Nested Monte Carlo Search for Two-Player Games

Tristan Cazenave
Abdallah Saffidine
Michael John Schofield
Michael Thielscher
2016

Date: 09.06.21
By: Nina Braunmiller
Subject: SE Seminar in AI
Supervisor: Prof. Johannes Fürnkranz

Outline

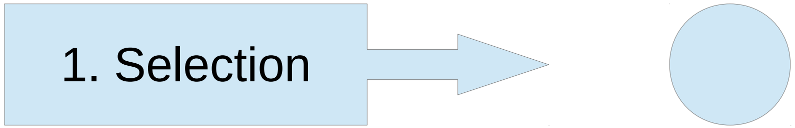
I. Nested Monte Carlo Tree Search (NMCS)
   i. definition
   ii. For one player games
   iii. Problems for two player games
II. NMCS improvements
   i. Discounting
   ii. Pruning
      - Cut on Win
      - Pruning on Depth
III. Results
IV. Conclusions

Outline

I. **Nested Monte Carlo Tree Search (NMCS)**
   i. **definition**
   ii. For one player games
   iii. Problems for two player games
II. NMCS improvements
   i. Discounting
   ii. Pruning
      - Cut on Win
      - Pruning on Depth
III. Results
IV. Conclusions

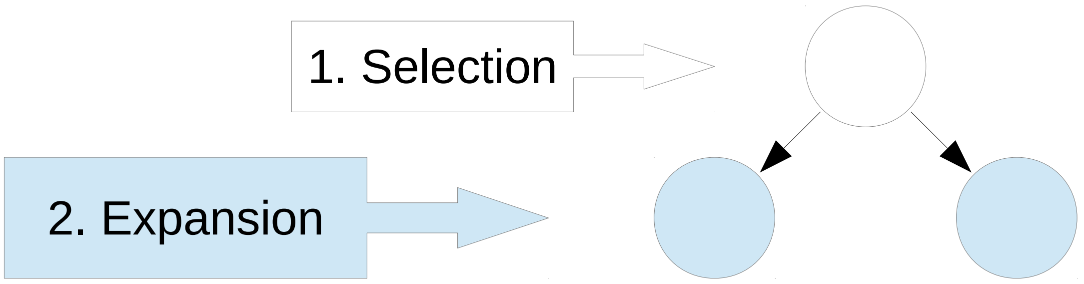
Nested Monte Carlo Tree Search (NMCS)

- What is NMCS?
- Monte Carlo Tree search:



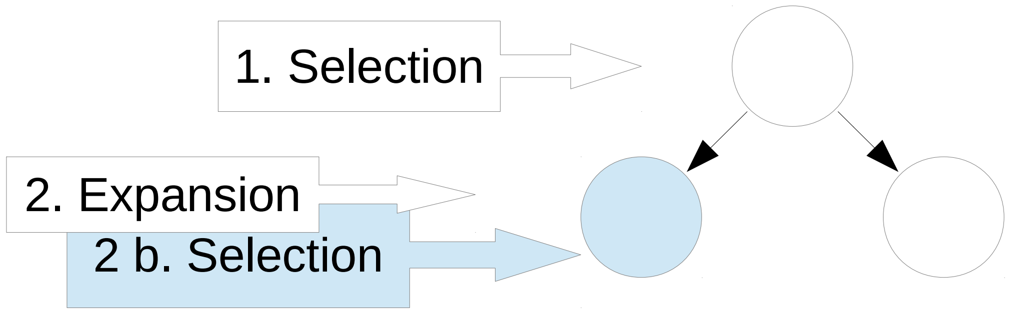
Nested Monte Carlo Tree Search (NMCS)

- What is NMCTS?
- Monte Carlo Tree search:



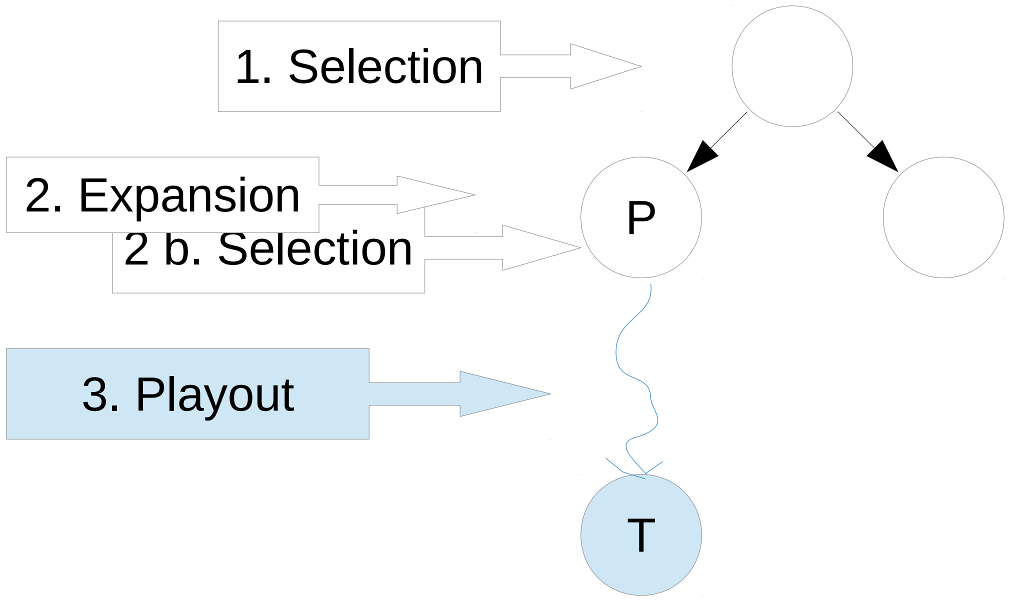
Nested Monte Carlo Tree Search (NMCS)

- What is NMCTS?
- Monte Carlo Tree search:



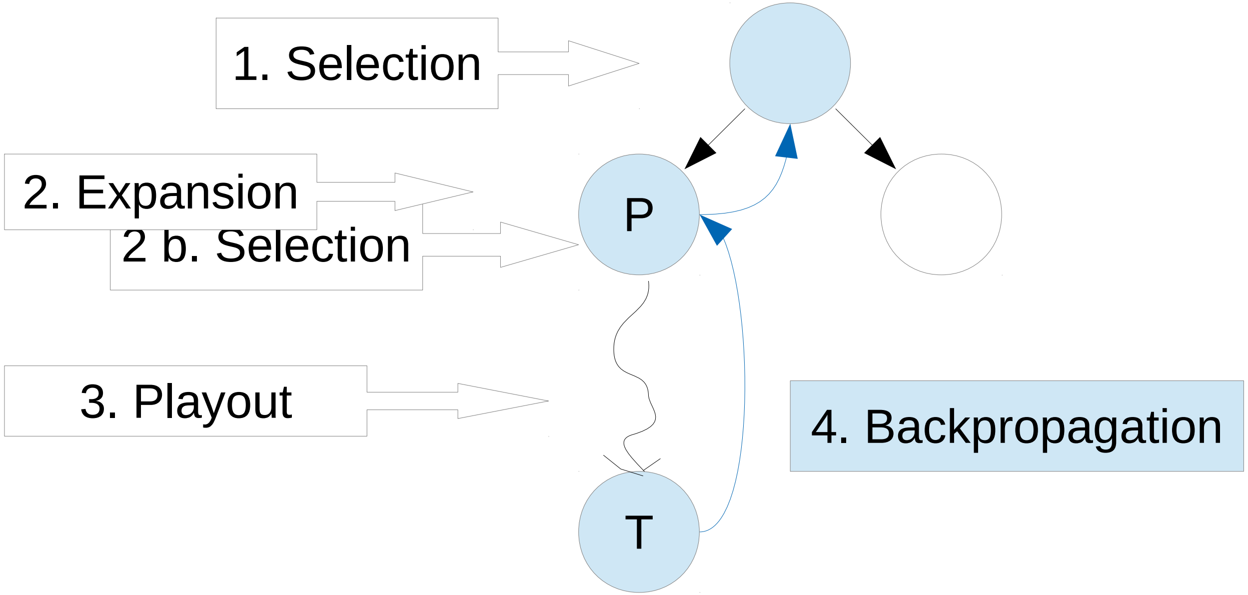
Nested Monte Carlo Tree Search (NMCS)

- What is NMCTS?
- Monte Carlo Tree search:



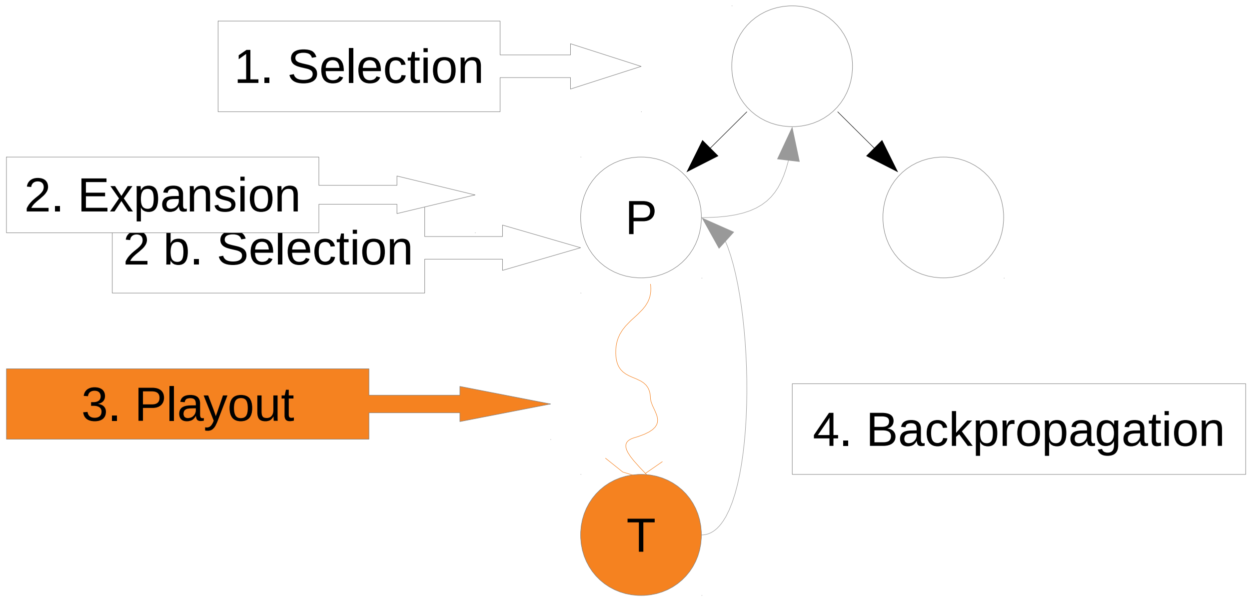
Nested Monte Carlo Tree Search (NMCS)

- What is NMCTS?
- Monte Carlo Tree search:



Nested Monte Carlo Tree Search (NMCS)

- What is NMCTS?
- Monte Carlo Tree search:

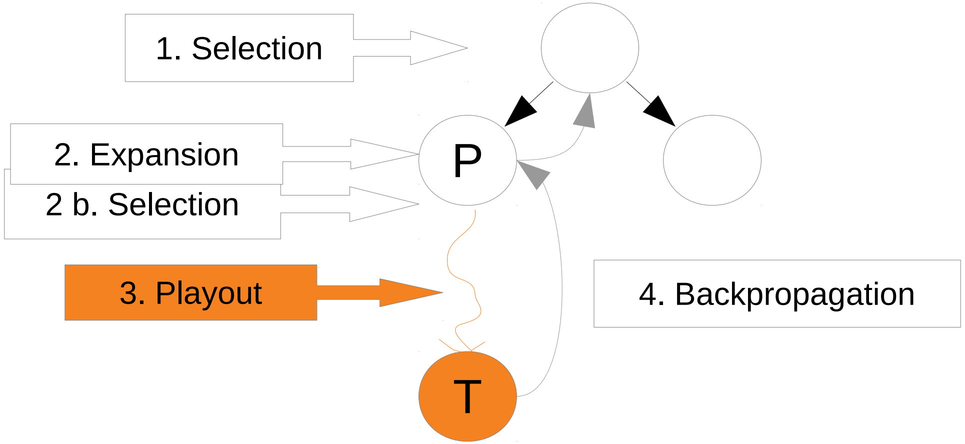


Nested Monte Carlo Tree Search (NMCS) : Definition

Playout function:

NMC(0)
→ Monte Carlo Tree Search

NMC(n+1) = $P_{\pi(V(NMC(n)))}$



n	nesting level
π	policy: tries to find best path from current node
P_π	playout function: get sequence of states according to π
V(NMC(n))	evaluation function: get terminal state value by using NMC(n)

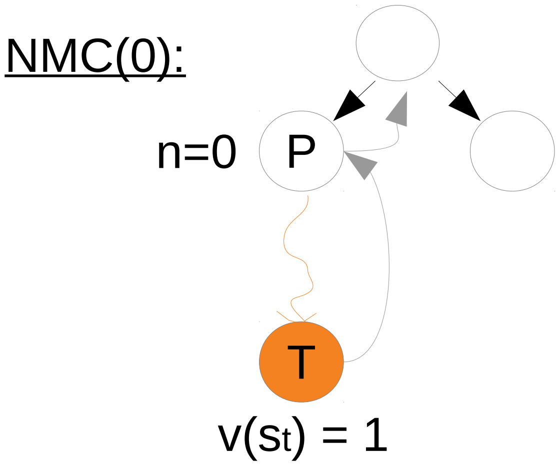
Nested Monte Carlo Tree Search (NMCS) : Definition

n=0

P

Algorithm 1: Two-player two-outcome NMCS.

```
nested (nesting n, state s, depth d, bound λ)
   while s ∉ T do
      s* ← rand(δ(s))

      s ← s*
      d ← d + 1

   return v(s)   1
```



Nested Monte Carlo Tree Search (NMCS) : Definition

Start with n=1, state s=P

Algorithm 1: Two-player two-outcome NMCS.

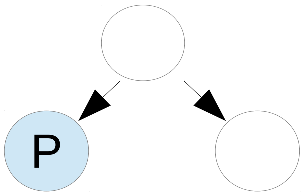
```
nested (nesting n, state s, depth d, bound λ)
    while s ∉ T do
        s* ← rand(δ(s))
        if τ(s) = max then l* ← -1/d else l* ← 1/d

        if n > υ then
            foreach s' in δ(s) do
                l ← nested (n − 1, s', d + 1, l*)
                if τ(s){l, l*} ≠ l* then s* ← s'; l* ← l

        s ← s*
        d ← d + 1

    return v(s)
```

Bad value for a terminal value

$$NMC(0+1) = P_{\pi(V(NMC(0)))}$$

<u>NMC(1):</u>

n=1



Nested Monte Carlo Tree Search (NMCS) : Definition

Start with n=1, state s=P, d=4

Algorithm 1: Two-player two-outcome NMCS.

```
nested (nesting n, state s, depth d, bound λ)
  while s ∉ T do
    s* ← rand(δ(s))
    if τ(s) = max then l* ← -1/d else l* ← 1/d

    if n > 0 then
      foreach s' in δ(s) do
        l ← nested (n − 1, s', d + 1, l*)
        if τ(s){l, l*} ≠ l* then s* ← s'; l* ← l

    s ← s*
    d ← d + 1

  return v(s)
```

Bad $v(s_t)$, here -1/4

Get $v(s_t)$
for each child

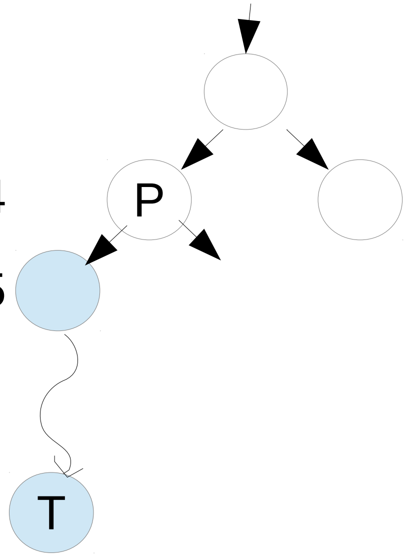
{-1/4, -1}= -1/4 → Pass

NMC(0+1) = $P_{\pi(V(NMC(0)))}$

<u>NMC(1):</u>

n=1, d=4

n=0, d=5



$v(s_t)$ = -1

Nested Monte Carlo Tree Search (NMCS) : Definition

Start with n=1, state s=P, d=4

Algorithm 1: Two-player two-outcome NMCS.

nested (*nesting* n, *state* s, *depth* d, *bound* λ)
  while $s \notin T$ **do**
    $s^* \leftarrow \mathrm{rand}(\delta(s))$
    if $\tau(s) = \max$ **then** $l^* \leftarrow \frac{-1}{d}$ **else** $l^* \leftarrow \frac{1}{d}$

    if $n > \upsilon$ **then**
      foreach s' **in** $\delta(s)$ **do**
        $l \leftarrow \texttt{nested}(n-1, s', d+1, l^*)$
        if $\tau(s)\{l, l^*\} \neq l^*$ **then** $s^* \leftarrow s'$; $l^* \leftarrow l$
        {-1/4, 1} = 1 → s* = c2, l* = 1
    $s \leftarrow s^*$
    $d \leftarrow d + 1$

  return $v(s)$

Bad v(s_t), here -1/4

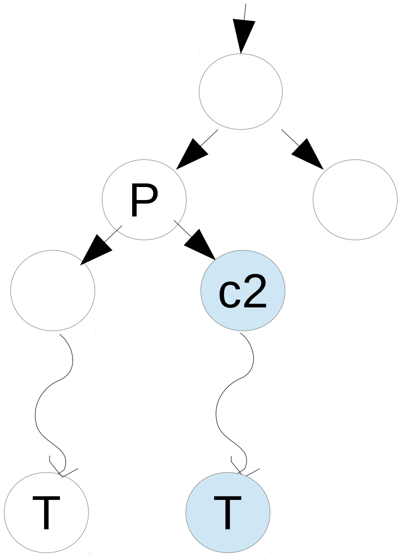
Get v(s_t)
for each child

NMC(0+1) = $P_{\pi(V(NMC(0)))}$

<u>NMC(1):</u>

n=1, d=4

n=0, d=5



v(s_t) = -1   v(s_t) = 1

Nested Monte Carlo Tree Search (NMCS) : Definition

Start with n=1, state s=P, d=4

Algorithm 1: Two-player two-outcome NMCS.

nested (*nesting* n, *state* s, *depth* d, *bound* λ)
  while $s \notin T$ **do**
    $s^* \leftarrow \text{rand}(\delta(s))$
    if $\tau(s) = \max$ **then** $l^* \leftarrow \frac{-1}{d}$ **else** $l^* \leftarrow \frac{1}{d}$

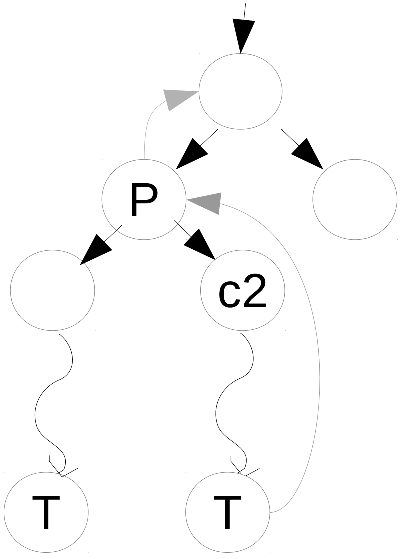
    if $n > 0$ **then**
      foreach s' **in** $\delta(s)$ **do**
        $l \leftarrow \texttt{nested}(n-1, s', d+1, l^*)$
        if $\tau(s)\{l, l^*\} \neq l^*$ **then** $s^* \leftarrow s'; l^* \leftarrow l$
        {-1/4, 1} = 1 → s* = c2, l* = 1
    $s \leftarrow s^*$
    $d \leftarrow d + 1$

  return $v(s)$

Bad v(st), here -1/4

Get v(st)
for each child

NMC(0+1) = $P_{\pi(V(NMC(0)))}$



NMC(1):

n=1, d=4

n=0, d=5

v(st) = -1   v(st) = 1

Nested Monte Carlo Tree Search (NMCS) : Definition

Start with n=2, state s=P

Algorithm 1: Two-player two-outcome NMCS.

nested (*nesting* n, *state* s, *depth* d, *bound* λ)
  while $s \notin T$ **do**
    $s^* \leftarrow \text{rand}(\delta(s))$
    if $\tau(s) = \max$ **then** $l^* \leftarrow \frac{-1}{d}$ **else** $l^* \leftarrow \frac{1}{d}$

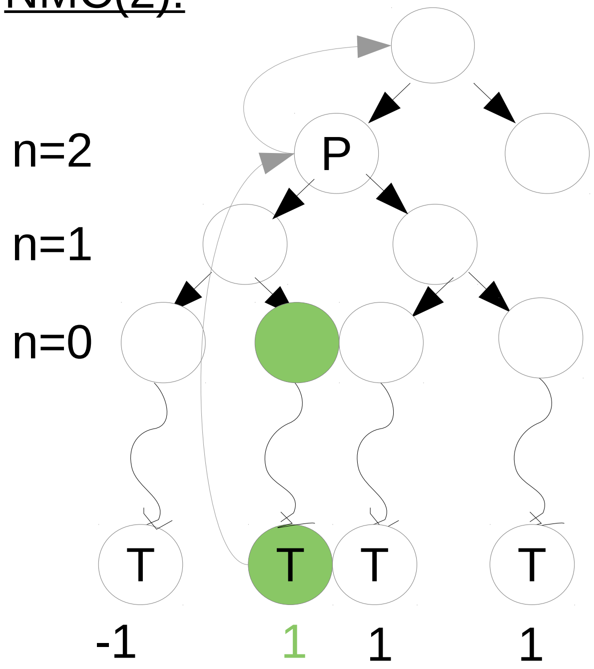
    if $n > \upsilon$ **then**
      foreach s' **in** $\delta(s)$ **do**
        $l \leftarrow$ nested $(n-1, s', d+1, l^*)$
        if $\tau(s)\{l, l^*\} \neq l^*$ **then** $s^* \leftarrow s'; l^* \leftarrow l$

    $s \leftarrow s^*$
    $d \leftarrow d + 1$

  return $v(s)$

NMC(1+1) = $P_{\pi(V(NMC(1)))}$

<u>NMC(2):</u>



Outline

I. **Nested Monte Carlo Tree Search (NMCS)**
   i. definition
   ii. **For one player games**
   iii. Problems for two player games

II. NMCS improvements
   i. Discounting
   ii. Pruning
      - Cut on Win
      - Pruning on Depth

III. Results

IV. Conclusions

NMCS: For one player games

- Same author Tristan Cazenave had idea for NMCS for single player games (2009)
- e. g. 16x16 Sudoku:
  - For each cell 16 possible values (domain)
  - Cell gets value → cells of same row/column get updated and value removed from their domain
  - Current cell always cell with smallest domain
  - Score: depth – # cells to of which values assigned
  - The deeper the better

Outline

I. **Nested Monte Carlo Tree Search (NMCS)**
   i. definition
   ii. For one player games
   iii. **Problems for two player games**

II. NMCS improvements
   i. Discounting
   ii. Pruning
      - Cut on Win
      - Pruning on Depth

III. Results

IV. Conclusions

NMCS: Problems for two player games

Single player	Two players
prefering long paths	win as fast as possible
wide range of scores	scores={-1;0;1}→which winning path better than the other one?
Save best sequence for faster execution and search improvement	MIN player as information lack for MAX player
Unlimited time	with each nesting level exponential increase of algorithm

Keep perfromance of NMCS without increasing the complexity to much!

Outline

I. Nested Monte Carlo Tree Search (NMCS)
   i. definition
   ii. For one player games
   iii. Problems for two player games
II. NMCS improvements
   i. Discounting
   ii. Pruning
      - Cut on Win
      - Pruning on Depth
III. Results
IV. Conclusions

NMCS improvements: Discounting

Aims of the two player game:

- Win as fast as possible such that opponent has less options of escaping
- Loosing as slowly as possible such that player has more chances of escaping

→ Express it algoritimically:

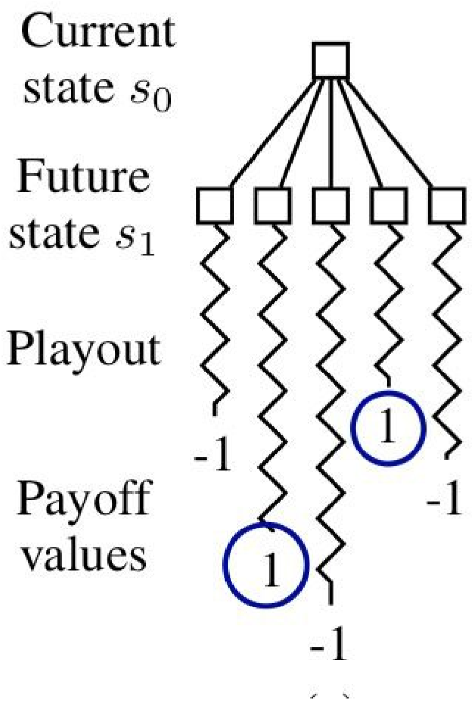
score of terminal state s_t:

$$V_D(P_\pi, s) = \frac{v(s_t)}{t+1}$$

moves from current state s to s_t

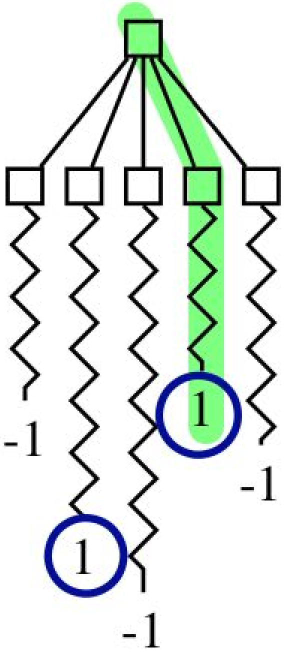
NMCS improvements: Discounting

NMC(1):



$V(P_\pi, s) = 1$

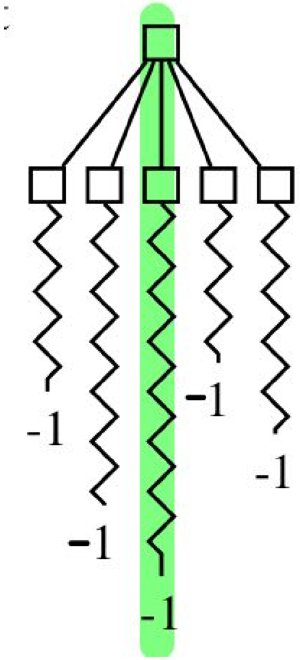
NMC$_D$(1):



$V_D(P_\pi, s) = \frac{1}{\text{Smallest\_t}}$ } Closest to 1 → Best value

NMCS improvements: Discounting

$NMC_D(1)$:



$$V_D(P_\pi, s) = \frac{-1}{Largest\_t} \Bigg\} \begin{array}{l} \text{Closest to 0} \\ \rightarrow \text{Best value} \end{array}$$

NMCS improvements: Discounting

- ✔ Better move selection
- ✗ No influence on complexity

Outline

I. Nested Monte Carlo Tree Search (NMCS)
   i. definition
   ii. For one player games
   iii. Problems for two player games

II. NMCS improvements
   i. Discounting
   ii. Pruning
      - **Cut on Win**
      - Pruning on Depth

III. Results

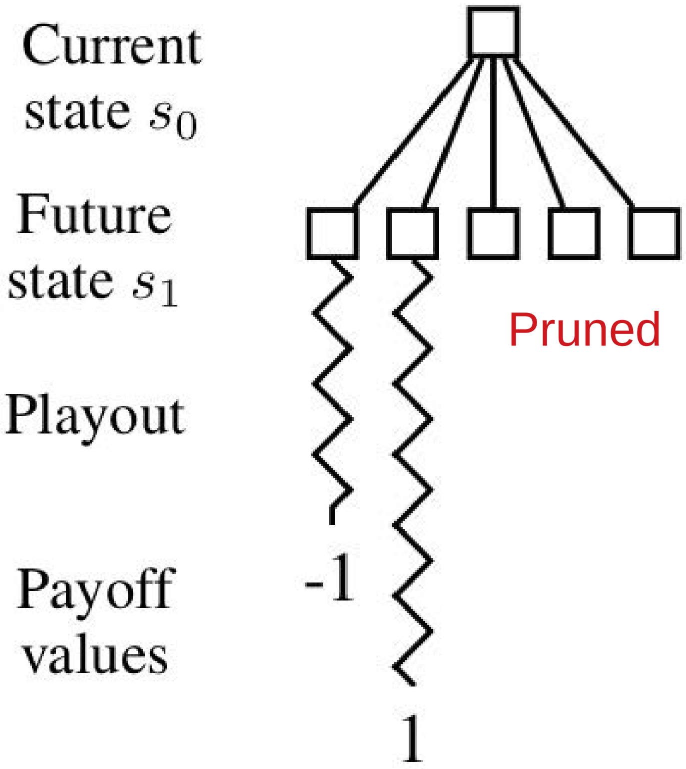
IV. Conclusions

NMCS improvements: Pruning → Cut on Win (COW)

- $\pi \rightarrow \pi_{\text{COW}}$
- Stop with NMCS when terminal state with value 1 found!
  - ✗ Discounting can‘t be used in combination

✗ Aim of fast wins ignored

✓ Complexity decreased



Outline

I. Nested Monte Carlo Tree Search (NMCS)
   i. definition
   ii. For one player games
   iii. Problems for two player games

II. NMCS improvements
   i. Discounting
   ii. Pruning
      - Cut on Win
      - **Pruning on Depth**

III. Results

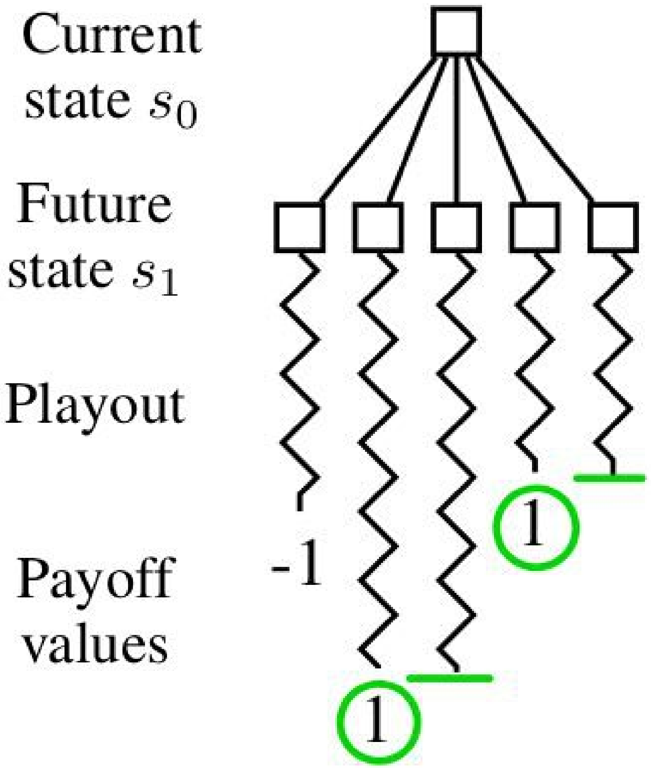
IV. Conclusions

NMCS improvements: Pruning → Pruning on Depth (POD)

- All playouts skipped which are longer than the playout of the shortest winner path found so far
  - ✔ Discounting used

✔ Shortest wins considered

✔ Complexity decreased



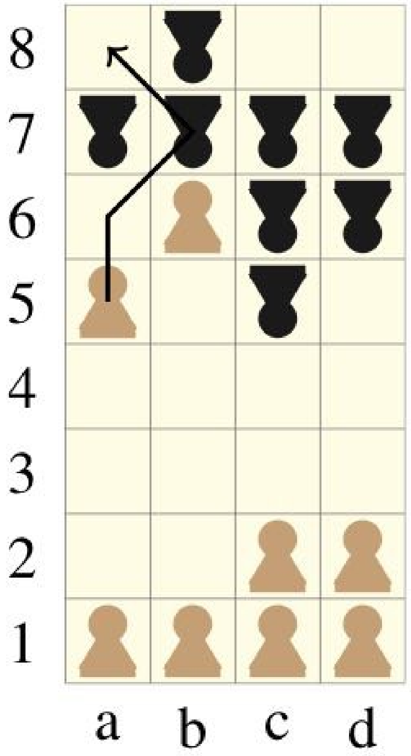
Outline

I. Nested Monte Carlo Tree Search (NMCS)
   i. definition
   ii. For one player games
   iii. Problems for two player games
II. NMCS improvements
   i. Discounting
   ii. Pruning
      - Cut on Win
      - Pruning on Depth
III. Outcoming algorithm
IV. Results
V. Conclusions

Results: Breakthrough

Rules:

- Move one piece straight/diagnoal forward when there are empty places
- When opponent diagnoally in front of a piece that piece can go to the opponent‘s one and replace it
- Aim: reach opponent‘s home row
- Draws are not possible



Results: Breakthrough

NMC(3):

Discounting	Pruning	States Visited(k)	Freq(%)
No	None	$4,459 \pm 27$	11.9 ± 2.2
No	$\text{COW}(\leq 1)$	$1,084 \pm 8$	12.3 ± 2.6
No	$\text{COW}(\leq 2)$	214 ± 2	10.9 ± 2.0
No	$\text{COW}(\leq 3)$	25 ± 1	9.8 ± 2.0
Yes	None	$2,775 \pm 26$	64.1 ± 3.4
Yes	$\text{POD}(\leq 1)$	$1,924 \pm 20$	64.7 ± 3.5
Yes	$\text{POD}(\leq 2)$	$1,463 \pm 16$	58.6 ± 3.5
Yes	$\text{POD}(\leq 3)$	627 ± 19	62.4 ± 3.3

(a) (b) (c) (b) (c)

(a) Activation in nesting level i or lower
(b) Average over 900 runs
(c) Confidence interval

Results: Breakthrough

NMC(3):

Discounting	Pruning	States Visited(k)	Freq(%)
No	None	$\boxed{4,459} \pm 27$	$\boxed{11.9} \pm 2.2$
No	COW(≤ 1)	$1,084 \pm 8$	12.3 ± 2.6
No	COW(≤ 2)	214 ± 2	cow 10.9 ± 2.0
No	COW(≤ 3)	cow $\boxed{25} \pm 1$	9.8 ± 2.0
Yes	None	$2,775 \pm 26$	64.1 ± 3.4
Yes	POD(≤ 1)	$1,924 \pm 20$	64.7 ± 3.5
Yes	POD(≤ 2)	$1,463 \pm 16$	58.6 ± 3.5
Yes	POD(≤ 3)	627 ± 19	62.4 ± 3.3

(a) (b) (c) (b) (c)

(a) Activation in nesting level i or lower
(b) Average over 900 runs
(c) Confidence interval

Results: Breakthrough

NMC(3):

Discounting	Pruning	States Visited(k)	Freq(%)
No	None	$4,459 \pm 27$	11.9 ± 2.2
No	COW($\leq$ 1)	$1,084 \pm 8$	12.3 ± 2.6
No	COW($\leq$ 2)	214 ± 2	10.9 ± 2.0
No	COW($\leq$ 3)	25 ± 1	9.8 ± 2.0
Yes	None	$\boxed{2,775} \pm 26$	$\boxed{64.1} \pm 3.4$
Yes	POD($\leq$ 1)	$1,924 \pm 20$	POD $\boxed{64.7} \pm 3.5$
Yes	POD($\leq$ 2)	$1,463 \pm 16$	$\boxed{58.6} \pm 3.5$
Yes	POD($\leq$ 3)	POD $\boxed{627} \pm 19$	$\boxed{62.4} \pm 3.3$
	(a)	(b) (c)	(b) (c)

(a) Activation in nesting level i or lower
(b) Average over 900 runs
(c) Confidence interval

Results: Breakthrough

Table 4: Win percentages of NMCS against a standard MCTS player for various settings and thinking times.

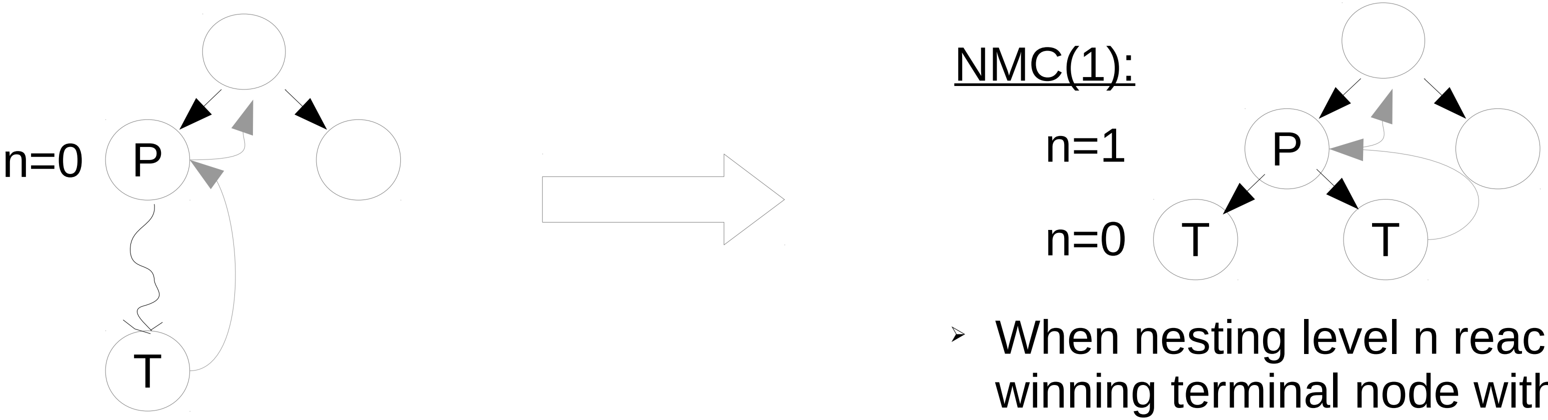
Game		n	COW	POD	10ms	20ms	40ms	80ms	160ms	320ms
Breakthrough		1			3.2	6.0	12.0	11.6	7.8	6.4
		1		✔	27.6	22.6	16.8	21.6	15.4	20.4
		1	✔		22.6	25.2	30.4	34.6	35.2	39.6
		2	✔		4.6	2.0	2.4	1.4	2.4	3.8
	misère	1			85.4	83.4	70.2	60.8	57.0	56.4
		1		✔	91.4	95.6	97.0	97.8	98.8	98.8
		1	✔		95.2	95.2	98.0	99.0	99.8	**99.8**
		2	✔		1.0	27.6	43.6	87.0	93.2	95.6

misère:
same game with reverse winning condition

Results: Breakthrough

Why NMCS better in misère version?

NMCS important in games where the last moves are important.



- When nesting level n reaches the depth of winning terminal node with only winning siblings then win.
  It is fully included in the MINMAX tree.

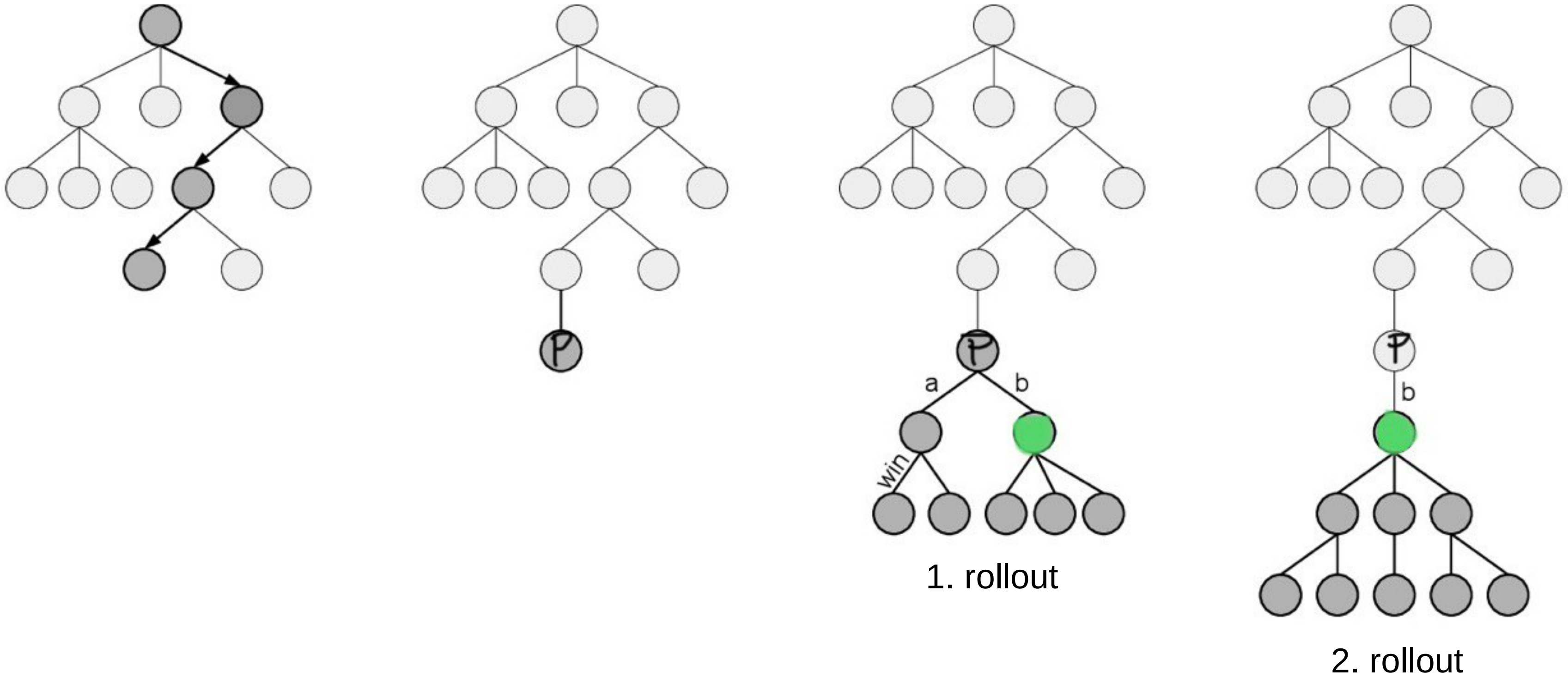
Results: Discounting effect

Winrates (%) of NMCS with discounting vs. NMCS without it for nesting levels 0 to 2 and game engine speed.

Game	Nesting Level				
	0		1		2
Breakthrough	79.6	<	99.6	>	99.4
misère	42.4	<	80.8	<	90.0
Knightthrough	78.6	<	100.0	=	100.0
misère	46.0	<	83.2	<	85.8
Domineering	71.2	<	77.0	<	83.8
misère	43.4	<	63.2	<	68.4
NoGo	62.8	<	76.4	<	83.4
misère	53.2	<	65.6	<	67.2
AtariGo	69.6	<	97.2	<	100.0

Results: NMCS vs. MCTS-MR

MCTS with Minimax Rollouts:



semanticscholar.org/paper/MCTS-Minimax-Hybrids-Baier-Winands/9e62cc6e2c6b3844341e8ebc8688b3598ef6a69c/figure/0

Results: NMCS vs. MCTS-MR

Win percentages of NMCS and MCTS-MR against standard MCTS with 320ms per move. NMCS uses COW and depth 1 while MCTS-MR uses depth 1 and 2.

Game	NMCS	MCTS-MR	
		MR 1	MR 2
Breakthrough	39.6	47.4	**50.2**
misère	**99.8**	49.4	48.6
Knightthrough	49.6	50.0	50.0
misère	**98.6**	49.8	45.0
Domineering	50.0	50.0	49.8
misère	**58.6**	46.0	44.2
NoGo	48.0	**59.4**	50.2
misère	60.8	54.2	**67.4**
AtariGo	**77.2**	44.0	47.0

Outline

Conclusions

- Discounting as tool for better move selection
- COW and POD as policies for reducing the search effort
- NMCS can deal with games in which the last steps are the curcial ones
- NMCS not always outperfoms MCTS

References

Hendrik Baier, Mark H. M. Winands: Monte-Carlo tree search and minimax hybrids. CIG 2013: 1-8

Tristan Cazenave, Abdallah Saffidine, Michael John Schofield, Michael Thielscher: Nested Monte Carlo Search for Two-Player Games. AAAI 2016: 687-693

Tristan Cazenave: Nested Monte-Carlo Search. IJ-CAI 2009: 456-461

Exkursus: Soduko

2		5	3		8	4		9
	7						5	
9		4				6		7
5				4				2
			5		7			
6				3				8
4		6				8		1
	2						6	
8		1	2		9	7		4

https://sudoku-club.de/produkt/1000-sudokus-9x9-schwer/