Simulation Search Control in General Game Playing

Yngvi Björnsson

Reykjavik University

Maastricht, May 2011

GGP Research Group at RU

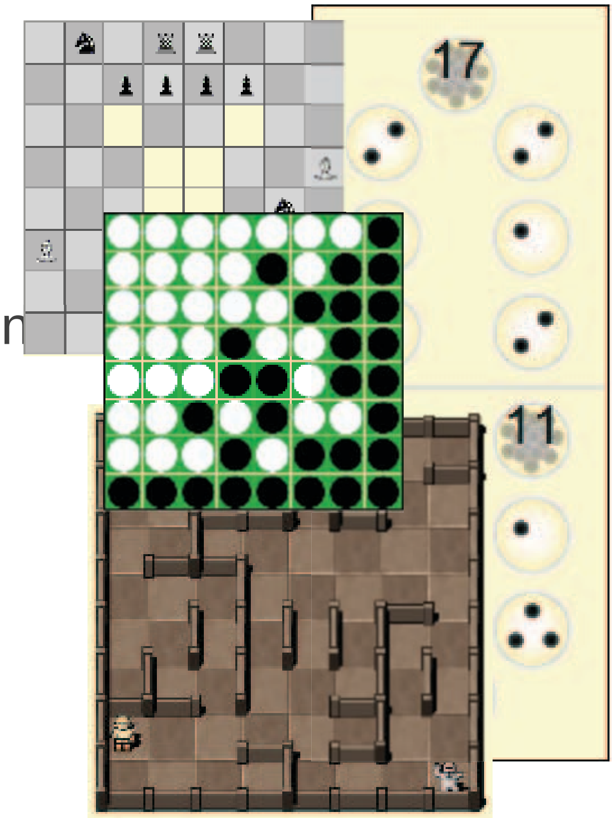
- Work described here is done by the GGP research group at RU
  - Yngvi Björnsson, Associate Professor
  - Hilmar Finnsson, PhD student
  - Stefán Freyr Guðmundsson, PhD student
  - Stephan Schiffel, Post-doc




- General Game Playing (GGP)
- CADIA Player
- Search Control Schemes
  - Selection Phase
    - UCT / RAVE
    - MA / ST / Knowledge Bias
  - Playout Phase
    - MAST / TO-MAST / PAST / FAST
  - Combined schemes
- Empirical Results
- Game Properties
- Conclusions

General Game-Playing (GGP)

- Play a wide variety of games
  - *n*-player games ($n \geq 1$)
  - Adversary, co-operative
  - Some limitations:
    - Deterministic and perfect-information
- Game rules described using
  - GDL (Game Description Lang.)
  - Logic based (Prolog-like)



Game Description Language (GDL)

- Predicate used to describe the current state:
  - (cell 1 1 blank)
    (cell 1 2 X)
    …
    (cell 3 5 O)
    (control xplayer)
- Implication rules use to describe:
  - Possible moves (**legal**)
  - How the new state looks like after a move is made (**next**)
  - If a state is terminal (**terminal**)
  - Outcome of a game (**goal**)
- Also, special keywords for listing roles etc.
  –

GDL Example for Legal Moves in TicTacToe

```
(<= (legal ?w (mark ?x ?y) )
    (true (cell ?x ?y blank))
    (true (control ?w)))
```

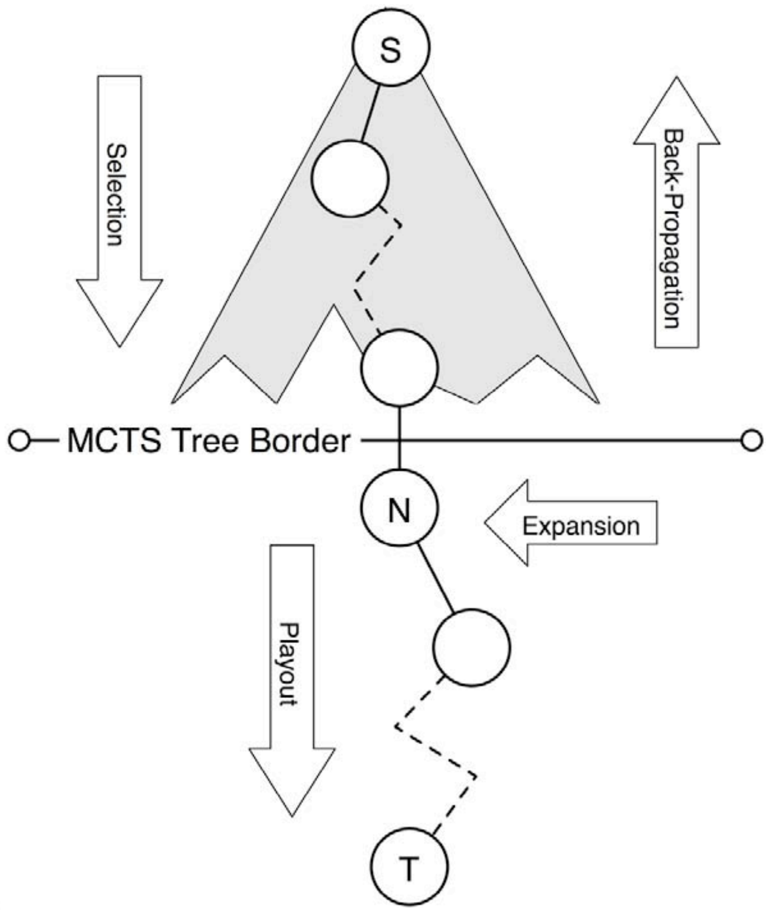
```
(<= (legal xplayer noop)
    (true (control oplayer)))
```

```
(<= (legal oplayer noop)
    (true (control xplayer)))
```

CADIA Player

- General Game-Playing Agent
  - (CADIA is the name of RU’s AI lab)
  - GGP competition
    - 1st place 2007 and 2008.
    - 6th place in 2009
    - 3rd place in 2010
- Technique:
  - MCTS based
  - Before CadiaPlayer GGP players were pre-dominantly knowledge-based alpha-beta players
  - Now, most players use MCTS

MCTS in CADIA Player



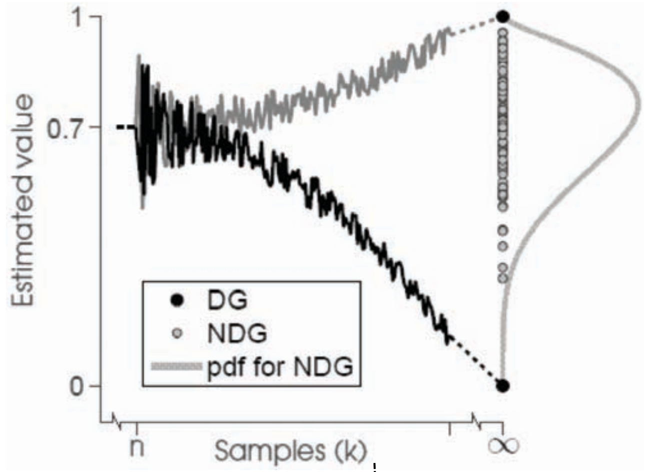
- Selection
  - UCT / RAVE
  - (+tie-breaking)
  - Selection enhancements
- Expansion
  - Add one node per simulation
- Back-propagation
  - Averaging
  - Learning / updating
- Playout
  - Using knowledge learned online

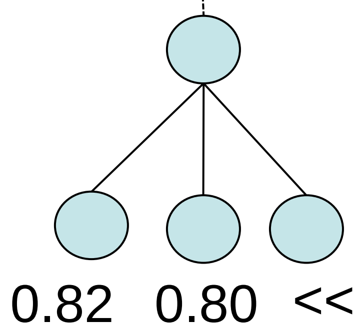
Search-Control in CADIA Player

- Selection Phase
  - UCT
  - RAVE
  - Deterministic Discrete Outcome Games
    - Moving Average (MA) / Sufficiency Threshold (ST)
  - Knowledge Bias
- Playout Phase
  - MAST (2008)
  - TO-MAST (2009)
  - PAST (2009)
  - FAST (2010)

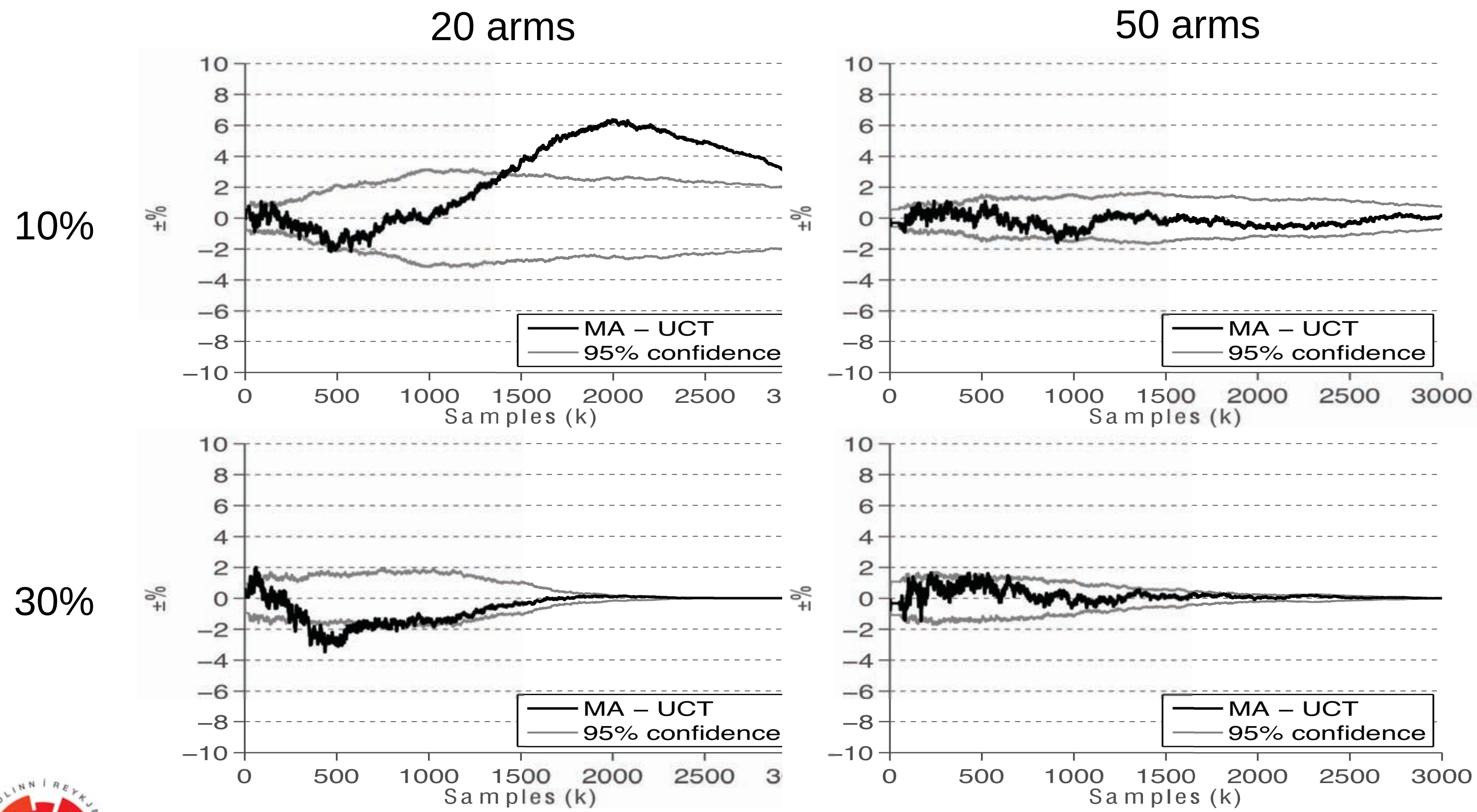
Deterministic Discrete Outcome Games

- UCT rooted in *n*-arm bandit exploration
- Can we do “better” if we know the game we are playing is
  - Deterministic / Discrete outcome
- Two possible problematic scenarios
  - Values can change drastically once wins/ losses are found, that information propagates slowly. = MA
  - Effort distinguishing between two likely “winning moves”. = ST
- Ran simulations using *n*-arm bandits
  - Each arm random walk to outcomes 0 or 1





Moving Average (MA)



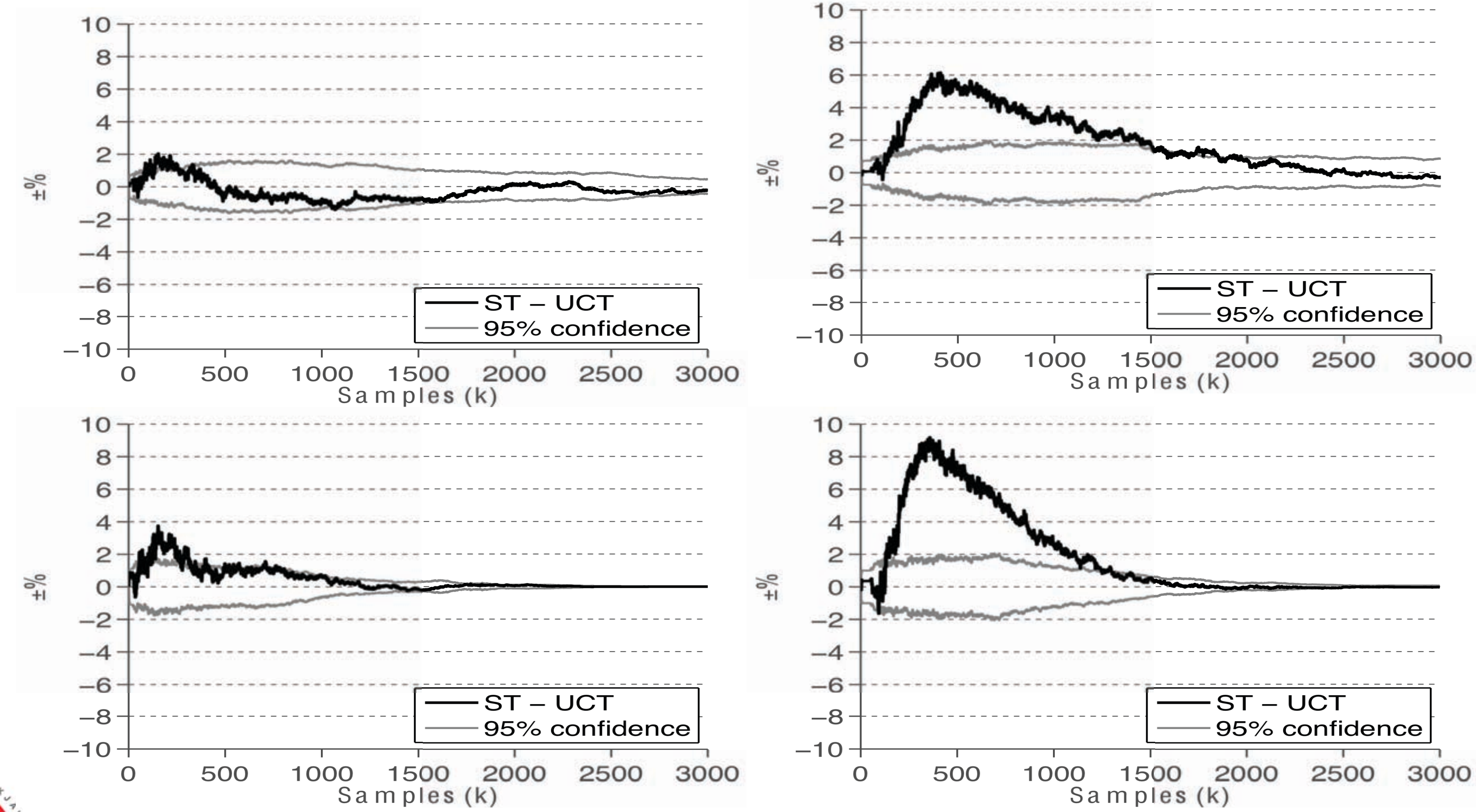
Sufficiency Threshold (ST)

20 arms

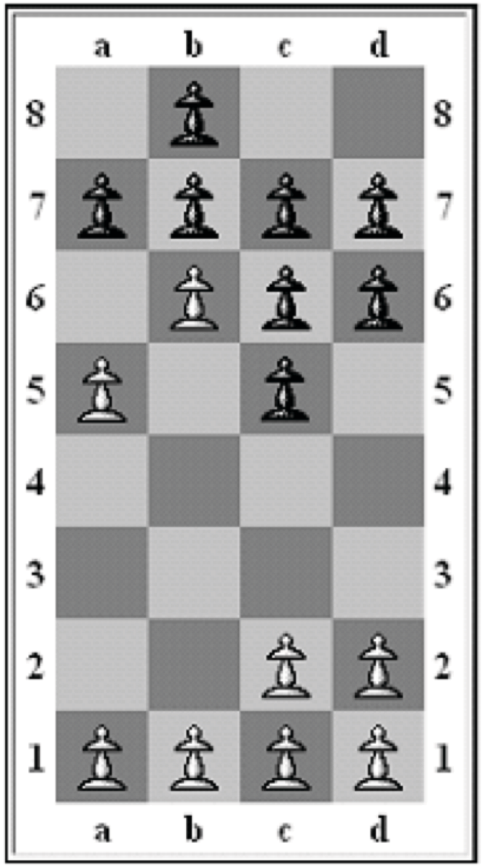
50 arms

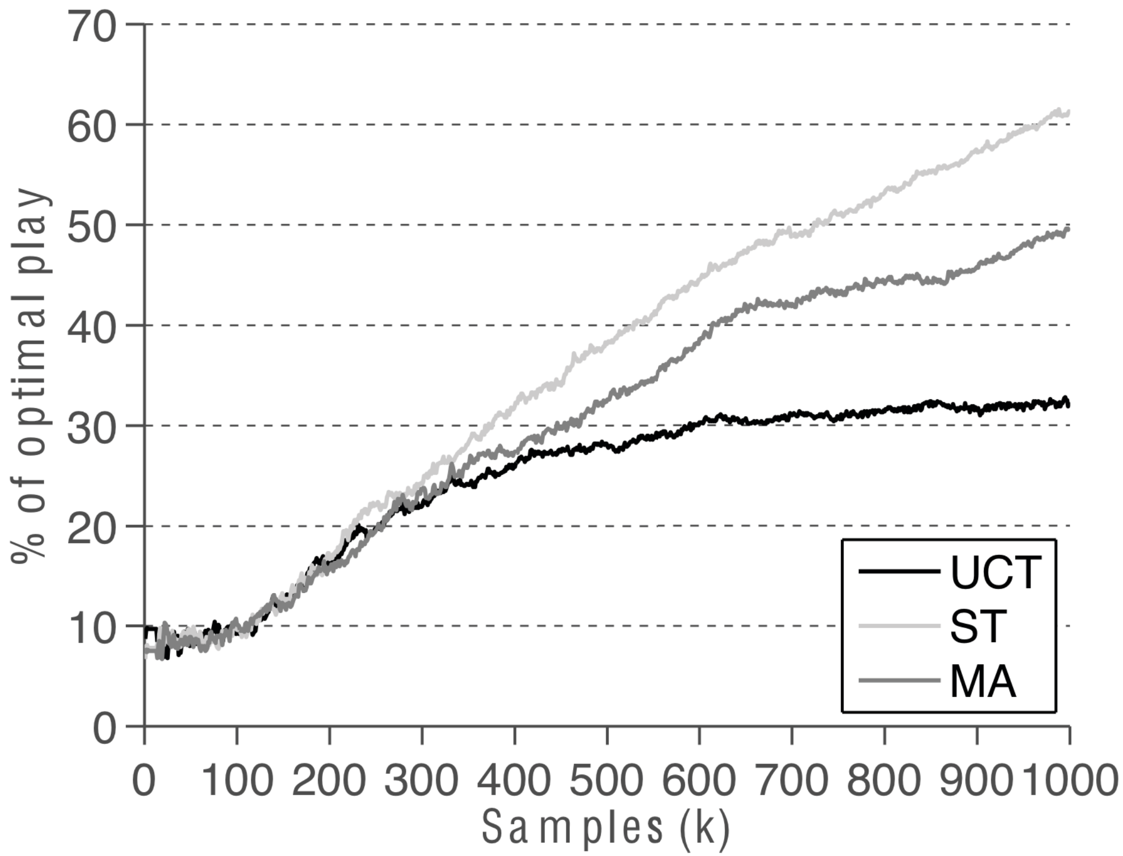
10%

30%

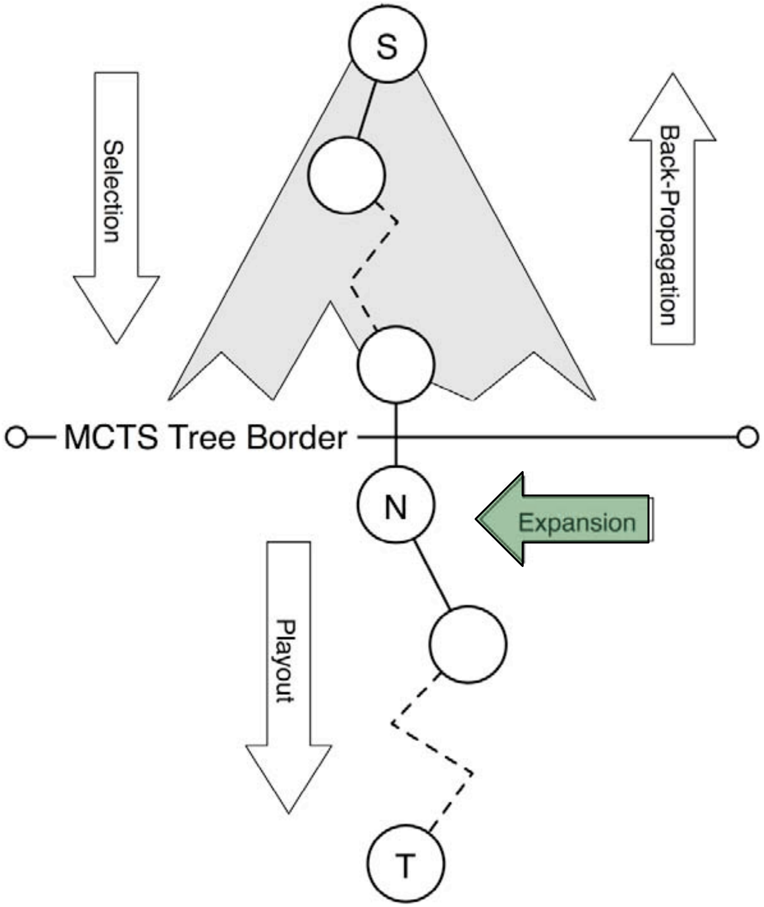


Breakthrough (problematic position)





Knowledge Bias



- FluxPlayer heuristics used to "evaluate" newly expanded nodes.
- Gives initial estimates for little explored nodes (a.la. Progressive Bias)

Search-Control Playouts

- Framework
  - Gibbs measure
- Schemes in the playout phase
  - MAST (2008)
  - TO-MAST (2009)
  - PAST (2009)
  - FAST (2010)
- Combined schemes
  - RAVE/MAST (2009)
  - RAVE/FAST (2010)

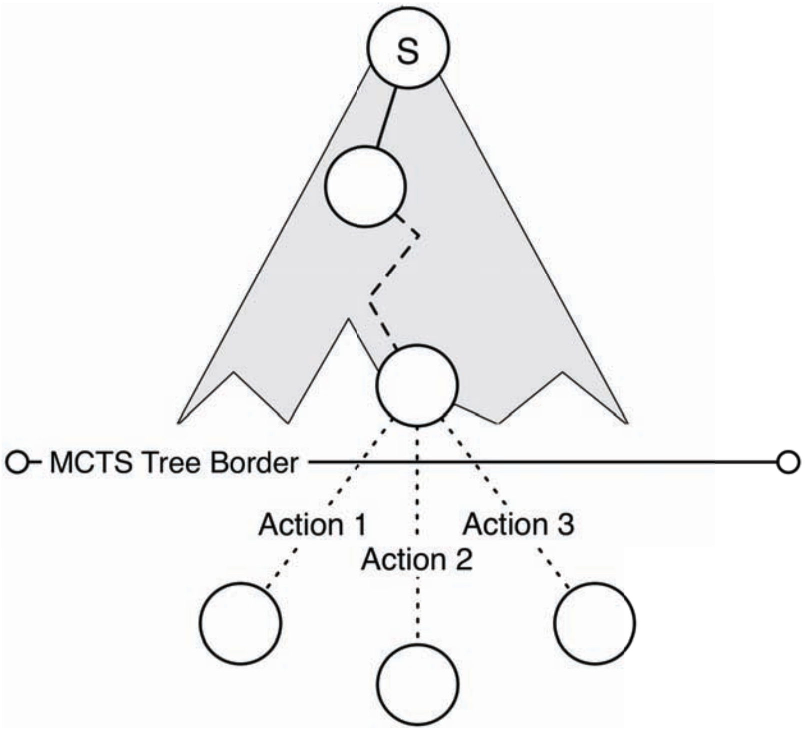
Search Control Framework

- Move selection is biased in the playout phase.
  - Assume that $Q_h(a)$ is a measure of a move's quality.
  - We then use Gibbs measure to choose a move with a probability:

$$\mathcal{P}(a) = \frac{e^{\mathcal{Q}_h(a)/\tau}}{\sum_{b=1}^{n} e^{\mathcal{Q}_h(b)/\tau}}$$

  - The tau parameter can be used to adjust how greedy the selection is towards the best moves
    - Stretch or flatten the distribution

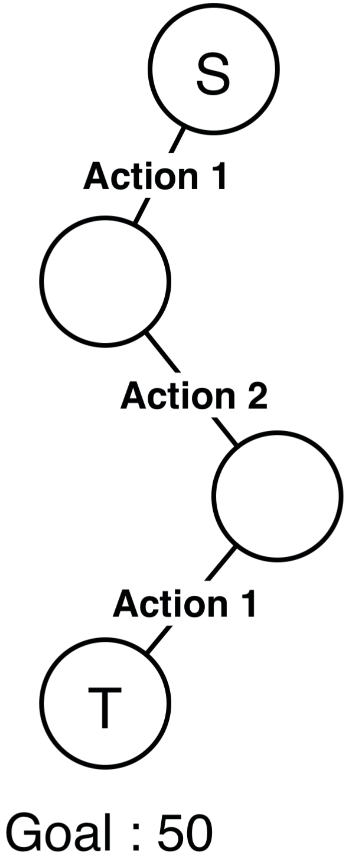
Search Control Scheme

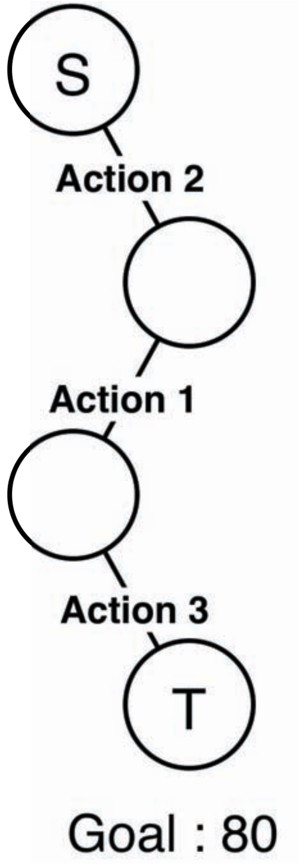


- Used
  - in the playout phase
  - Fringe of the MCTS tree to choose between unexplored moves.

$$\mathcal{P}(a) = \frac{e^{\mathcal{Q}_h(a)/\tau}}{\sum_{b=1}^{n} e^{\mathcal{Q}_h(b)/\tau}}$$

Move Average Sampling Technique (MAST)



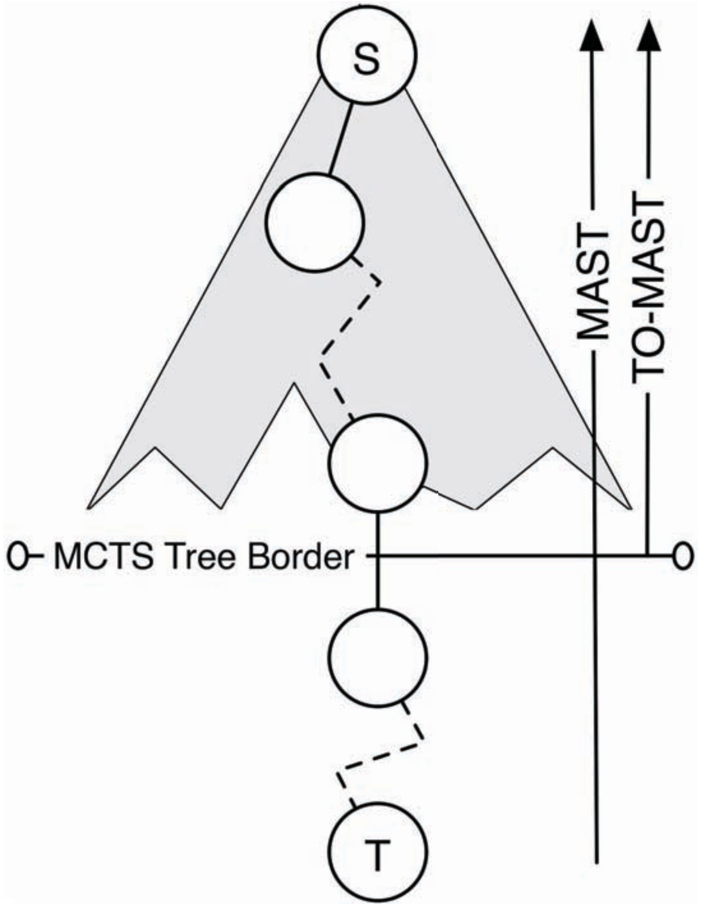


Action	Samples	Q(a)
Action 1	3	60
Action 2	2	65
Action 3	1	80

With $\tau = 10$:

P(Action 1) = 9.9%
P(Action 2) = 16.4%
P(Action 3) = 73.6%

Tree Only MAST (TO-MAST)



- Same as MAST, but
  - Only update Q when in the MCTS tree
- Samples
  - Fewer samples
  - More relevant, better quality (?)
- Generalization
  - Generalization more local (to early part of playoffs)

PREDICATE AVERAGE SAMPLING TECHN. (PAST)

- Look at actions in correlation with state predicates
  - Finer granularity of generalization
  - Possible to detect if an action is good in a given context
    - E.g. place a piece on a3 is good only if opponents piece on a2
- Keep statistic:

Action /Pred	Predicate 1	Predicate 2	Predicate 3
Action 1	Q(p,a) = 60	Q(p,a) = 50	Q(p,a) = 65
Action 2	Q(p,a) = 80	Q(p,a) = 65	Q(p,a) = 50
Action 3	Q(p,a) = 80	Q(p,a) = 0	Q(p,a) = 80

PAST

- Action selection:
  - Map into our Q(a) based framework.

- For every action *a* available in state
  - Q(a) is the maximum Q(p, a) over predicates *p* in the current state
  - High variance valued ignored (too few samples).

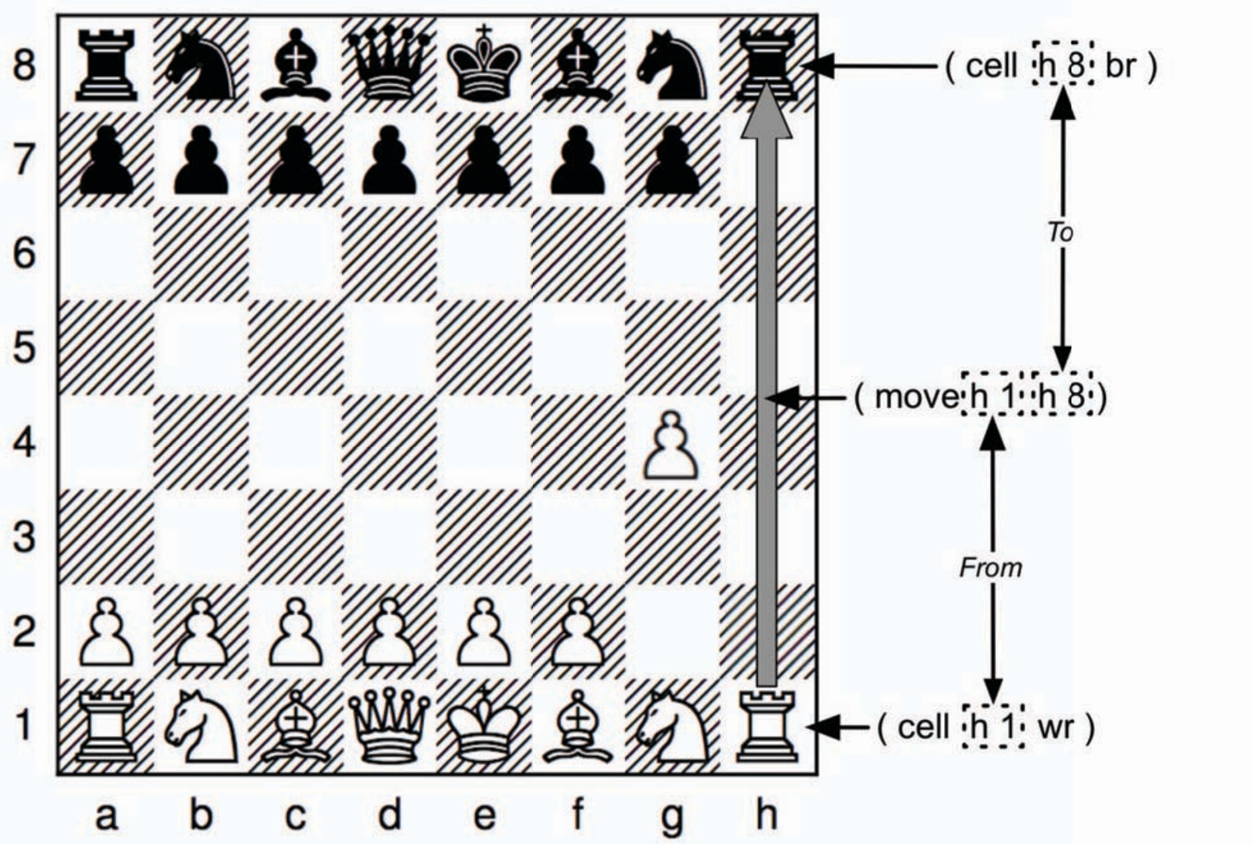
- Notes:
  - Using maximum value works for us significantly better than averaging values.

FEATURES TO ACTION SAMPLING TECHN. (FAST)

- Use template matching to identify common board game features. We currently detect:
  - Pieces of different types (piece based)
  - Board locations (location based)

- Use reinforcement learning, TD(λ), to learn the relative importance of detected features
  - Learns after each simulation episode during game play.
  - Learning kicks in once stable “enough”

FAST

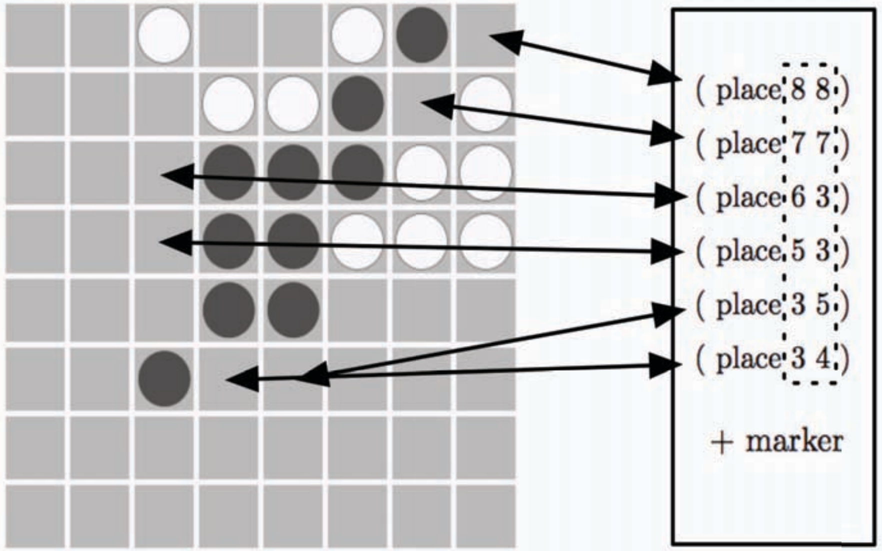


Feature	Value
...	
br	-4.98
...	
wr	5.26
...	

The values learnt for the piece type features found at the from and to positions are then fed to the formula for this move:

$$
\begin{aligned}
Q_h(\text{move h 1 h 8}) &= -(2 * br + wr) \\
&= -(2 * -4.98 + 5.26) \\
&= 4.7
\end{aligned}
$$

FAST



Feature	Value
...	
3 4 white	0.7334
7 7 white	0.8382
...	
5 3 white	1.6626
6 3 white	1.6909
...	
3 5 white	1.2550
...	
8 8 white	4.1129
...	

The values learnt for the position with our marker are then fed to the formula for these moves :

$$Q_h(\text{place } 3\ 4) = c * [3\ 4\ \text{white}]) = 0.7334c$$

$$Q_h(\text{place } 3\ 5) = c * [3\ 5\ \text{white}]) = 1.2550c$$

$$Q_h(\text{place } 5\ 3) = c * [5\ 3\ \text{white}]) = 1.6626c$$

$$Q_h(\text{place } 6\ 3) = c * [6\ 3\ \text{white}]) = 1.6909c$$

$$Q_h(\text{place } 7\ 7) = c * [7\ 7\ \text{white}]) = 0.8382c$$

$$Q_h(\text{place } 8\ 8) = c * [8\ 8\ \text{white}]) = 4.1129c$$

EMPIRICAL EVALUATION

- Hardware
  - Linux based 8-processor Intel® Xeon® 2.66 GHz CPU
  - 4GB of RAM
  - Each program uses single thread
- Both *start* and *play* clocks are 10 seconds
- Four different games used as a test-bed
  - All turn-taking 2-player zero-sum games
- Each data-point based on
  - 300 games (except last table, is 200 games)

INDIVIDUAL SCHEMES

Table: Tournament against the MCTS agent.

Game	MAST win %	TO-MAST win %	PAST win %
Breakth.	90.00 (± 3.40)	85.33 (± 4.01)	85.00 (± 4.05)
Checkers	56.00 (± 5.37)	82.17 (± 4.15)	57.50 (± 5.36)
Othello	60.83 (± 5.46)	50.17 (± 5.56)	67.50 (± 5.24)
Skirmish	41.33 (± 5.18)	48.00 (± 5.29)	42.33 (± 5.16)

Game	RAVE win %	FAST win %
Breakthr.	63.33 (± 5.46)	81.67 (± 4.39)
Checkers	82.00 (± 4.08)	50.33 (± 5.36)
Othello	70.17 (± 5.11)	70.83 (± 5.10)
Skirmish	46.33 (± 5.30)	96.33 (± 1.86)

All schemes offer genuine improvements.

INDIVIDUAL SCHEMES

- MAST used in the 2008 winning agent. Baseline for the later improvements:

Game	TO-MAST win %	PAST win %	RAVE win %	FAST win %
Breakthr.	52.33 (± 5.66)	45.67 (± 5.65)	20.33 (± 4.56)	39.67 (± 5.55)
Checkers	82.00 (± 4.18)	55.83 (± 5.35)	78.17 (± 4.36)	46.17 (± 5.33)
Othello	40.67 (± 5.47)	49.17 (± 5.60)	58.17 (± 5.49)	56.83 (± 5.55)
Skirmish	56.00 (± 5.31)	43.33 (± 5.26)	59.83 (± 5.15)	97.00 (± 1.70)

- Notes:
  - PAST not very effective (fewer number of simulations)
  - TO-MAST particularly effective in Checkers (harmful for Othello)
  - RAVE effective in many games, but harmful in others.
  - FAST particularly effective for Skirmish (chess-like game)

COMBINED SCHEMES

- RAVE used in the selection phase, whereas MAST/FAST used in the playout phase.

Table: Tournament with MAST against RAVE/MAST and RAVE/FAST.

Game	**RM** win %	**RF** win %
Breakthrough	50.50 (± 6.95)	38.50 (± 6.76)
Checkers	83.50 (± 4.87)	74.00 (± 5.81)
Othello	73.75 (± 6.01)	66.00 (± 6.43)
Skirmish	53.00 (± 6.47)	97.00 (± 2.04)

- RAVE/MAST does significantly better in both Checkers and Othello.
- RAVE/FAST does significantly better in Othello.

GAME PROPERTIES AND MCTS PERFORMANCE

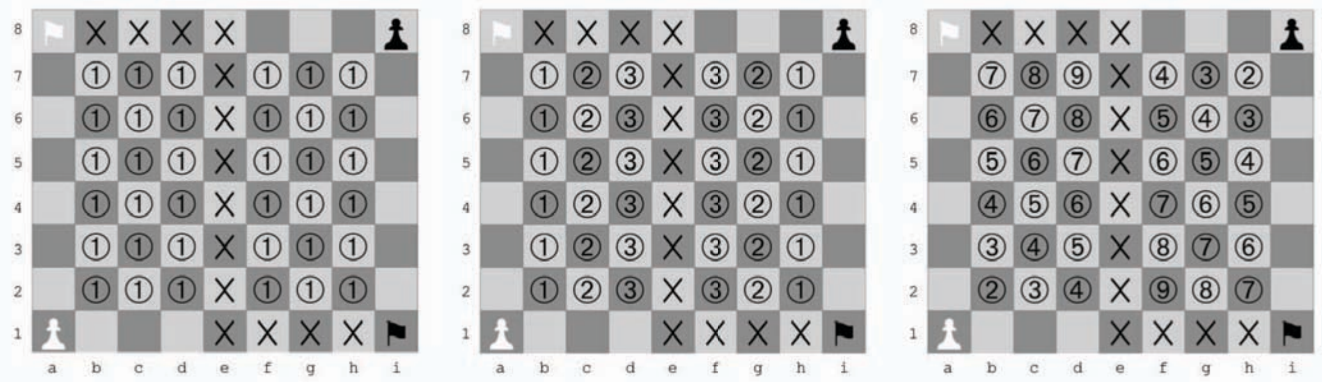


Figure 1: (a)Penalties Game (b)Shock Step Game (c)Punishment Game

- Tree Depth vs. Width
  - Difficult to generalize
- Progression
  - Surprisingly low ratio of “good simulations” required
  - Difficulty in games where had to “commit to a strategy”
- Optimistic Moves
  - Big problem

SUMMARY AND FUTURE WORK

- Summary
  - Learning of search-control in the playout phase is very important for MCTS based GGP agents
    - Because no a priori knowledge can be incorporated.
  - Difficult to come up with schemes that are robust across a wide range of games
  - Combining schemes is helpful
- Future work
  - Online detection of scheme's applicability as well as of more game-specific properties
  - Understanding better how different game properties affect MCTS
  - Combined MCTS / alphabeta approaches are needed in GGP