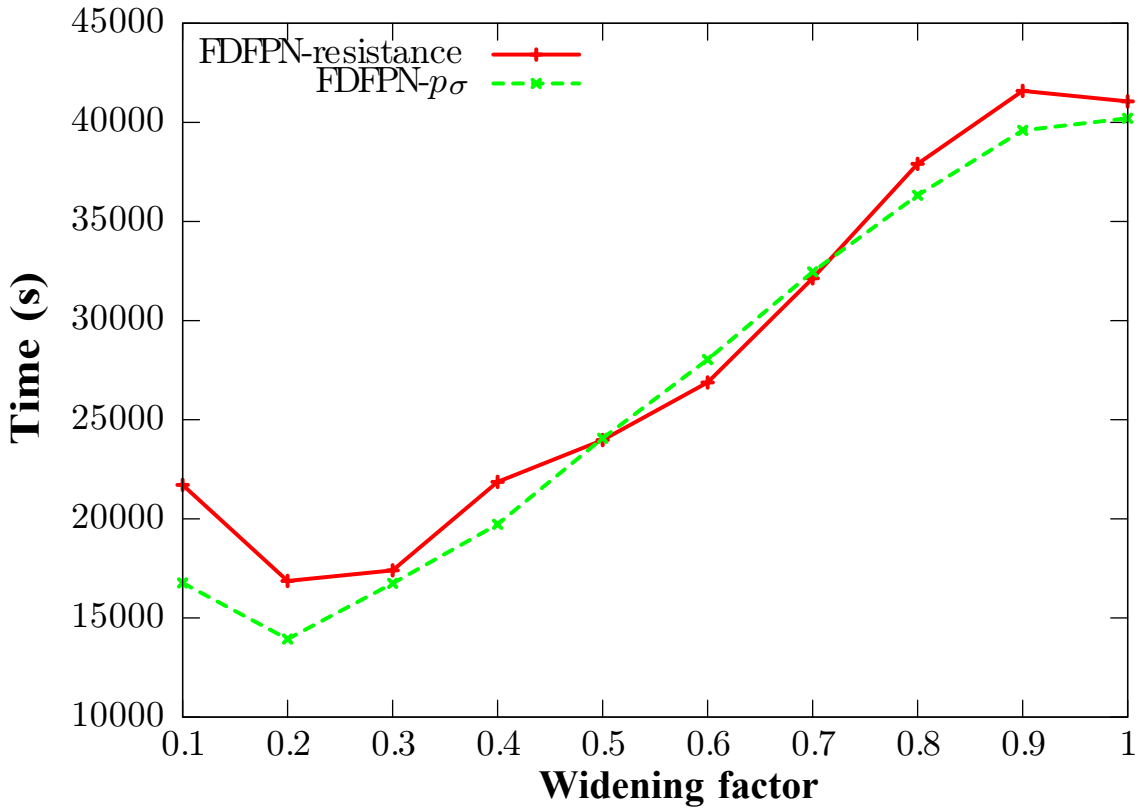


Figure: Top k prediction accuracy on the test data

MSE of value net: 0.083.

Experiments: compare policy net and Resistance

Could FDFPN perform better by just replacing its move ordering function with policy net?



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

policy net selects better moves than Resistance.

Experiments: FDFPN-CNN

Solving all 8x8 opening positions:
Improvement:

- #expansion: 46.7%
- #time: 40% (with Tensorflow 1.0)

Experiments: further investigation of the value net

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

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

What is the **error rate** of f?
about 14.3%.

Conclusions

Conclusion:

- FDFPN-CNN is stronger than FDFPN

Reason:

- Better move selection using policy net
- Creating “artificial” non-uniform branching factor with value net.

Further work

Further improvement:

1. Better neural network design, perhaps residual neural net, and 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.