Enhanced Forward Pruning

Artificial Intelligence for Games

Qingyang Cao

02.05.2019

CONTENTS

Background

Plausible-move generating programs

tactical weaknesses

Until the mid-1970s

Brute-force search programs

- **α-β search**

  TECH (Gillogly, 1972) and CHESS 4.X (Slate and Atkin, 1977)

BELLE (Condon and Thompson, 1983a,b),
DEEP THOUGHT (Hsu, Anantharaman, Campbell, and Nowatzyk, 1990),
HITECH (Berliner and Ebeling, 1990; Berliner, 1987; Ebeling, 1986),
CRAY BLITZ (Hyatt, Gower, and Nelson, 1990), …

Forward-pruning programs

- **null-move pruning**

  Beal, 1989; Goetsch and Campbell, 1990; Donninger, 1993
- **In Principle Variation Search**
- **multi-cut**

variable-depth search(selective extensions)

In the early 1990s

CONTENTS

α-β pruning search

lower (α) and upper (β) bounds on the expected values of the tree:
α : the minimum score that the maximizing player is assured of, initially negative infinity
β : the maximum score that the minimizing player is assured of, initially positive infinity

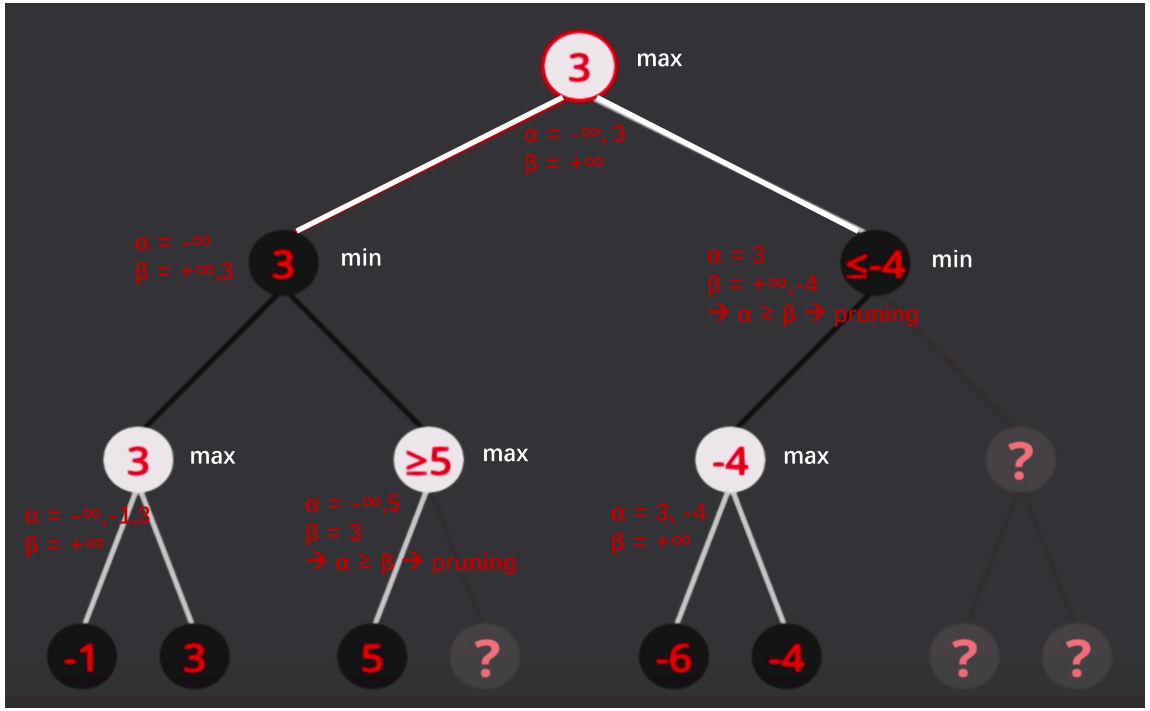
At maximizing player node: whenever alpha ⩾ beta → no need to consider further descendants of this node

More efficient when it is ordered with the most possible one first

```
function minimax(position, depth, alpha, beta, maximizingPlayer)
  if depth == 0 or game over in position
    return static evaluation of position

  if maximizingPlayer
    maxEval = -infinity
    for each child of position
      eval = minimax(child, depth - 1, alpha, beta, false)
      maxEval = max(maxEval, eval)
      alpha = max(alpha, eval)
      if beta <= alpha
        break
    return maxEval

  else
    minEval = +infinity
    for each child of position
      eval = minimax(child, depth - 1, alpha, beta, true)
      minEval = min(minEval, eval)
      beta = min(beta, eval)
      if beta <= alpha
        break
    return minEval
```



CONTENTS

Verified null move pruning

STANDARD NULL-MOVE PRUNING

- Cutoff decisions on dynamic criteria → greater tactical strength
- **Assumptions:** Null move is never a best choice
  (a null move search → a lower bound α (updating))
- **Null move search**
  1. swap the side.
  2. then conduct a regular search with **reduced depth**
- **Cutoffs**
  1. β ⩽ value : a cutoff (a fail-high)
  2. α < value ⩽ β: update α = value.
  3. value < α: no cutoff nor updating
- **Benefit :** β ⩽ value → cutoff
- **Minimal-window null-move search around β**

R: depth reduction factor

```
/* the depth reduction factor */
#define R  2
int search (alpha, beta, depth) {
  if (depth <= 0)
    return evaluate(); /* in practice, quiescence() is called here */
  /* conduct a null-move search if it is legal and desired */
  if (!in_check() && null_ok()){
    make_null_move();
    /* null-move search with minimal window around beta */
    value = -search(-beta, -beta + 1, depth - R - 1);
    if (value >= beta) /* cutoff in case of fail-high */
      return value;
  }
  /* continue regular NegaScout/PVS search */
  ...
}
```

Verified null move pruning

STANDARD NULL-MOVE PRUNING

- **Flaws**
  Zugzwang: Null move is the best choice
  Horizon effect (Berliner, 1974) : when the reduced-depth search misses a tactical threat

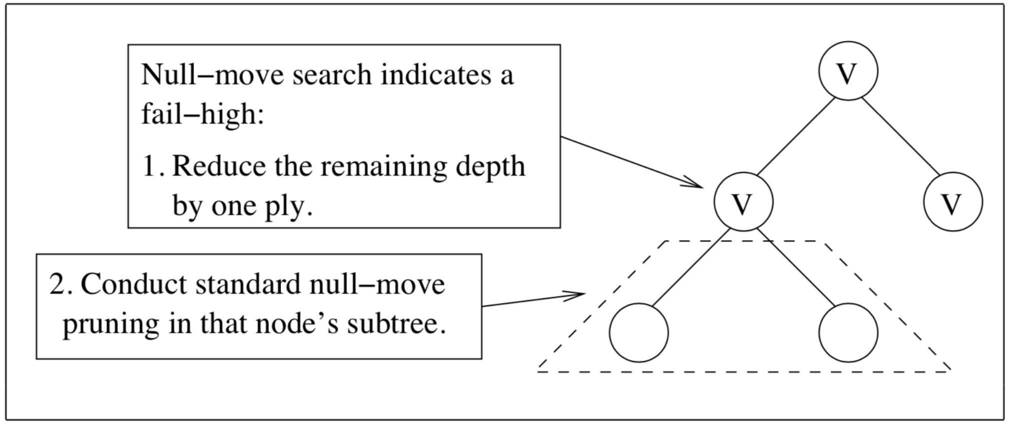
- **Choice of R**
  R = 2 performs better & mostly used (Feist, 1999);
  R = 1 too conservative; R = 3 too aggressive (Heinz, 1999)
  Adaptive R (Donninger,1993) → adaptive null-move pruning (Experiments by Heinz, 1999)

Verified null move pruning

VERIFIED NULL-MOVE PRUNING

- Verification idea
  No immediate pruning:
  When a fail-high occurs: continue the search with reduced depth (Goetsch and Campbell, 1990)
  Can prevent errors (Plenkner, 1995)

- Verified null move pruning
  1. at each node: null-move search (R = 3)
  2. at nodes that value ⩾ β: 1) reduce the depth by one ply
     2) continue the search for that node' s subtree using standard null-move pruning (with R = 3)



- cutoffs
  1. nodes having another null-move search fail-high indication in one of its ancestors → cutoffs
  2. the null-move search: cutoff;
     the search: the best value < β → Zugzwang → restore the original depth + re-search

Verified null move pruning

VERIFIED NULL-MOVE PRUNING

- **Strength**

  1. Reduced search tree size
  2. Greater tactical strength(Good with zugzwang positions)
  3. Easy to implement
  5. Applicable to all standard null-move pruning program

- **Experimental results**

  - The NEGASCOUT/PVS (Campbell and Marsland, 1983; Reinefeld, 1983) search algorithm
  - History heuristic (Schaeffer, 1983, 1989)
  - Transposition table (Slate and Atkin, 1977; Nelson, 1985)
  - The tactical strength differences: one-ply check extensions on leaf nodes

VERIFIED NULL-MOVE PRUNING

- **Experimental results**

  138 test positions from *Test Your Tactical Ability* by Yakov Neishtadt
  Depths: 9 and 10 plies
  R = 1, R = 2, R = 3, and verified R = 3.

Depth	Std $R = 1$	Std $R = 2$	Std $R = 3$	Vrfd $R = 3$
9	1,652,668,804 (+267.46%)	603,549,661 (+34.19%)	267,208,422 (-40.58%)	449,744,588 -
10	11,040,766,367 (+661.64%)	1,892,829,685 (+30.57%)	862,153,828 (-40.52%)	1,449,589,289 -

Table 1: Total node count of standard $R = 1, 2, 3$ and verified $R = 3$ at depths 9 and 10, for 138 Neishtadt test positions.

Depth	Std $R = 1$	Std $R = 2$	Std $R = 3$	Vrfd $R = 3$
9	64	62	53	60
10	71	66	65	71

Table 2: Number of solved positions using standard $R = 1, 2, 3$ and verified $R = 3$ at depths 9 and 10, for 138 Neishtadt test positions.

Verified null move pruning

VERIFIED NULL-MOVE PRUNING

- **Experimental results**

  869 positions from the *Encyclopedia of Chess Middlegames* (ECM)4.
  Depth: 11 plies

Depth	Std $R = 1$	Std $R = 2$	Std $R = 3$	Vrfd $R = 3$
9	64	62	53	60
10	71	66	65	71

Table 2: Number of solved positions using standard $R = 1, 2, 3$ and verified $R = 3$ at depths 9 and 10, for 138 Neishtadt test positions.

Depth	Std $R = 2$	Std $R = 3$	Vrfd $R = 3$
9	5,374,275,763 (+10.84%)	2,483,951,601 (-48.76%)	4,848,596,820 -
10	16,952,333,579 (+17.40%)	7,920,812,800 (-45.14%)	14,439,185,304 -
11	105,488,197,524 (+106.51%)	24,644,668,194 (-51.75%)	51,080,338,048 -

Table 3: Total node count of standard $R = 2$, $R = 3$, and verified $R = 3$ at depths 9, 10, and 11, for 869 ECM test positions.

CONTENTS

Forward pruning works in PVS

THREE NODE TYPES

- **Principle Variation node**
  1) The root of the tree;
  2) a successor PV node: The best move found at a PV node
  All children have to be explored; Best move must be considered first;
  Returned score s, [a,b], a<s<b; On the principal variation

- **Cut node** (fail-high nodes)
  1) all the other investigated children at a PV node;
  2) successors of an ALL node
  Only one child(the first) has to be explored in a perfectly ordered tree; s ⩾ b;
  Best move must be considered first; Alternatives to the principal variation

- **All node**(fail-low nodes)
  1) successors of a Cut node
  All children have to be explored(no move will cause a beta-cutoff)

- **expected CUT node→ ALL node** (If none of the moves causes a cutoff at this expected CUT node)
- **expected ALL node→ CUT node** (If one of the children turns out not to be a CUT node)
- **new principal variation:** (all expected CUT nodes on a path from the root to a leaf node have become ALL nodes)

04 Forward pruning works in PVS

PRINCIPAL VARIATION SEARCH

- **Null window searche**s for **none PV-nodes**
  To prove a move is worse or not than an already safe score from the principal variation

- Determine the **expected & true type** of a node:

  expected*
  1. **PV nodes**: first node explored at the root & subsequent PV nodes → best value
  2. **none PV-nodes**: nodes not on the principal variation: alternately CUT or ALL nodes
     Null-window search (closed αβ-window/ NW: $\beta = \alpha + 1$(J.P. Fishburn, 1981, 1984))
     → score s

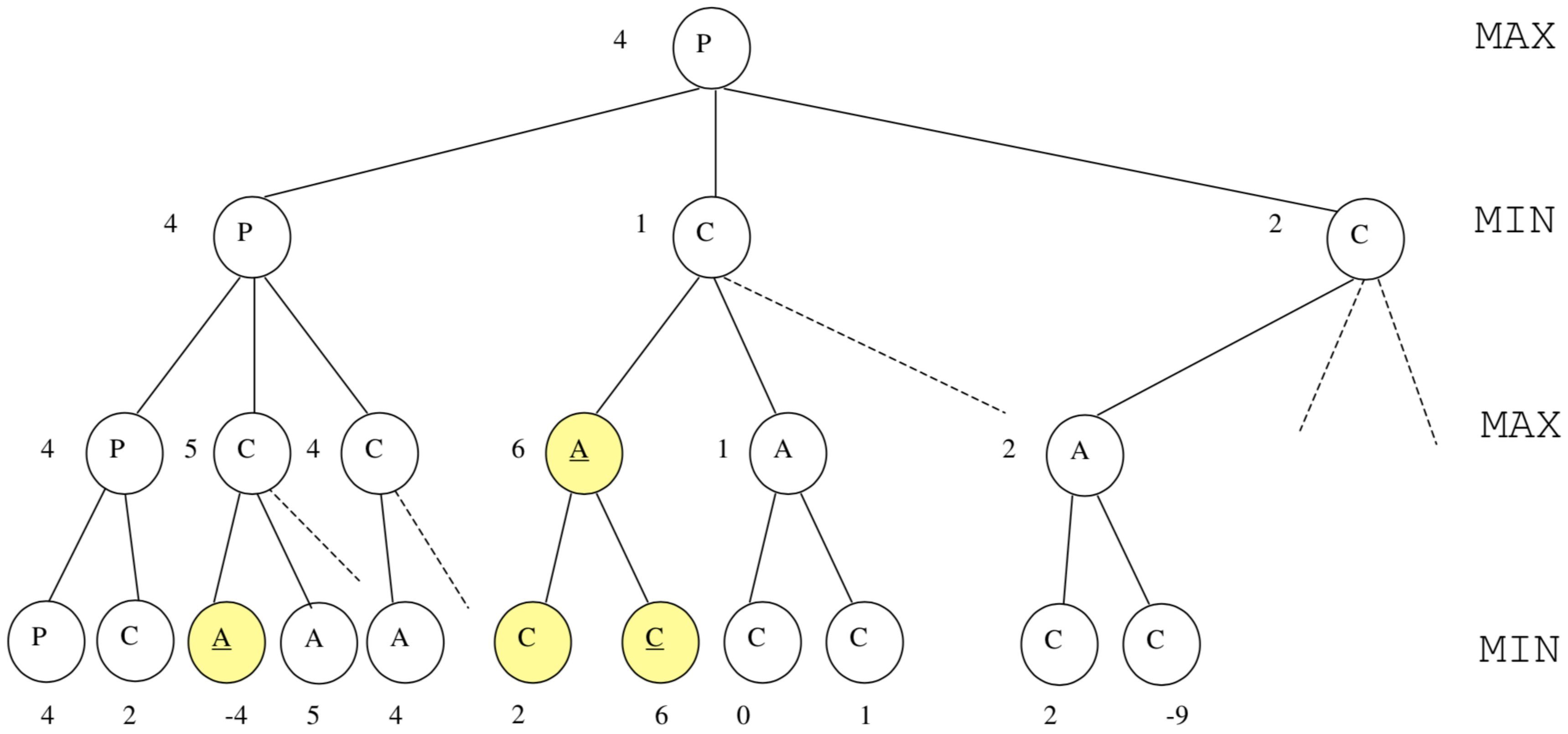
  true?
  1) $s \leqslant \alpha$ → that particular sibling has been proved inferior.
  2) $s > \alpha$ → re-search (αβ-window is opened/full window search & the child node → PV node.)

* Expected node types are determined by tree topology, probing the transposition table, or comparing scores of a static evaluation considering threats, or even a reduced search or quiescence search, with the bounds, may be considered by various (parallel) search algorithms and in decisions concerning selectivity .

Forward pruning works in PVS

PRINCIPAL VARIATION SEARCH

Assumes an underlying hierarchical processor organization



How to derive 3 type nodes with PVS: T. A. Marsland and M. Campbell. 1982. Parallel Search of Strongly Ordered Game Trees. ACM Comput. Surv. 14, 4 (December 1982), 533-551. DOI=http://dx.doi.org/10.1145/356893.356895

Forward pruning works in PVS

PRINCIPAL VARIATION SEARCH

- **Forward-pruning** only for the **NWS part**
- **Outcome** of Forward pruning by mistake
  1. at an expected **PV node**: too risky
  2. at an expected **CUT node**: fail low mistake*
  3. at an expected **ALL node**: fail high mistake**

*Fail low: The score returned is a upper bound on the exact score of the node. alpha; appears at All nodes: indicates that this position was not good enough for you. You will not reach this position, because you already have other choices that is better. You will not make the move that allowing the opponent to get you into this position.
**Fail high: beta; appears at Cut nodes: indicates that the search found something that was "too good". What this means is that the opponent can, which is already found by the search, avoid getting into this very bad position for himself, And since the opponent can, and he will avoid this position, there is no point to search its successors

fail-soft αβ: the alphabeta function may return values (v) that exceed the α and β bounds set (v < α or v > β) by its function call arguments.
fail-hard αβ: limits its function return value into the inclusive range of α and β.

Forward pruning works in PVS

PRINCIPAL VARIATION SEARCH

- **Remedies for Forward-pruning:**
  1. To avoid that a backed-up value of a forward-pruned ALL node causes a β-cutoff at the PV node lying above, β is returned in case of a cut-off (β =α + 1 at an ALL node).

  2. If the window of the PV node was already closed(with α、β being updated) and the NWS should return a value of β (α + 1), a re-search is still have to be done

  3. If a re-search is done and the returned value of the NWS equals α + 1, we should do a re-search with α as lower bound.

  4. CUT nodes where a fail-low has occurred with a value equal to α are not stored in the transposition table because their values are uncertain.

CONTENTS

MULTI-CUT

- The first **M child** nodes of an expected CUT node are searched to a depth reduced with **a factor R** before examining it to full depth.
  1) At least **C** child nodes return a value larger than or equal to β → cutoff
  2) Otherwise → re-exploring this node to a full depth d

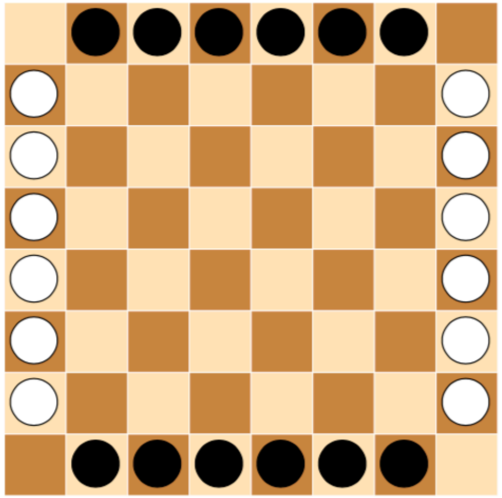
- **Experiment**
  Q: Is multi-cut also useful at ALL nodes?
  A: Multi-cuts at ALL nodes (MC-A) when combined with other forward-pruning mechanisms give a significant reduction of the number of nodes searched.

  As a comparison: a (more) aggressive version of the null move (variable null-move bound) gives less reduction at expected ALL nodes.

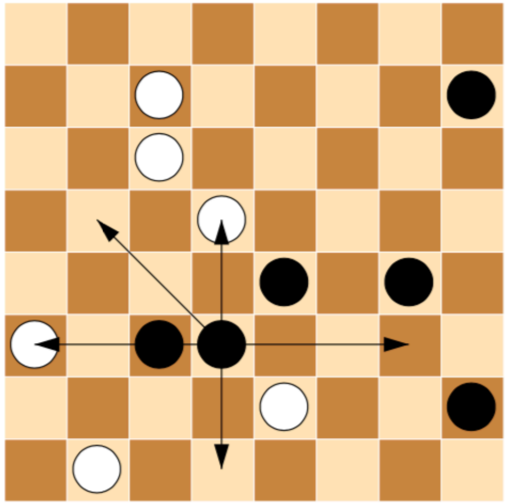
MULTI-CUT

- **Experiment**

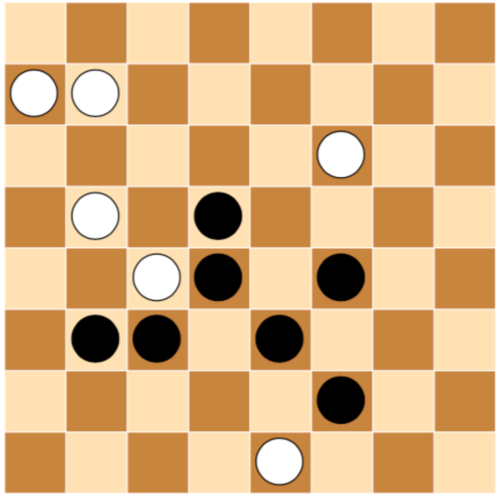
  Game of Lines of Action (LOA)
  - two-person zero-sum chess-like connection game with perfect information
  - 8 × 8 board by two sides; 12 pieces; starting with Black
  - A move takes place as many squares as there are pieces of either colour anywhere along the line of movement; a player may jump over its own pieces, not the opponent' s; capture pieces by landing on them
  - Goal: be the first to create a configuration in which all own pieces are connected in one unit



The initial position of LOA



An example of possible moves in a LOA



A terminal LOA position

Black wins

MULTI-CUT

- **Experiment**

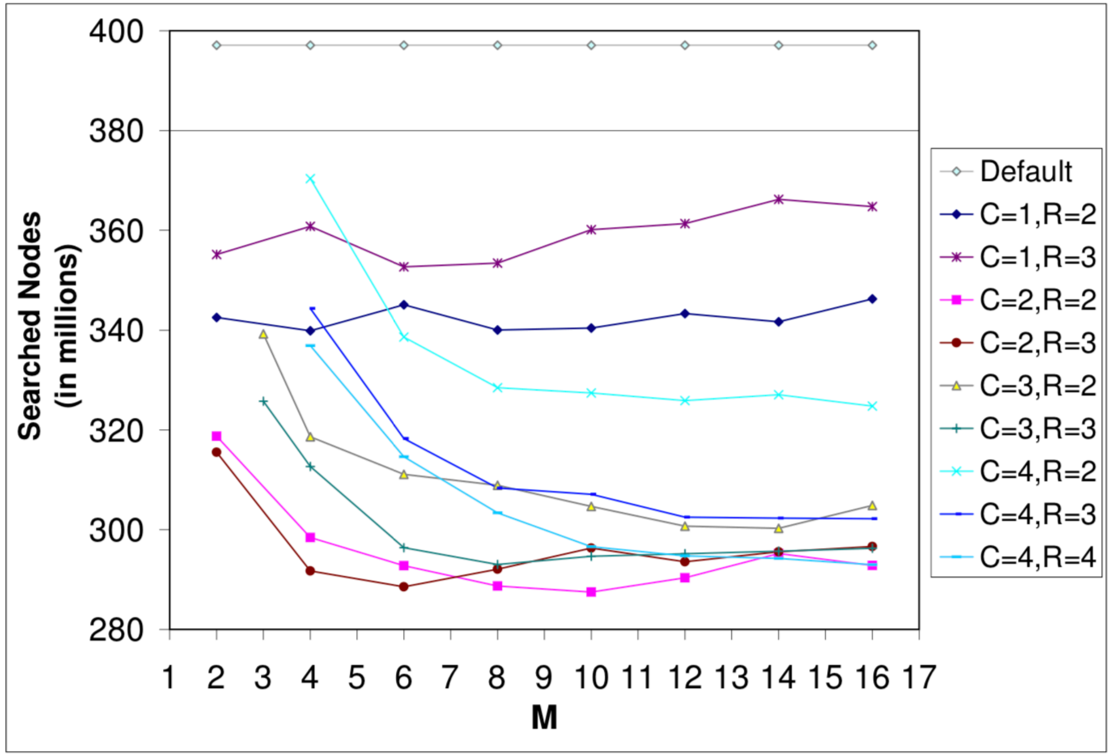
  Search engine MIA(Maastricht In Action)*

  - An αβ depth-first iterative-deepening search in the PVS frame- work
  - To prune a subtree or to narrow the αβ window: two-deep transposition table
  - At all interior nodes which are more than 2 ply away from the leaves:
    Enhanced Transpo- sition Cutoffs (ETC) scheme for
  - A null move is performed first with R at CUT nodes and at ALL nodes
  - To set R at a CUT node: adaptive null move
    R is set to 3 when: 1) the remaining depth is more than 6
    2) the number of pieces of the side to move is lower than 5 the remaining depth has to be more than 8
    R is set to 2 when: other case
  - MC-C: 1) R=3: C=3,M =10, and R= 3
  - …

*The program and the test sets can be found at the website: http://www.cs.unimaas.nl/m.winands/loa/

MULTI-CUT

- **Experiment**



Tree sizes for different C, M and R

MULTI-CUT

- **Experiment**

	No Forward pruning			Only Null move		
d	No MC-A	MC-A	%	No MC-A	MC-A	%
5	5,071,689	4,995,845	98.5	3,504,759	3,404,882	97.2
6	19,896,101	19,286,868	96.9	10,109,533	9,518,082	94.1
7	113,653,808	110,663,056	97.4	36,265,257	34,671,647	95.6
8	416,549,038	406,489,302	97.6	92,749,650	89,483,140	96.5
9	2,427,406,280	2,395,844,102	98.7	314,507,126	303,466,596	96.5
10	9,635,185,102	9,460,591,510	98.2	891,348,022	813,032,326	91.2
11	-	-	-	2,930,142,106	2,599,157,486	88.7
12	-	-	-	8,362,297,395	7,080,475,905	84.7

	Only MC-C			Null move and MC-C		
d	No MC-A	MC-A	%	No MC-A	MC-A	%
5	2,097,908	1,955,564	93.2	2,012,835	1,897,600	94.3
6	7,314,731	5,549,772	75.9	6,083,136	5,122,496	84.2
7	28,656,432	20,221,202	70.6	20,491,711	17,423,109	85.0
8	85,103,638	50,333,688	59.1	50,018,470	42,242,144	84.5
9	297,239,554	149,671,128	50.4	142,182,834	116,784,068	82.1
10	1,286,515,396	393,490,307	30.6	397,092,800	283,350,391	71.4
11	4,860,474,957	1,358,658,246	28.0	1,223,918,717	846,066,886	69.1
12	23,806,355,059	3,536,842,482	14.9	3,328,838,963	2,162,692,924	65.0
13	-	-	-	9,869,101,893	6,289,563,990	63.7
14	-	-	-	30,087,791,323	17,578,589,423	58.4

Added value of MC-A

MULTI-CUT

- **Experiment**

Depth	Original Set (171 positions)	Validation Set (156 positions)
5	94.3	94.7
6	84.2	86.8
7	85.0	83.7
8	84.5	82.0
9	82.1	79.8
10	71.4	73.9
11	69.1	72.0
12	65.0	68.5
13	63.7	64.8
14	58.4	61.4

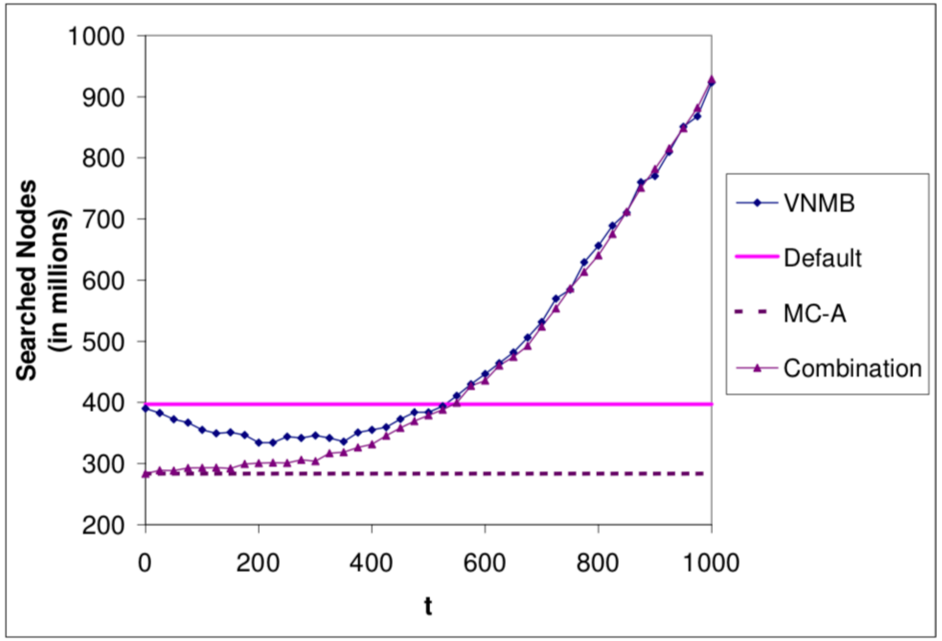
Relative performance of MC-A in combination with null move and MC-C

MULTI-CUT

- **Experiment**

  Variable null-move bound

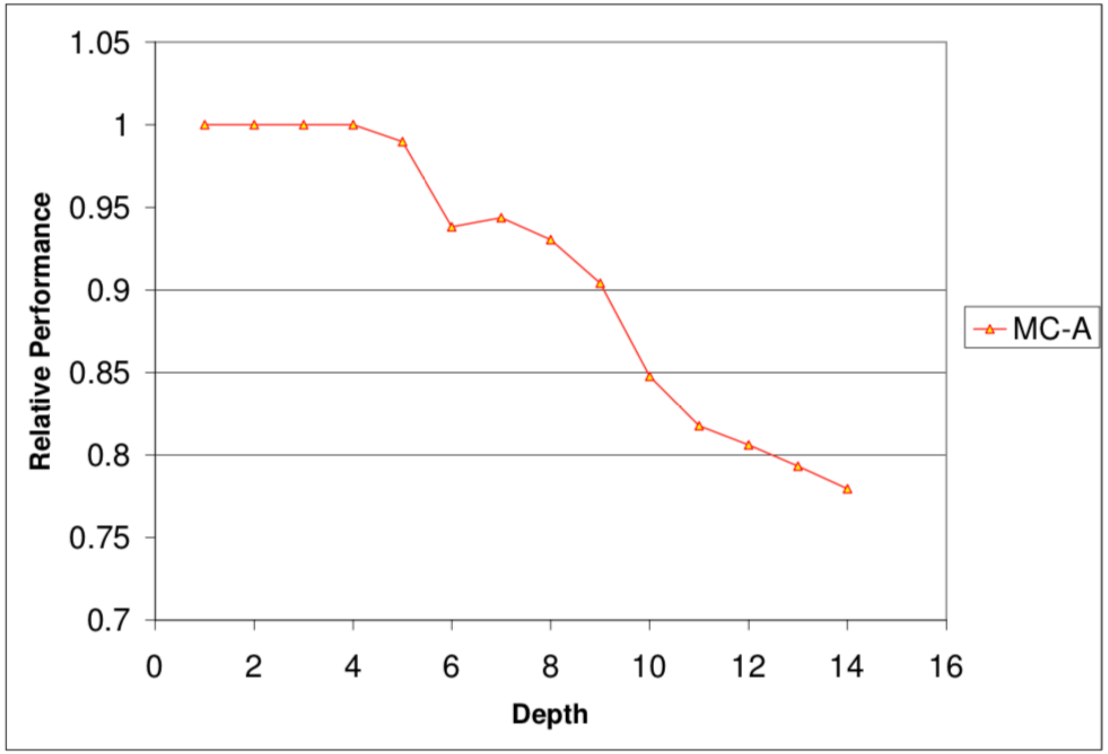
  - A null-move cutoff can be forced if the returned null-move search value is larger than or equal to $\beta - t$, where t is the minimal value of a tempo depending on the evaluation function
  - Allows a larger part of the null-move searches to cause cut-offs



Variable null-move bound

MULTI-CUT

- **Experiment**



MC-A compared to variable null-move bound

MULTI-CUT

- **Experiment***

	Score	Winning ratio
MC-A vs. Default	549-451	1.21

1000-game match results

modified version outplayed the original version with a winning ratio of 1.21 (i.e., scoring 21% more winning points than the oppo- nent).
→ MC-A improves the playing strength of MIA significantly

- **Conclusion**

forward pruning at expected ALL nodes is safe and beneficial

*please find more experimental information in [3]

CONTENTS

Reference

[1] Marsland, T. A.. "A Review of Game-Tree Pruning." ICGA Journal 9 (1986): 3-19.

[2] David, Omid and Nathan S. Netanyahu. "Verified Null-Move Pruning." ICGA Journal 25 (2002): 153-161.

[3] Winands, Mark H. M. et al. "Enhanced forward pruning." Inf. Sci. 175 (2003): 315-329.

[4] Marsland, T. Anthony and Murray Campbell. "Parallel Search of Strongly Ordered Game Trees." ACM Comput. Surv. 14 (1982): 533-551. (for PVS)

[5] Björnsson, Yngvi & Marsland, T. Anthony & Schaeffer, Jonathan & Junghanns, Andreas. (1997). Searching with Uncertainty Cut-Offs in Game-Tree Pruning. International Computer Chess Association Journal. 20. 29-37.

[5] Yngvi Björnsson. 2002. Selective Depth-First Game-Tree Search. Ph.D. Dissertation. University of Alberta, Edmonton, Alta., Canada. Advisor(s) Tony Marsland. AAINQ68547.

[6] Marsland, T. Anthony and Fred Popowich. "Parallel Game-Tree Search." IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-7 (1985): 442-452.

Reference

[7] Marsland, T. Anthony. “Relative Efficiency of Alpha-Beta Implementations.” IJCAI (1983).

[8] Marsland, T. Anthony & Björnsson, Yngvi. (2001). Variable-Depth Search. 9-24.

[9] Björnsson, Yngvi & Marsland, T. Anthony. (2001). Multi-cut alpha-beta-pruning in game-tree search. Theoretical Computer Science. 252. 177-196. 10.1016/S0304-3975(00)00081-5.

Thanks for listening

Artificial Intelligence for Games

Qingyang Cao

02.05.2019