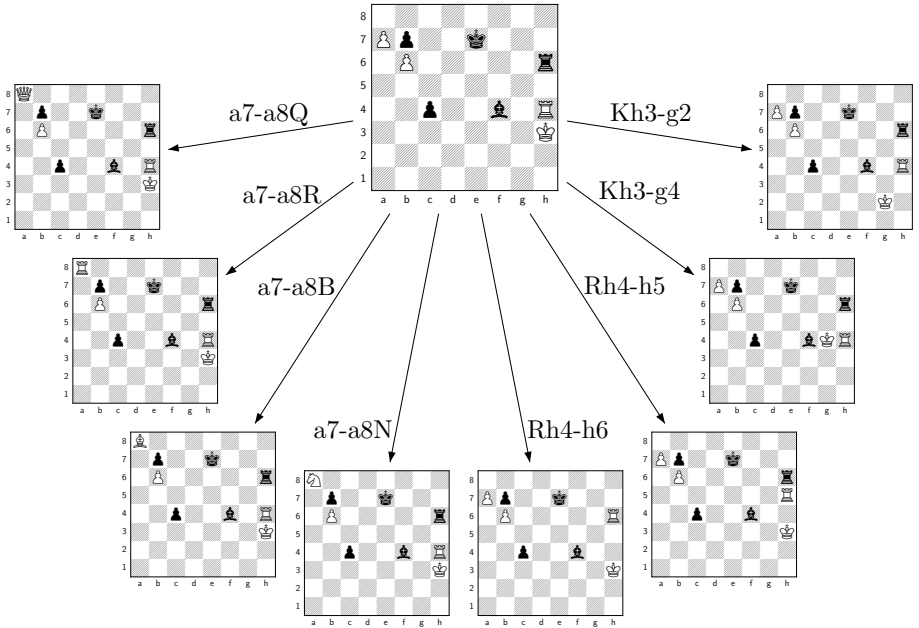
Split Moves for Monte-Carlo Tree Search

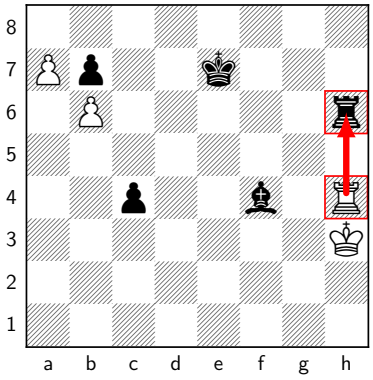
Jakub Kowalski[1], Maksymilian Mika[1], *Wojciech Pawlik*[1],
Jakub Sutowicz[1], Marek Szykuła[1],
Mark H. M. Winands[2]

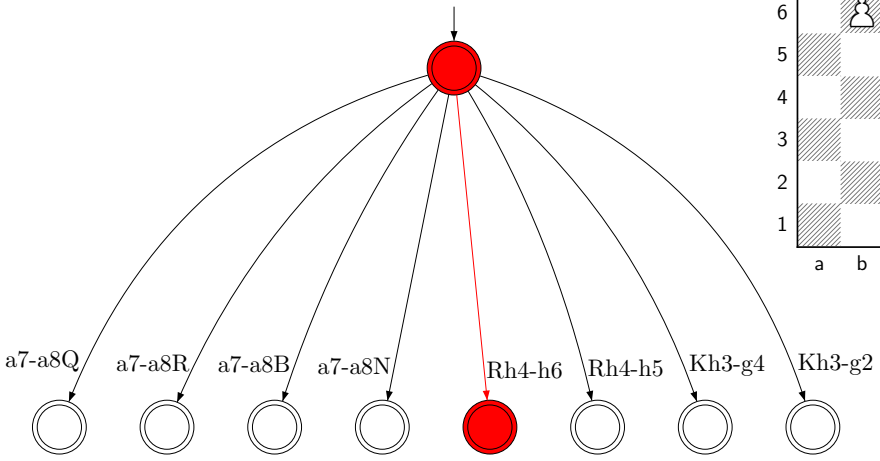
[1]University of Wrocław, Faculty of Mathematics and Computer Science
[2]Maastricht University, Department of Data Science and Knowledge Engineering

Game Tree

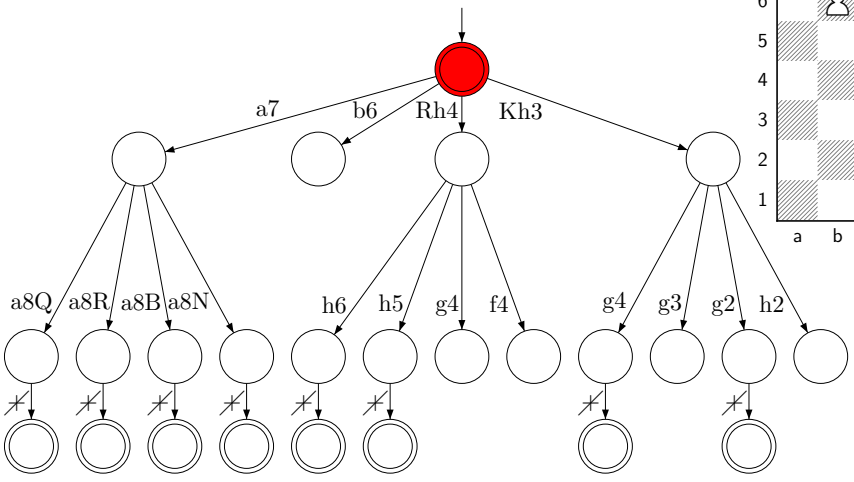


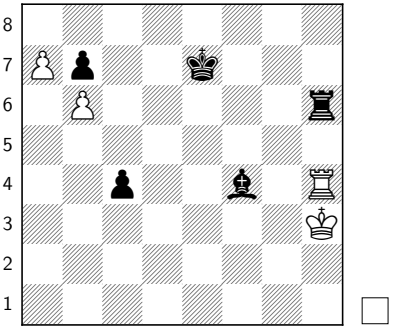
Orthodox move



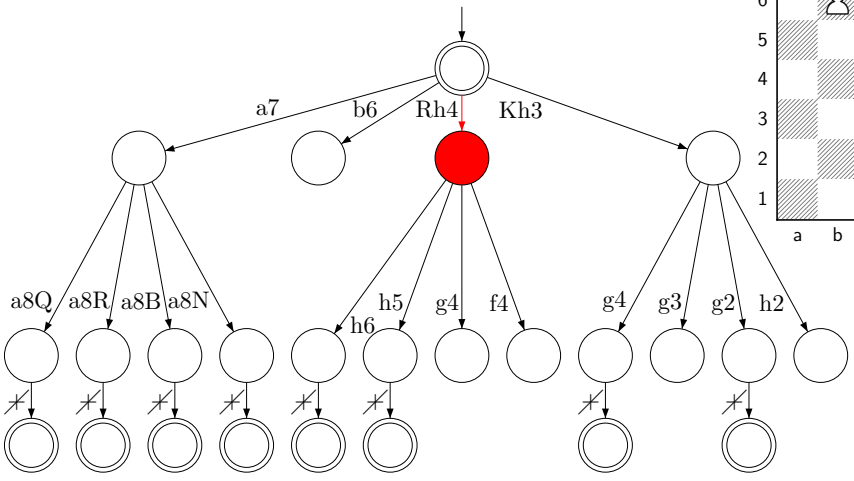


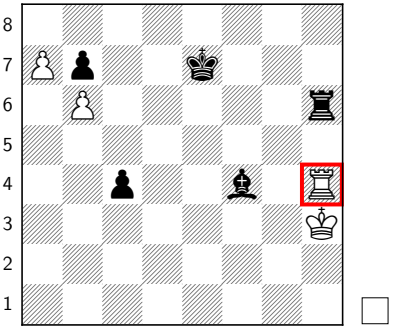
Split move: *Mod* split strategy



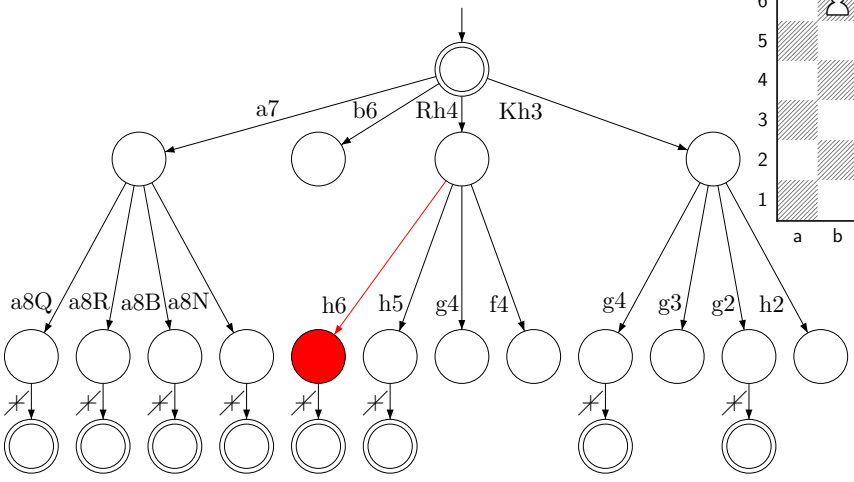


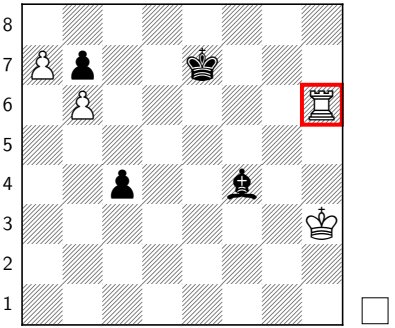
Split move: *Mod* split strategy



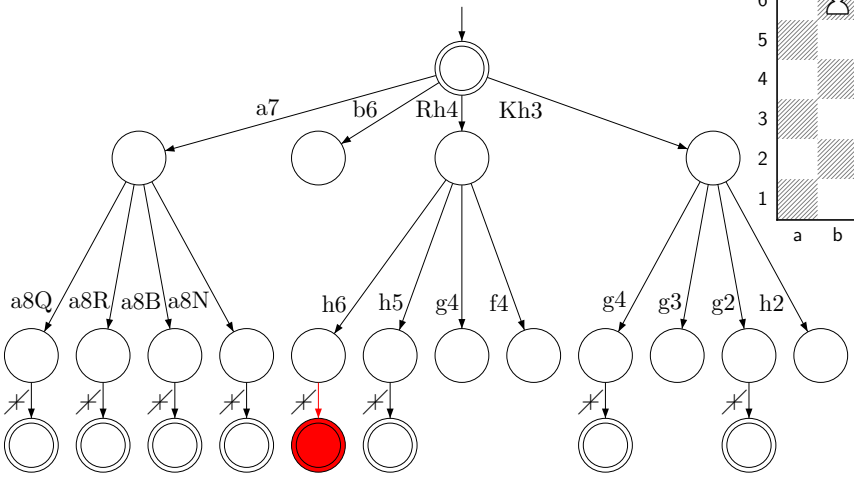


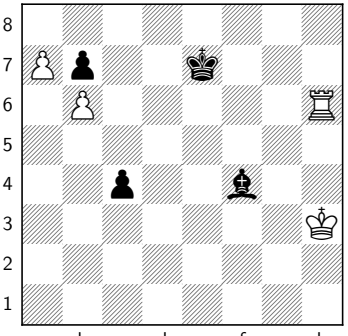
Split move: *Mod* split strategy



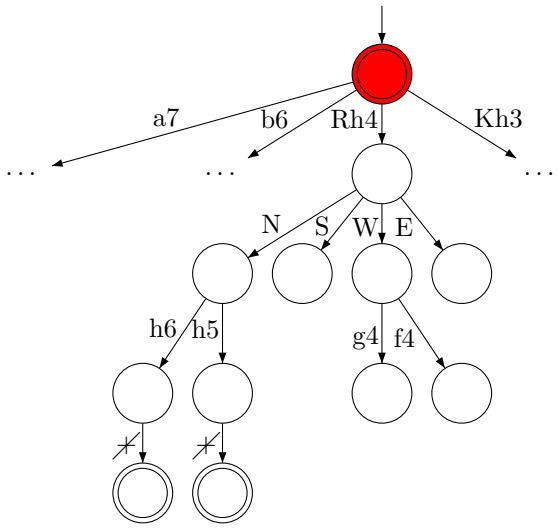


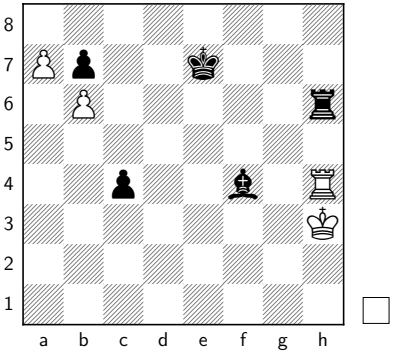
Split move: *Mod* split strategy



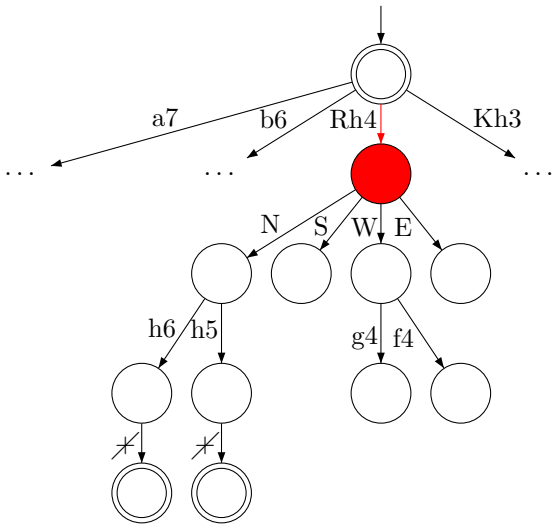


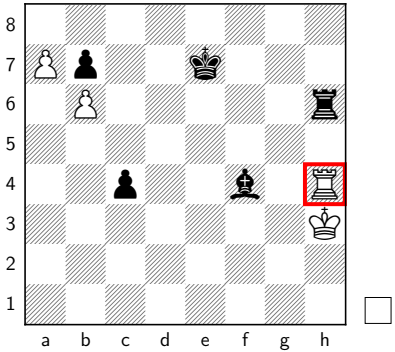
Split move: *ModPlus* split strategy



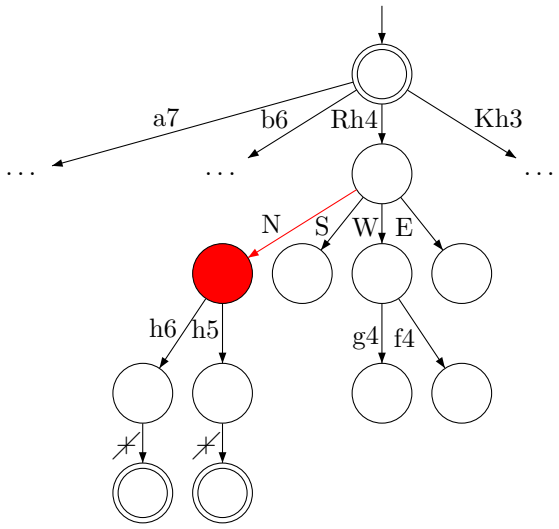


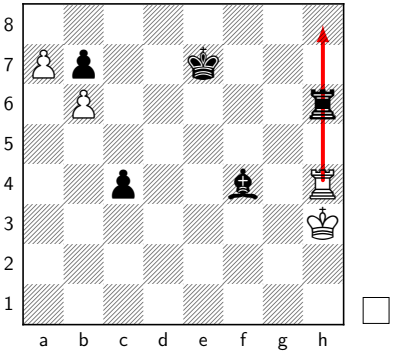
Split move: *ModPlus* split strategy



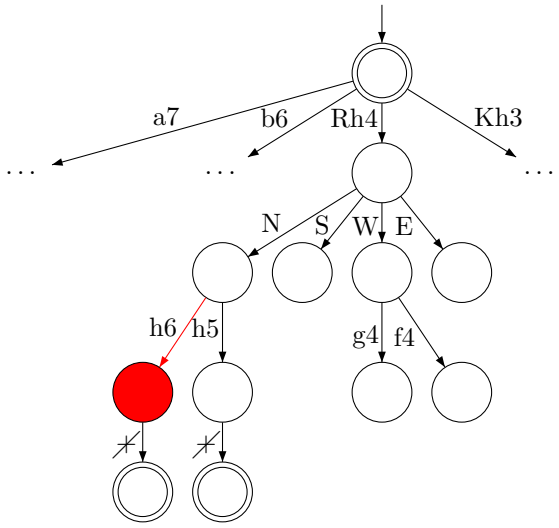


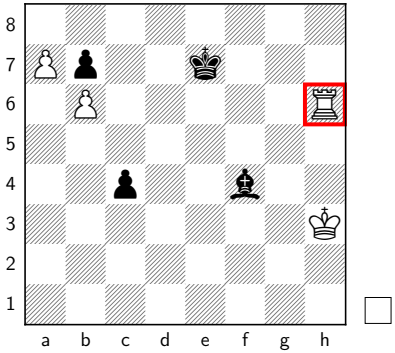
Split move: *ModPlus* split strategy



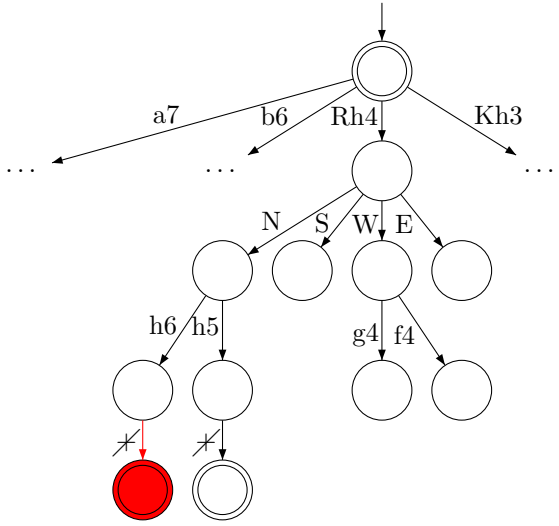


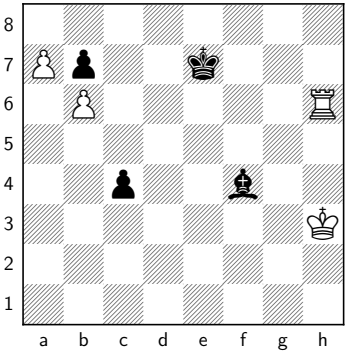
Split move: *ModPlus* split strategy



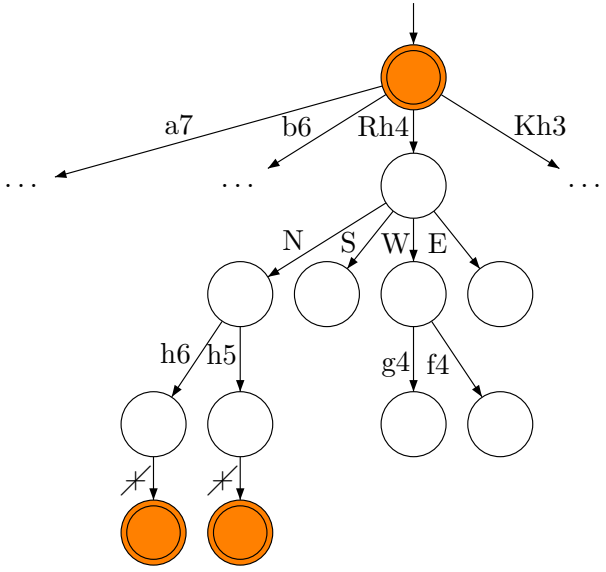


Split move: *ModPlus* split strategy

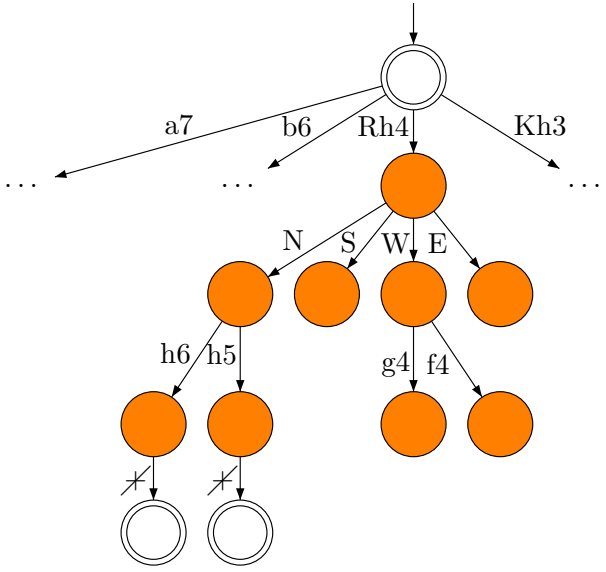




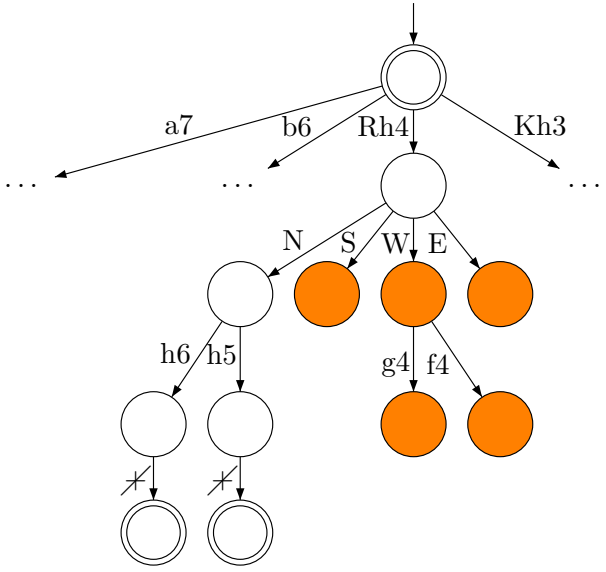
Nodal states



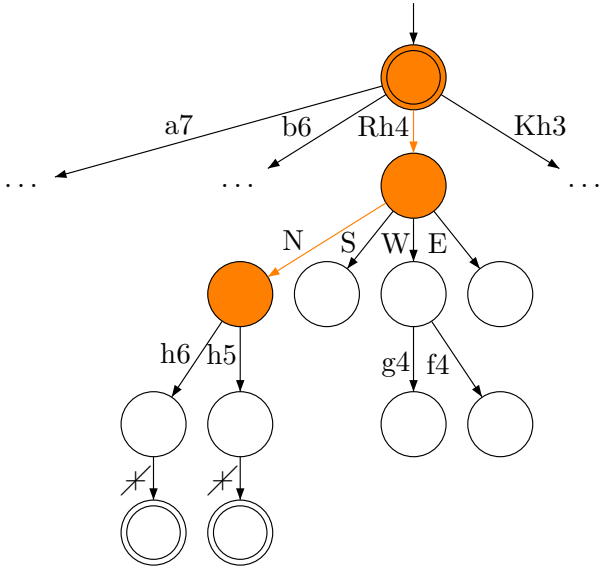
Intermediate states



Dead states



Common prefix of moves: Rh4h6 and Rh4h5



Motivation and applicability

- Applicable to *any* game-playing algorithm: Monte-Carlo Tree Search, Min-Max, evolutionary search, neural networks, ...

Except simple cases, the method and its effects were not previously investigated in the literature.

- Improve efficiency.
- Reduce branching factor.
- Share information between moves.

Motivation and applicability

- Applicable to *any* game-playing algorithm: Monte-Carlo Tree Search, Min-Max, evolutionary search, neural networks, ...

Except simple cases, the method and its effects were not previously investigated in the literature.

- Improve efficiency.
- Reduce branching factor.
- Share information between moves.

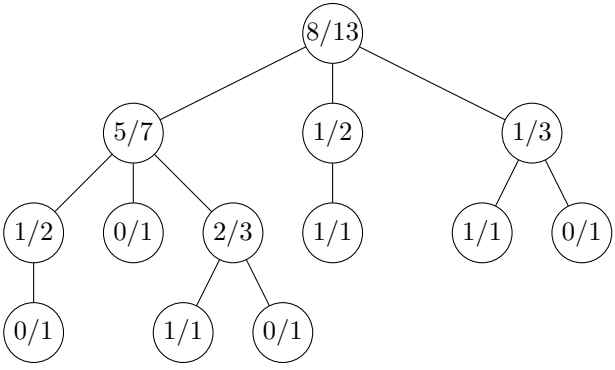
Motivation and applicability

- Applicable to *any* game-playing algorithm: Monte-Carlo Tree Search, Min-Max, evolutionary search, neural networks, ...

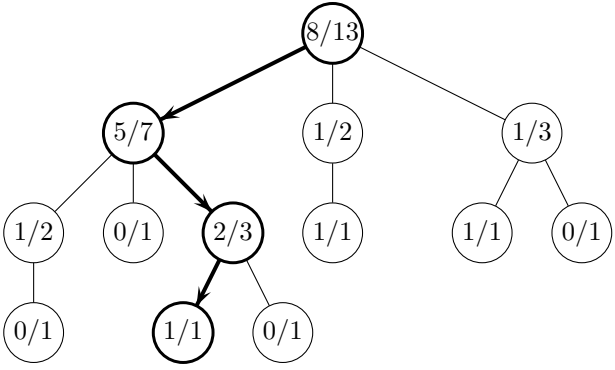
Except simple cases, the method and its effects were not previously investigated in the literature.

- Improve efficiency.
- Reduce branching factor.
- Share information between moves.

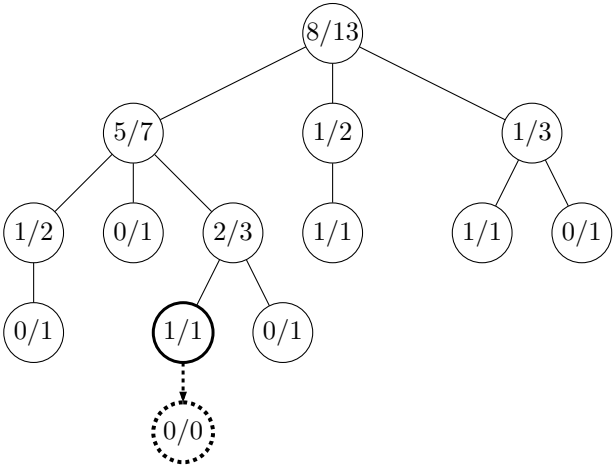
Monte Carlo Tree Search



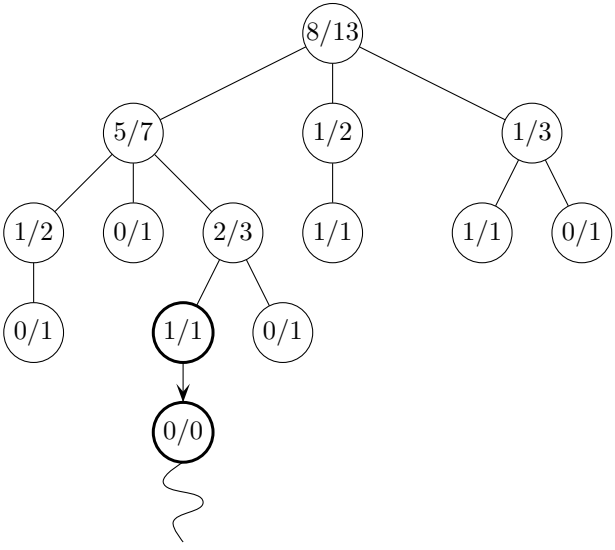
Monte Carlo Tree Search – selection



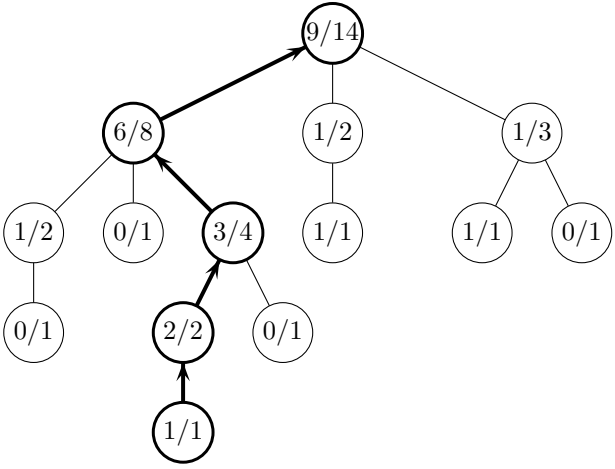
Monte Carlo Tree Search – expansion



Monte Carlo Tree Search – simulation



Monte Carlo Tree Search – backpropagation



Semisplit MCTS

- Effective handling of "dead" states.
- Different strategies in selection and simulation phases.
- Variant: *raw*/*nodal*, *roll-up* (combining semimoves dynamically), final selecton strategies, ...
- Heuristics MAST and RAVE: division and joining of moves, context and mixed variants.

Semisplit MCTS

- Effective handling of "dead" states.
- Different strategies in selection and simulation phases.
- Variant: *raw*/*nodal*, *roll-up* (combining semimoves dynamically), final selecton strategies, ...
- Heuristics MAST and RAVE: division and joining of moves, context and mixed variants.

MCTS variant		Standard		Split		Join		Context	
Tree	Simulation	MAST	RAVE	MAST	RAVE	MAST	RAVE	MAST	RAVE
orthodox	orthodox	✓[J]	✓[J]	✓	–	✓	✓	~~✓[J]~~	~~✓[J]~~
orthodox	semisplit	–	–	✓	–	–	✓	✓	~~✓[J]~~
semisplit	orthodox	–	–	✓	✓	✓[*]	–	✓	✓
semisplit	semisplit	✓[S]	✓[S]	✓	✓	–	–	✓	✓
roll-up	orthodox	–	–	✓	–	✓[*]	–	✓	✓
roll-up	semisplit	–	–	✓	–	–	–	✓	✓

Implementation

- Based on *Regular Boardgames* General Game Playing system.
- The compiler generates a reasoner with automatically split moves, according to the given *split strategy*.
- *Just-in-time* compilation which takes into account the game rules, algorithm of the agent, configuration parameters.
- Many low-level optimizations, dedicated data structures.
- Over 700 available configurations.

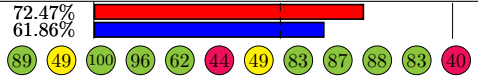
Differences in speed

Game	Orthodox MCTS		Semisplit MCTS	
	States/sec.	Avg. branching factor	Speed-up factor	Avg. branching factor
Amazons	236 269	457.40	11.42	6.91
Breakthrough	2 495 330	25.69	2.03	7.70
Breakthru	11 088	12 958.00	174.38	12.13
Chess	285 631	22.80	4.96	2.96
Chess no-check	710 881	33.22	3.89	3.92
English Draughts	4 411 795	5.22	1.09	2.50
Fox and hounds	13 940 118	4.12	0.95	2.65
Go	173 452	130.35	0.33	72.56
Knightthrough	2 159 981	37.32	2.51	8.65
Pentago	492 027	171.24	3.33	15.09
Skirmish	679 837	34.35	4.07	4.29
The Mill Game	2 298 726	14.85	1.91	4.25

Results (win ratios)

Vanilla MCTS

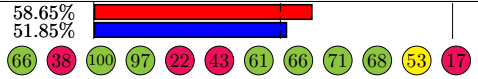
Tree:	*Semisplit nodal*
Simulation:	*Semisplit*
Split strategy:	*ModPlus*



Tree:	*Orthodox*
Simulation:	*Semisplit*
Split strategy:	*Mod*

MCTS with action-based heuristics: MAST and RAVE

Tree:	*Semisplit nodal, RAVE-split*
Simulation:	*Semisplit, MAST-split*
Split strategy:	*Mod*



Tree:	*Roll-up nodal*
Simulation:	*Semisplit, MAST-split*
Split strategy:	*Mod*

1. Amazons, 2. Breakthrough, 3. Breakthru, 4. Chess, 5. Chess (no-check), 6. English Draughts, 7. Fox and Hounds, 8. Go, 9. Knightthrough, 10. Pentago, 11. Skirmish, 12. The Mill Game

Summary

Division of moves

- Wide applicability: many problems (games) and algorithms.
- No previous studies: so far, split moves were applied only to trivial cases.

MCTS Agent

- Lots of variants.
- Highly-optimized, generic implementation.

Results

- Significant improvement of results on average.
- Highly dependent on the game. Sometimes more complex variant are required.

Future work

- Applying to other algorithms.
- How to select the best variant and split strategy for the given game?

Summary

Division of moves

- Wide applicability: many problems (games) and algorithms.
- No previous studies: so far, split moves were applied only to trivial cases.

MCTS Agent

- Lots of variants.
- Highly-optimized, generic implementation.

Results

- Significant improvement of results on average.
- Highly dependent on the game. Sometimes more complex variant are required.

Future work

- Applying to other algorithms.
- How to select the best variant and split strategy for the given game?

Thank you!