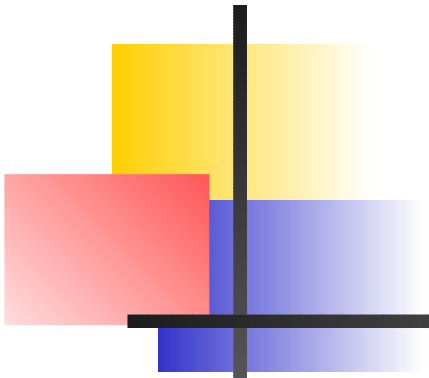


Monte-Carlo Proof-Number Search for Computer Go

Authors: Jahn-Takeshi Saito, Guillaume Chaslot,

Jos W.H.M. Uiterwijk, and H. Jaap van den Herik

Presenter: Philip Henderson

Overview

- Proof Number Search
- MCPNS
- Experimental Setup
- Results and Analysis
- Conclusions and Future Work

Proof Number Search

- Introduced by Allis at al. (1994)
- Goal: to prove the value of a game
- Solved Connect-4, Qubic, Go-Moku
- Variants by Breuker, Nagai, Winands et al.
- Applied to combinatorial games: shogi, go, checkers

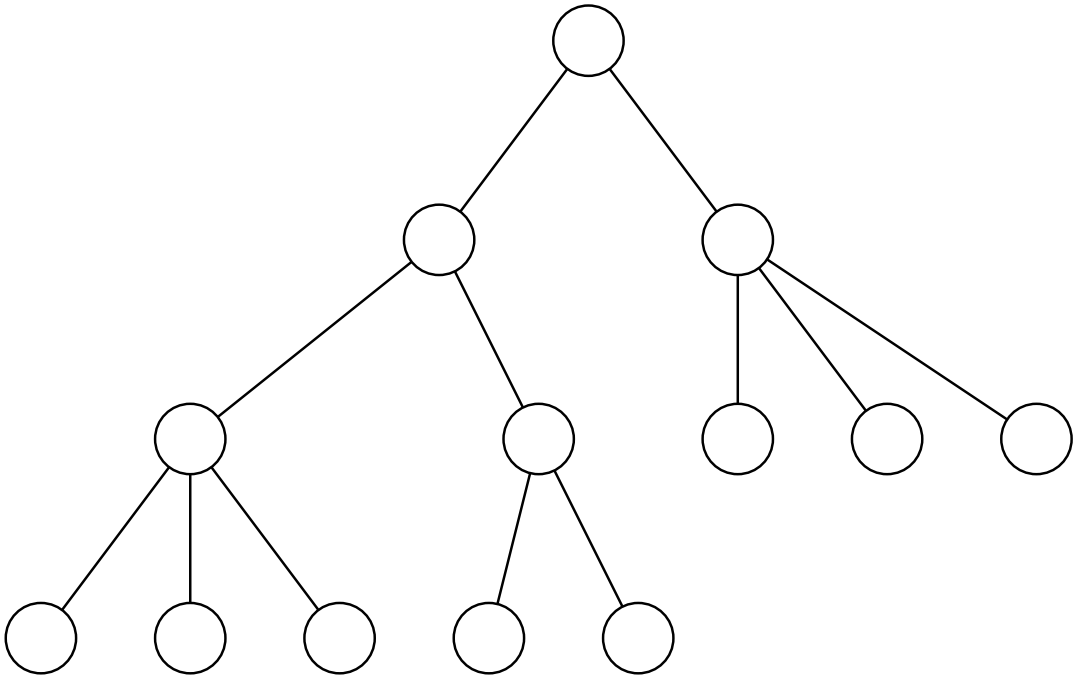
PNS Details (1)

- Most proving node: node which can contribute the most to the establishment of the root's minimax value with the least possible effort
- At max nodes, select branch where likely to prove value 1 with least effort
- At min nodes, select branch where likely to prove value 0 with least effort

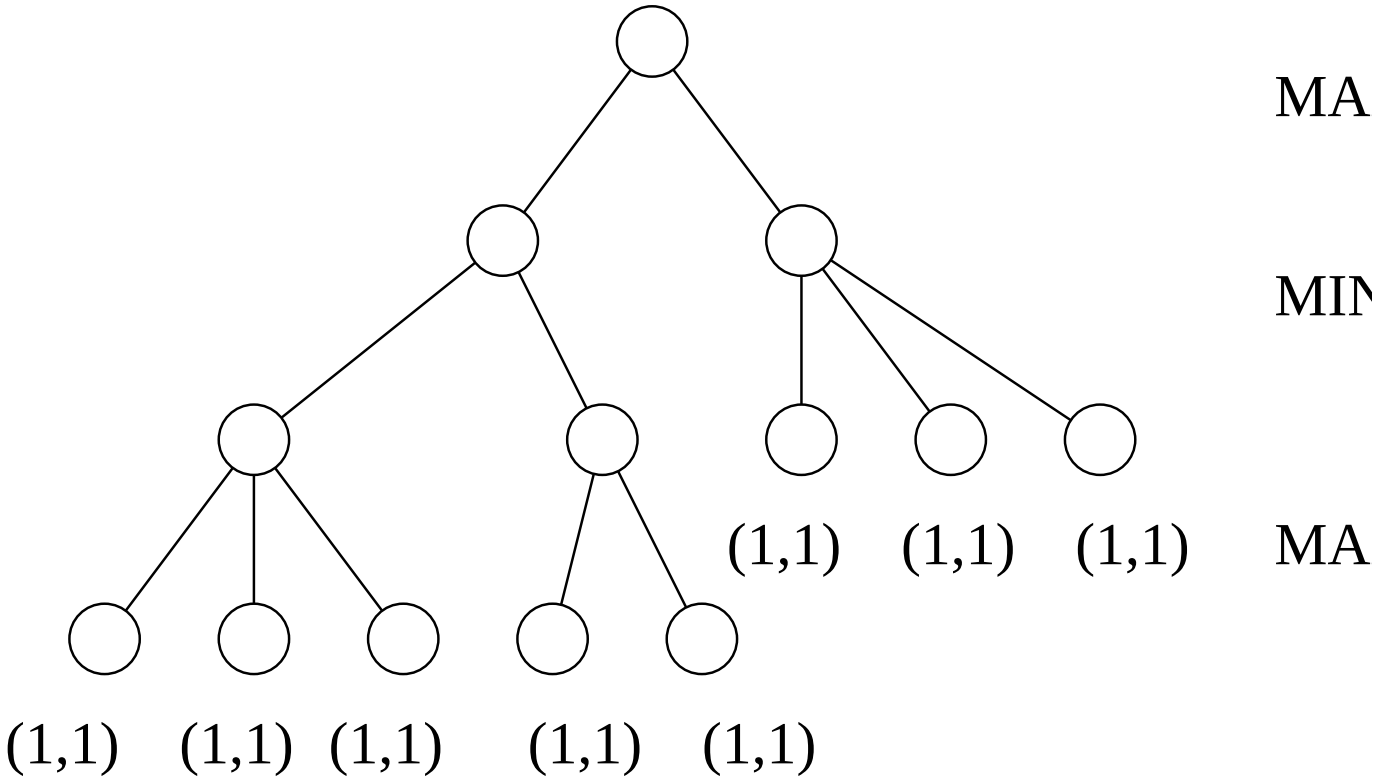
PNS Details (2)

- At each node visited, store (proof number, disproof number)
- Final game states: win $= (0, \infty)$, loss $= (\infty, 0)$
- Temporary leaves: $(1, 1)$ - assumes win/loss have equal chance
- Max internal nodes: (min proof number of children, sum of disproof numbers of children)
- Min internal nodes: (sum of proof numbers of children, min disproof number of children)

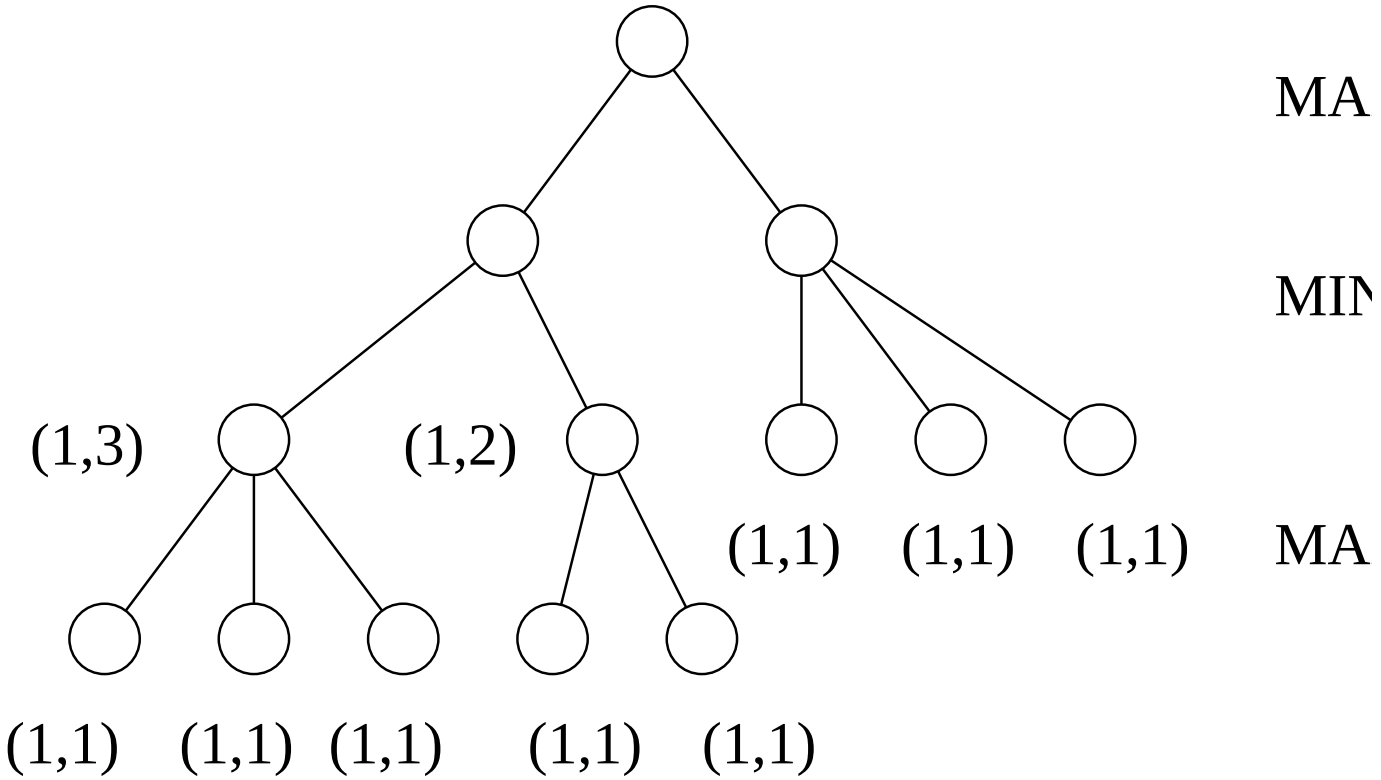
PNS Example



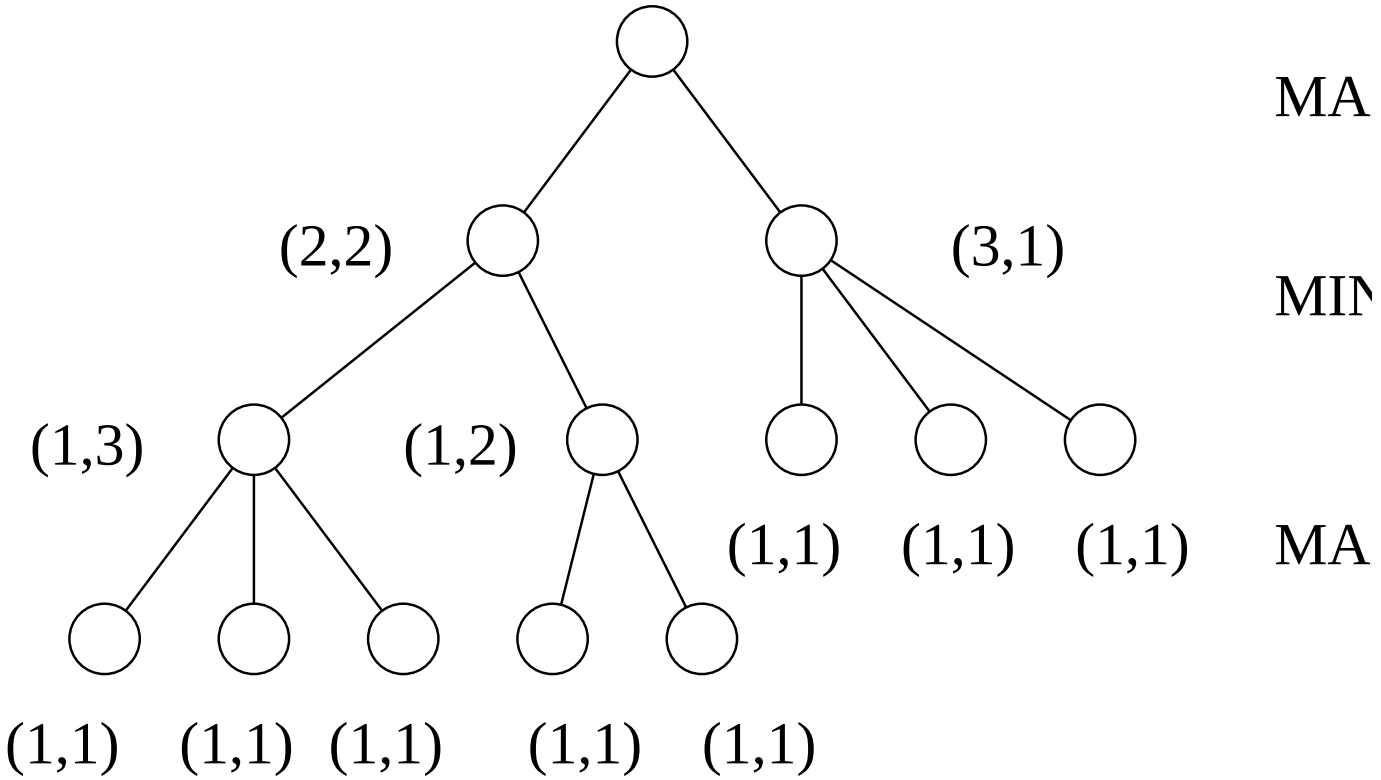
PNS Example



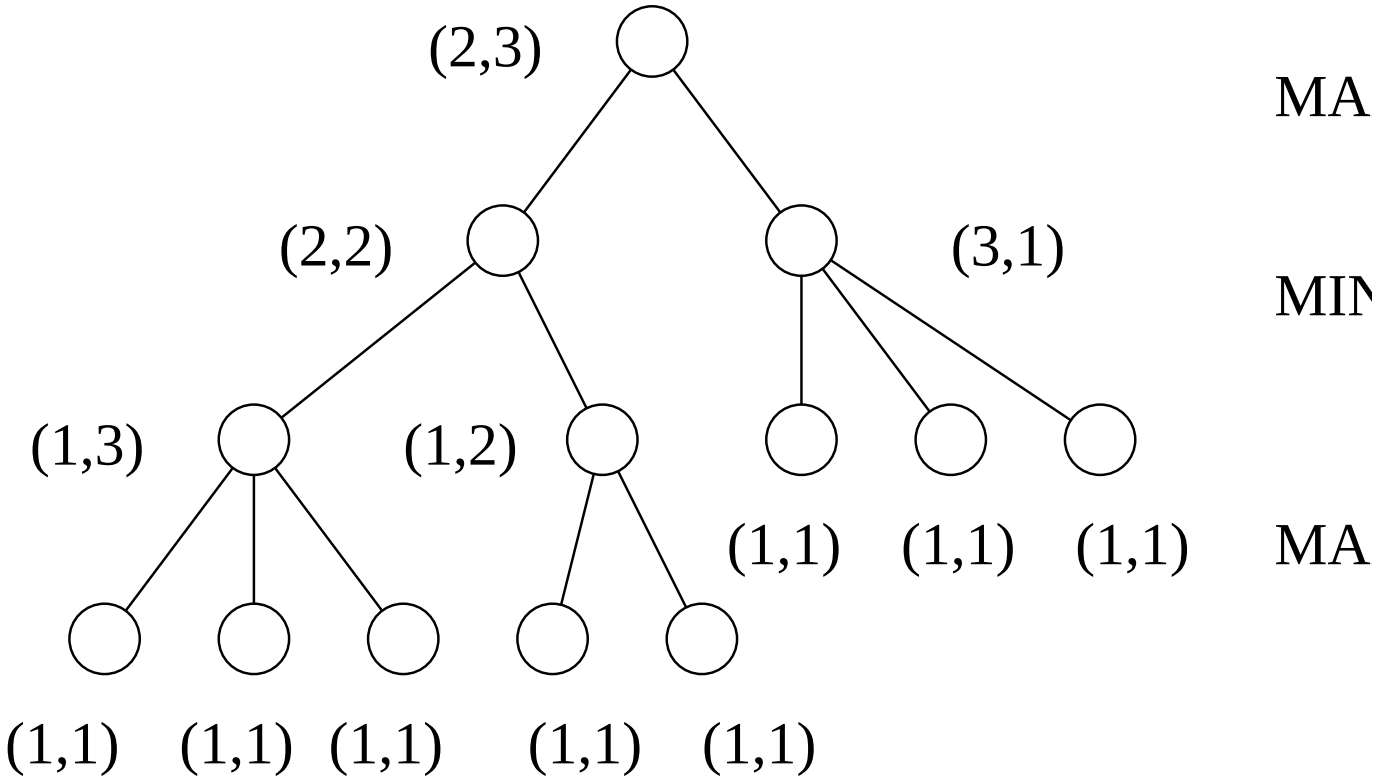
PNS Example



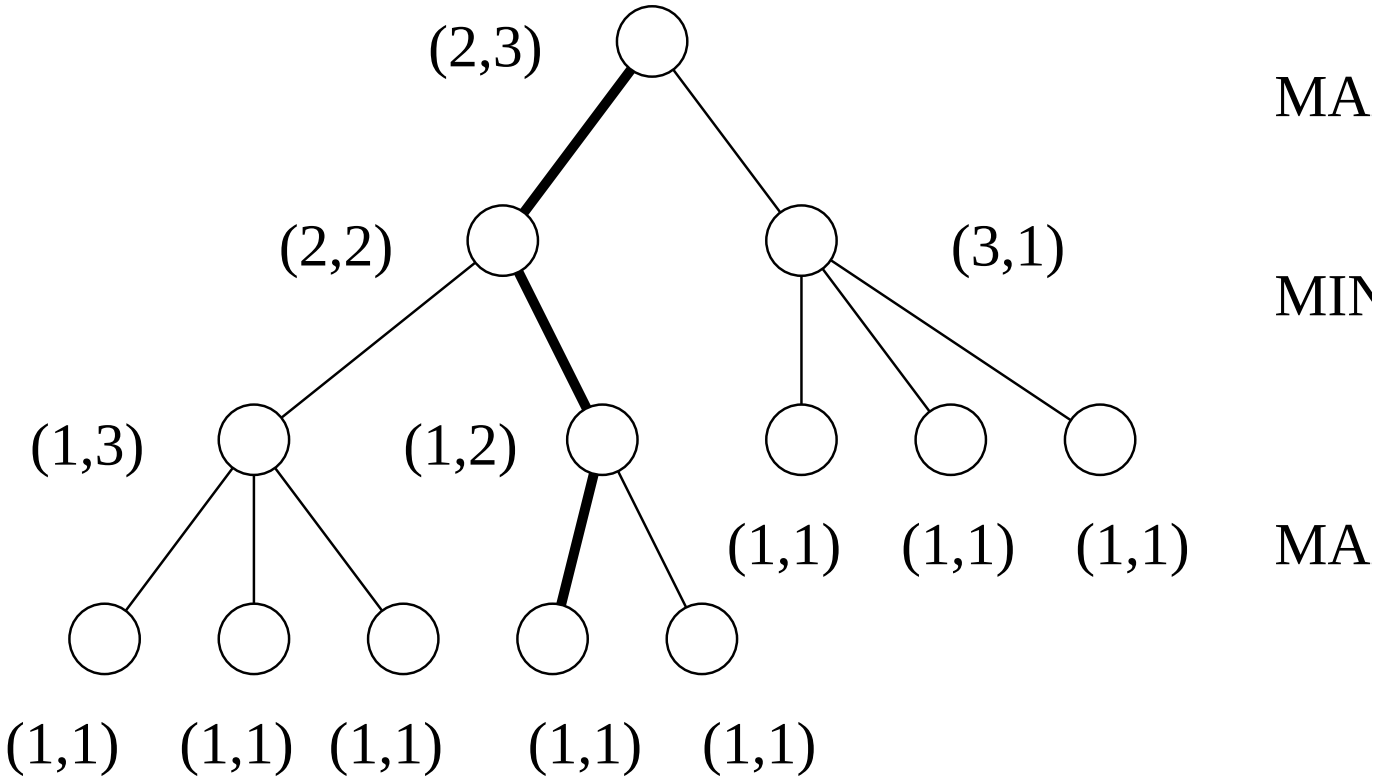
PNS Example



PNS Example



PNS Example



MCPNS

- PNS uses no domain-dependent information
- For temporary leaves, win/loss are not always equally likely
- MC can provide us a better estimate of win/loss likelihood
- Use MC to give proof/disproof numbers in range (0, 1] for temp leaves

Tests

- 30 tsumego, 10k - 1d level from GoBase
- Alive/dead categorization only (ko and seki omitted)
- All positions advantageous for Black (who moves first)
- Annotations added for which groups to be decided and which intersections playable ($I \in [8, 20]$)

MCPNS Parameters

- N - number of simulated games $\in (3, 5, 10, 20)$
- la - lookahead depth (max length of gameplay) $\in (3, 5, 10)$
- depth - level at which start using MC as a guide $\in (I, I/2, 3I/4)$

Additional Info

- 32 GB of working memory available
- Implemented in C++
- MANGO used for MC evaluation, no adaptation for tsumego
- Each test repeated 20 times, aggregates analyzed
- MCPNS variants compared with basic PNS

Results

- Fastest: $p_{fast} = (3, 10, 3)$ - twice as fast, expands $< 1/4$ the nodes
- Smallest: $p_{narrow} = (20, 10, 3)$ - expands $< 1/5$ nodes, but a little slower than PNS
- MCPNS variants expand fewer nodes than PNS
- Time required is more variable - PNS fastest in 6 cases, tied for fastest in another 6

Analysis

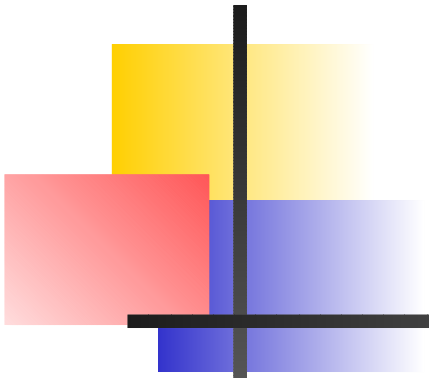
- Tradeoff between number nodes expanded and time spent per node
- MC is time-costly, so use few of these
- PNS fastest on simpler problems, but for complex problems the absolute time savings by MCPNS are significant: 47 seconds versus 6 seconds on most complex problem

Conclusions

- Added domain dependent info to PNS by using MC
- Fewer nodes expanded, but higher time cost per node
- Correct choice of parameters is faster for more complex problems

Future Work

- Larger and more complex problem set needs to be tested
- Only aggregates analyzed, how large is variance of MCPNS
- Use this idea to extend Depth-First PNS



Any Questions?