Faculty of Mathematics and Information Science
Warsaw University of Technology

Multigame Playing - A Challenge for Computational Intelligence

Jacek Mańdziuk

Grand challenges - MULTIGAME PLAYING

- Deep Blue II is unable to play simple tic-tac-toe
- Meta-level approach (autonomous game analysis, learning and problem solving)
- Focus on universal, generic AI/CI algorithms rather than solutions and metods optimized for single (particular) game:
  - Abstract reasoning
  - Knowledge representation
  - Game-independent learning
  - Planning
  - Knowledge transfer
  - Life-long learning
  - ...
- **Return to the idea of Strong AI (AGI)**

Multigame playing - SAL

- Search And Learning (M. Gherrity, 1993)

- Deterministic, two-player, perfect-information board games
- Game independent kernel with the general knowledge about this game genre – pre-defined, unchanged
- Game related knowledge (move-generator, constants defining the board, the pieces, etc.) provided by the user in a separate game-specific module (written in C)
- Two evaluation functions - 1hl MLP (➔ non-zero-sum games)
- Each represented as a NN with TD-learning
- Positional features (pieces, quantities), Dynamic features (moves, captures) and rule-based features (threathened pieces, controlled squares, etc)
- Shallow search (2-ply only)

- EXTREMELY SLOW LEARNING
- EXPLOSIVE NUMBER OF FEATURES

Multigame playing - HOYLE

- Hoyle (Susan Epstein, 1991-2004)
- Two-player, perfect-information, deterministic, finite board games
- Only the rules of the game are provided
- Three tiers of advisors commenting on particular aspects of a game position
- Tier 1: binary assessment of particular moves from the point of view of a given Advisor, e.g. Victory Advisor – winning moves, Wiser – winning in 1 ply, Sadder Advisor – losing, Dont' Lose – not losing in 1 ply, etc.)
- Tier 2: advocating certain plans of play (achieving particular goals)
- Tier 3: *non-binary* assessment of moves (in Advisor's expertise area) – Fork, Freedom (Mobility), Material, etc.
- Tiers 1 and 2 – sequential decision. If not conclusive then weighted decision made in Tier 3

- REASONABLE LEVEL OF PLAY in 18 two-player games (incl. t-t-t).
- WEAK CHESS playing (novice level)

Multigame playing - METAGAMER

- METAGAMER (Barney Pell, 1992-1996)
- Any „symmetric chess-like" (SCL) game, i.e. chess, checkers, shogi
- Universal evaluation function – linear combination of simple pre-defined features (partial goals)
- Similarly to Hoyle, some number of Advisors was employed, which voted on potential usefulness of particular features (in the form of numerical estimation of the worth of particular features)
- Among 23 Advisors there were 4 related to *mobility*, 4 to *threats and capturing*, 4 to *goal functions* and 11 to *material values*
- Shallow search – 1 ply with 1 ply non-quiescence search

- Novice level in chess (GNU Chess level 1 with a Knight handicap) and intermediate level in checkers
- No positive results in ad-hoc defined SCL games

PART 2

General Game Playing

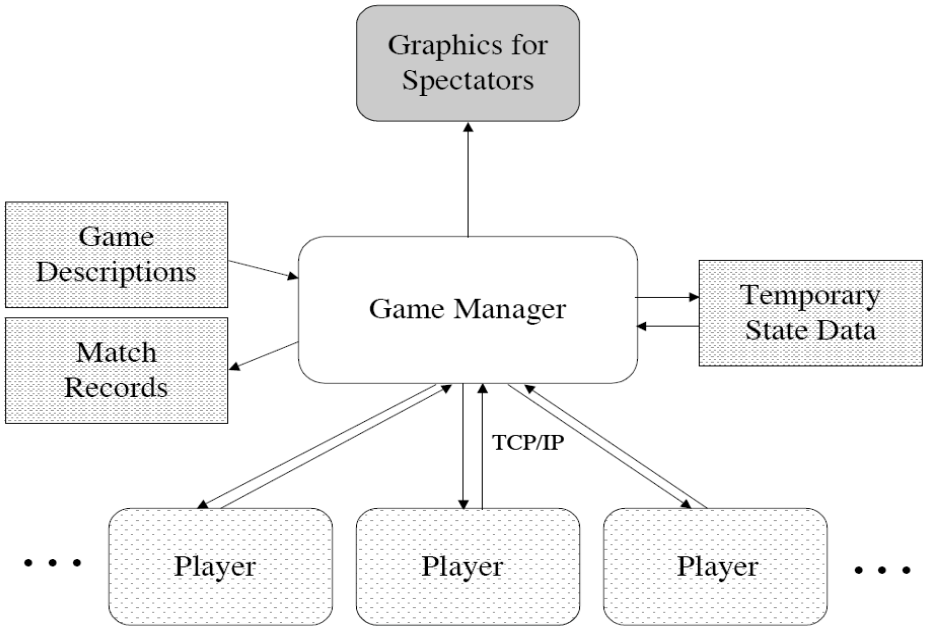
UCT search

General Game Playing

- Building programs (agents) able to play any game (within a certain class of games) without human intervention.
- Annual GGP tournament (since 2005)
- Stanford online course (2014)
- Game controlled by a GameMaster
- Communication via HTTP

General Game Playing

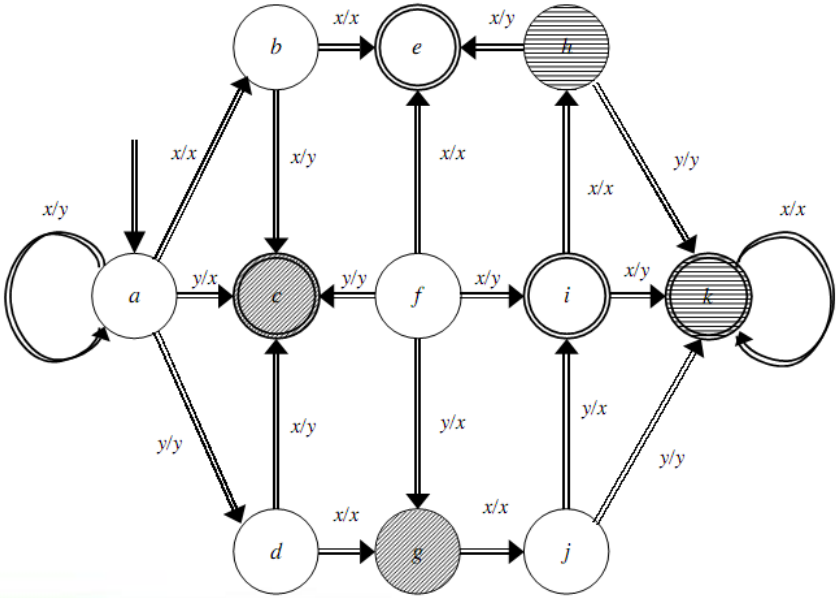
- GameMaster
  - Provides game description to each player
  - Verifies legality of proposed actions
  - Sends out info about (other) players' actions
  - Monitors time limits
- Time limits
  - StartClock
  - PlayClock



[http://games.stanford.edu/readings/aaai.pdf]

Genre of games

- Multiplayer
- Finite
- Synchronous
- Deterministic



M. Genesereth, N. Love, and B. Pell. General Game Playing: Overview of the AAAI Competition. AI Magazine, 26(2):62-72, 2005.

Game Definition Language (GDL)

- Allows more consise description of game rules
- Uses a version of Datalog – a subset of Prolog
- Specific notation of variables, relations, logic operators
- A set of keywords

Main GDL keywords/relations


```
(role xplayer)
(role oplayer)

(init (cell 1 1 b))
(init (cell 1 2 b))
...
(init (cell 3 3 b))
(init (control xplayer))

;; Cell
(<= (next (cell ?m ?n x))
   (does xplayer (mark ?m ?n))
   (true (cell ?m ?n b)))

(<= (next (cell ?m ?n o))
   (does oplayer (mark ?m ?n))
   (true (cell ?m ?n b)))

...

(<= (next (control xplayer))
   (true (control oplayer)))

(<= (next (control oplayer))
   (true (control xplayer)))

(<= (row ?m ?x)
   (true (cell ?m 1 ?x))
   (true (cell ?m 2 ?x))
   (true (cell ?m 3 ?x)))
```

```
(<= (column ?n ?x)
   (true (cell 1 ?n ?x))
   (true (cell 2 ?n ?x))
   (true (cell 3 ?n ?x)))

(<= (diagonal ?x)
   (true (cell 1 1 ?x))
   (true (cell 2 2 ?x))
   (true (cell 3 3 ?x)))

(<= (diagonal ?x)
   (true (cell 1 3 ?x))
   (true (cell 2 2 ?x))
   (true (cell 3 1 ?x)))


(<= (line ?x) (row ?m ?x))
(<= (line ?x) (column ?m ?x))
(<= (line ?x) (diagonal ?x))


(<= open
   (true (cell ?m ?n b)))

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

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

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

```
(<= (goal xplayer 100)
   (line x))

(<= (goal xplayer 50)
   (not (line x))
   (not (line o))
   (not open))

(<= (goal xplayer 0)
   (line o))

(<= (goal oplayer 100)
   (line o))

(<= (goal oplayer 50)
   (not (line x))
   (not (line o))
   (not open))

(<= (goal oplayer 0)
   (line x))


(<= terminal
   (line x))

(<= terminal
   (line o))

(<= terminal
   (not open))
```

role(r)
init(p)
true(p)
legal (r,a)
does(r,a)
next(p)
terminal
goal(r,value)

Closed world assumption

GGP simulation scheme

- Calculate legal actions **[legal]**
- For each role $i, i = 1, \ldots, N$ **[does]**
  - move[i] **(select an action)**
  - perform(move[i]) **(send out all moves)**
- Calculate the next game step **[next]**
- Check if terminal **[terminal]**
  - YES → goal(role,value) and STOP **[goal]**
  - No → goto [legal]

Games Played

GGP Player

Chess

IBM Deep Blue II

Chess
Checkers
Chinese Checkers (3,4,6)
Connect Four
Connect Five
Connect Four Suicide
Othello
Quarto
Pentago
Pilgrimige
Blocker
Tic-Tac-Toe
9-Board Tic-Tac-Toe
Numeric Tic-Tac-Toe
Tic-Tac-Chess (3,4)
Counterstrike (Simplified)
Eight Puzzle (solving)

Sudoku Puzzle (solving)
Breakthrough
Knight-through
Free For All (2,3,4)
Rock-Paper-Scissors
Lights Out
Cephalopod
Chess and Othello (simultaneously)
Farmers (3)
Zhadu
Nim
Cylinder Checkers
Nine Men's Morris
Finding a Knight's Tour
Flipping Pancakes
…

GGP approaches - summary

- 2005 – 2006
  - heuristic-based syntactical game analysis
  - construction of the evaluation function

- 2007-2016 → **MCTS**
  - simulation-based methods
  - Cadia Player (2007, 2008, 2012) – Reykiavik University, Iceland
  - Ary (2009, 2010) – Paris 8, France
  - TurboTurtle (2011, 2013) – USA
  - Sancho (2014) – USA & UK
  - Galvanise (2015)
  - WoodStock (2016) - France

MCTS - MOTIVATION

- MCTS combines the focused and precisely-defined tree search with the power of massive random sampling of the problem space
- MCTS can be applied to any problem which relies on sequential decisions implemented as a tree-search process (whose solutions can be represented as paths in a tree)
- MCTS is a well suited method for problems intractable by min-max / alpha -beta search (deep tree, large branching factor, lack of compact and reliable assessment function)
- DAG

MONTE CARLO METHODS - BASICS

Rooted in statistical physisc – numerical approximation of complex integrals.

Used for estimating the quality of actions in certain domains (games, planning) based on **uniform sampling** (now called ***flat Monte Carlo***)

Flat MC in many domains / problems is ineffective as it does not assume modeling the opponent in any way

Significant improvement in 2006 ➔ UCB1 action selection policy ➔ UCT search tree method

UCB1 (K-ARMED BANDIT)

- UCB = Upper Confidence Bounds (UCT = UCB applied to Trees)
- k-Armed Bandit Problem or k (one-arm) Bandits Problem
- Distributions of pay-offs $\{X_{j,t}\}_{t=1,2,\ldots}$ **j=1,2,…,k** are fixed, but unkno wn
- How to maximize the total (long-term) reward?

- Exploration vs. exploitation balance

- First, play each arm once
- Then, use the following rule:

$$A^* = \arg\max_{i=1,\ldots,k} \left\{ \overline{X}_i + C\sqrt{\frac{\ln n}{n_i}} \right\}, \quad C = \sqrt{2}$$

UCT - MULTIPLE SIMULATIONS

Perform multiple simulations according to the following pattern:

Choose an action not yet selected (if exists)

If all had already been tried select action a^*

$$a^* = argmax_{a \in A(s)} \left\{ Q(s,a) + C\sqrt{\frac{\ln N(s)}{N(s,a)}} \right\}$$

Q(s,a) – the average result for a pair (*state*, *action*)

N(s) – the number of visits in state *s*

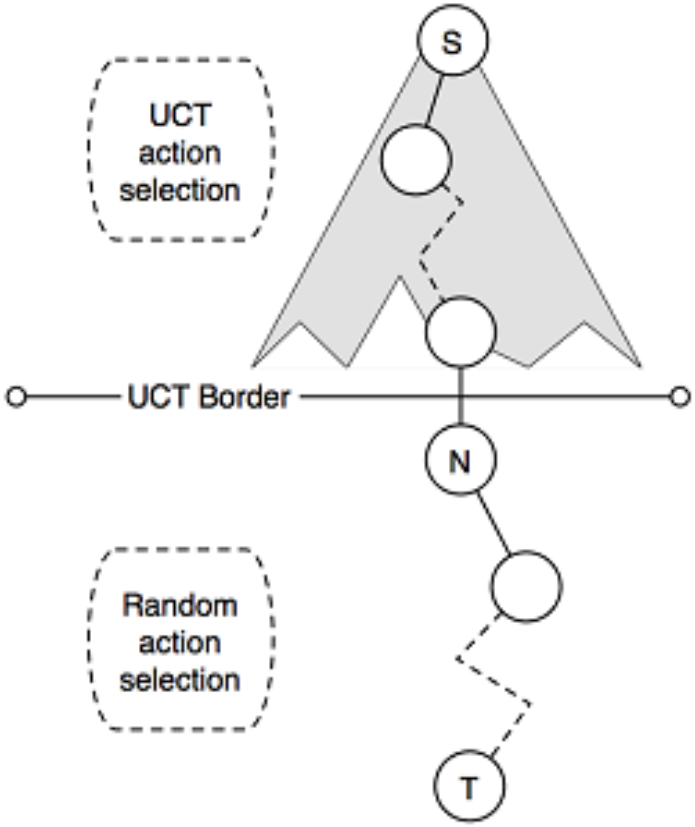
N(s,a) – the number of times action *a* was selected in state *s*

Once the simulation is completed propagate the result back in the tree

The saliency of C parameter

The UCT-based GGP player performs as many such simulations as possible (within allotted time), thus estimating the Q(s,a)

MCTS WITH UCT

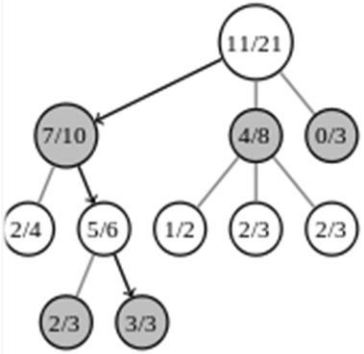


- Due to memory limitations:
  - A game tree is built gradually – one new node is added in each simulation.
  - Once the real (not simulated) action is selected by the system part of the tree above a given node is deleted from memory
- Random simulation to the leaf node.
- Node's assessment is equal to the average payoff of the simulations made so-far.

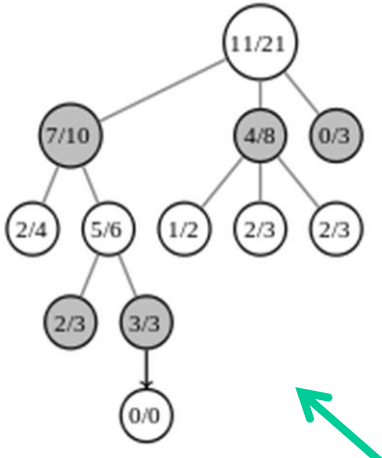
Bjornsson, Finnsson, IEEE TCIAI G, 1(1), 2009

MCTS/UCT

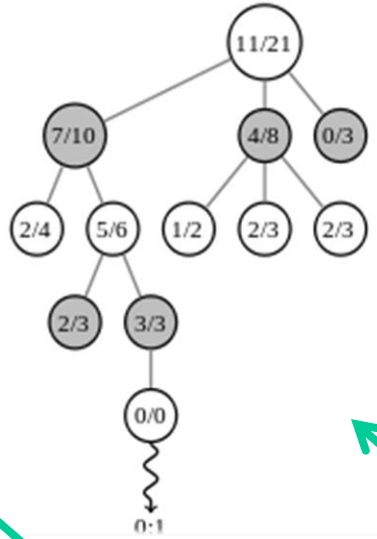
selection



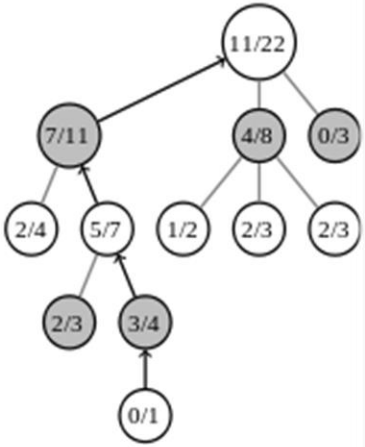
expansion

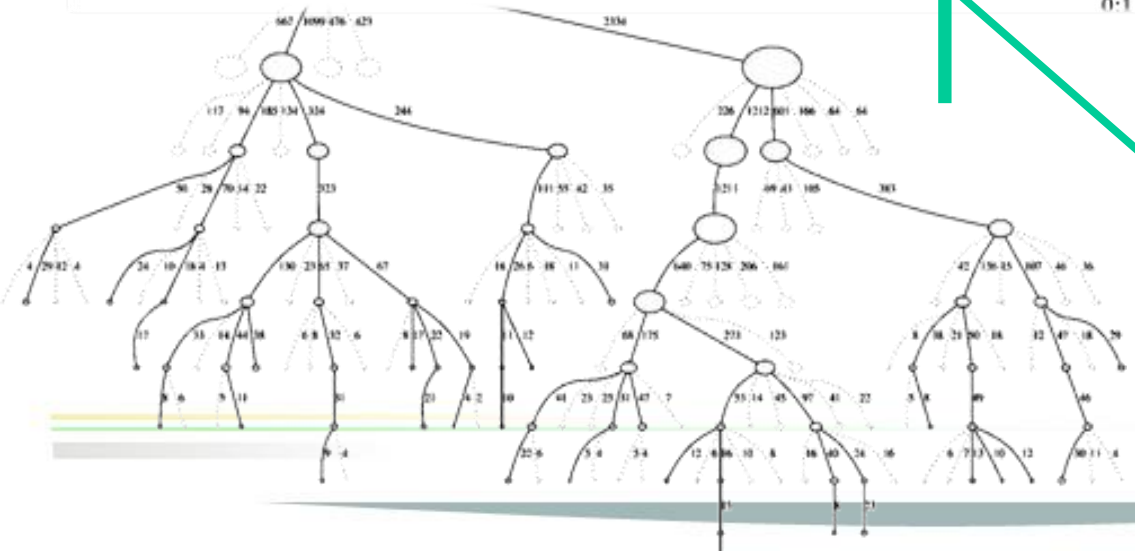


simulation



backpropagation





Tree policy (UCB1)

Default policy (random sampling)

MCTS/UCT

- In practice there are some constraints:
  - Computational budget
  - Time constraints
  - Memory limitations

- The system performs as many simulations as possible estimating the Q(s,a), in particular **Q(root, a)**

- Afterwards the action to be taken (in real NOT simulated mode) is selected

MCTS - ACTION SELECTION CRITERIA

$$\arg\max_a Q(root,a)$$ – ***Max child policy***

$$\arg\max_a N(root,a)$$ – ***Robust child policy***

Continue sampling until both measures point the same node - ***Max Robust child policy***

A combination of *Max* and *Robust* – ***Secure child policy***:

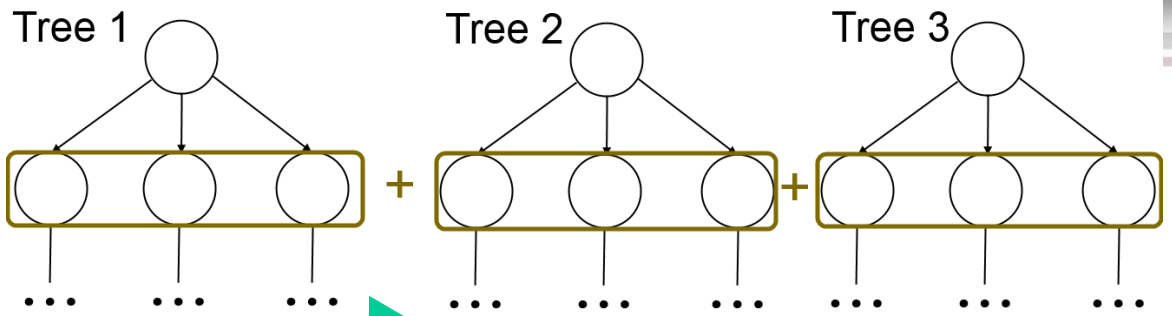
$$\arg\max_a \left[Q(root,a) + \frac{k}{\sqrt{N(root,a)}} \right]$$

MCTS - HIGHLIGHTS

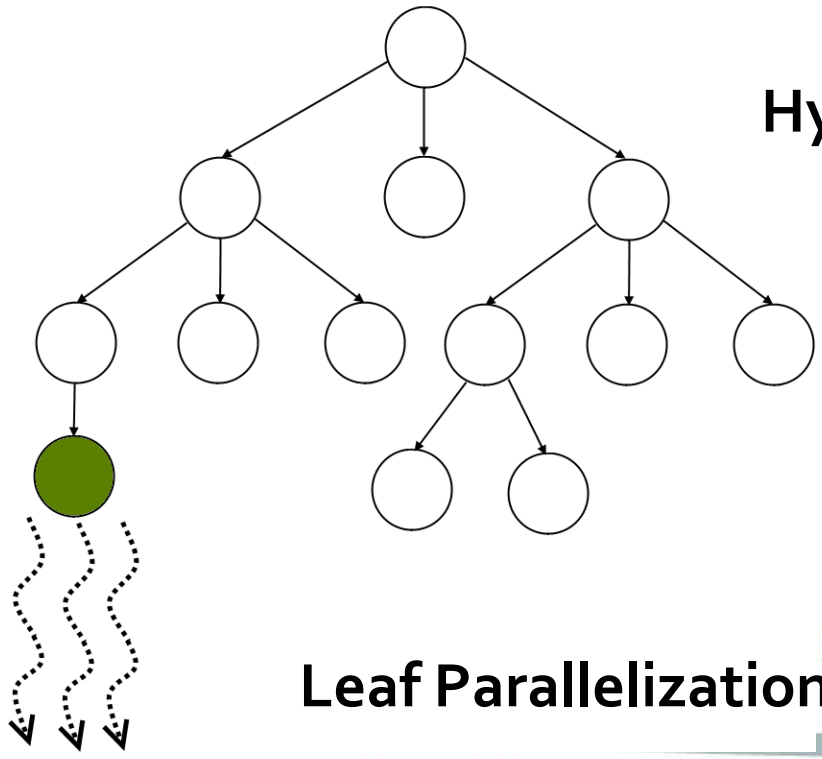
- Estimates true min-max value
- Anytime
- Aheuristic (Knowledge-free)
- Asymmetric (Best-First-like search)
- Well scaling and easily parallelized

MCTS – EASY PARALLELIZATION

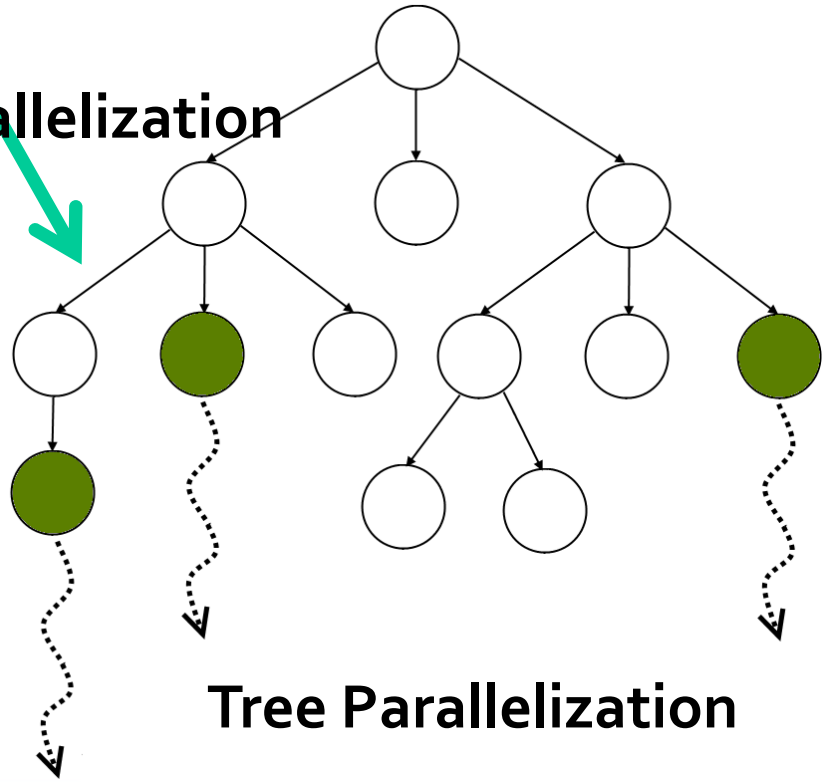
Root Parallelization



Hybrid Parallelization



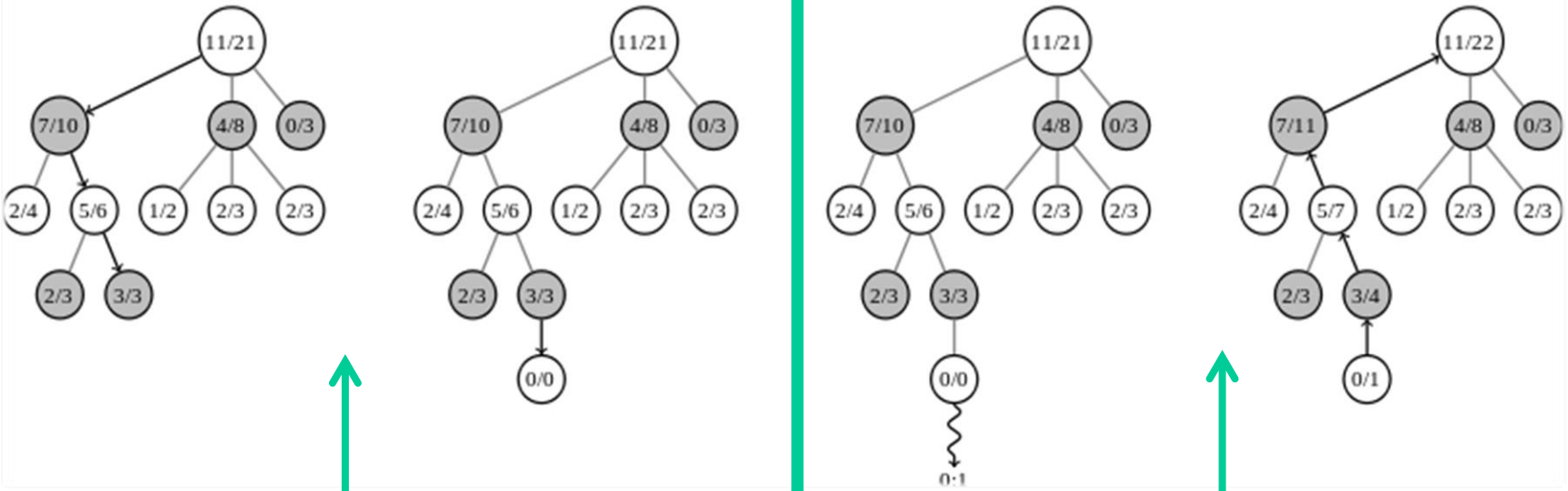
Leaf Parallelization



Tree Parallelization

MCTS/UCT

selection expansion simulation backpropagation



Tree policy (UCB1)

Default policy (random sampling)

TREE POLICY ENHANCEMENTS

- Three types of approaches:
  - Replacing UCB1 by another exploration-exploitation formula
  - Tuning UCB1 (variations)
  - UCB1 supported with an evaluation function

UCB1-TUNED

- Distributions of pay-offs $\{X_{j,t}\}_{t=1,2,\ldots}$ **j=1,2,…,k** are fixed, but unknown
- First, play each arm once
- Then, use the following rule: $A^* = \arg\max_{i=1,\ldots,k}\left\{\overline{X}_i + C\sqrt{\frac{\ln n}{n_i}}\right\},\ C = \sqrt{2}$

$$C = \sqrt{\min\left\{\frac{1}{4}, V_i(n_i)\right\}}$$

where $$V_i(r) = \left(\frac{1}{r}\sum_{m=1}^{r} X_{i,m}^2\right) - \overline{X}_{i,r}^2 + \sqrt{\frac{2\ln n}{r}}$$

UCB1 AND EVALUATION FUNCTION

- Suppose we have a „light"evaluation function:
  - Initial sorting of not-yet-selected actions
  - Initial estimation of actions' strengths based on some number of simulations

EXPANSION ENHANCEMENTS

- Rarely considered in practice
- One may try to optimize the number of new nodes added
  - Hard to estimate
  - Problem dependent
- One may optimize the number of concurrent simulations
  - The information is not updated (accurate)

BACKPROPAGATION ENHANCEMENTS

- Weighting simulation results
  - Some simulations are more important than the others
  - Newer simulations preferred over the later ones (esp. in the case of changing environment)
  - Shorter simulations are more accurate than the longer ones
  - Each simulation is assigned a positive integer weight ***w*** and is treated as performed ***w*** times

- Decaying Reward
  - Preference for shorter wins over the longer ones
  - A reward value is decayed by a value of $\gamma, 0 < \gamma \leq 1$ between each two nodes on the backpropagation path

DEFAULT POLICY ENHANCEMENTS

The biggest room for potential improvement

Motivation:

- The true min-max estimation, but …
- In the case of insufficient number of simulations: there is too much randomness in actions' assessment
- The sequences of actions are often „not realistic"

HISTORY HEURISTICS - GIBBS DISTRIBUTION

- An action which was successful in the past (regardless of the state) is treated as a potential candidate also in a given state.
- After each simulation statistics for each action are updated (the average performance – expected reward)

- Gibbs distribution:

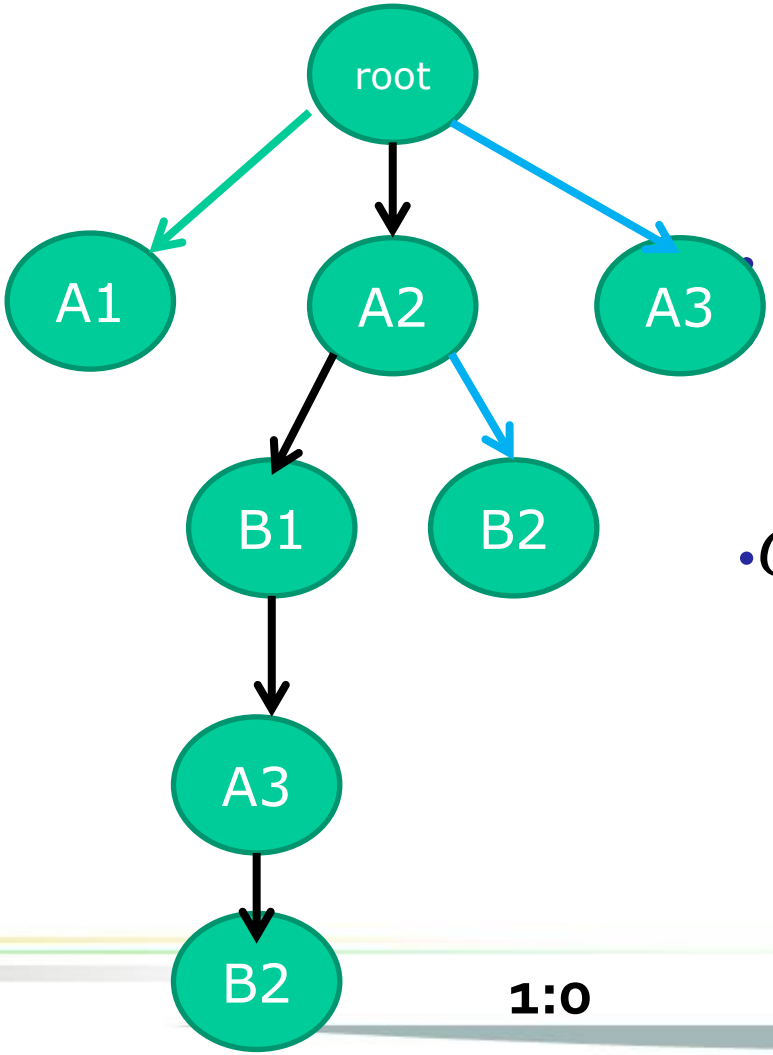
$$\pi_s(a) = \frac{e^{\frac{Q(a)}{\tau}}}{\sum_{b \in A(s)} e^{\frac{Q(b)}{\tau}}} \qquad \tau > 0$$

Roulette wheel or **Epsilon-greedy**

LAST GOOD REPLY POLICY

- Used specifically in games
- Each move is considered a reply to the opponent's last move and treated as successful if the player won that game
- For each move the last successful reply is stored (frequently overridden)
- In the simulation, if the LGR move is legal it will be played, otherwise the default policy will be used

RAVE AND ITS VARIANTS



- Used specifically in games
- Speeds up the learning process – especially at the beginning

- $Q_{RAVE}(s,a)$ is updated whenever action *a* is taken inside the simulation (not only as the starting action)

1:0

RAVE AND ITS VARIANTS

$$Q(s,a) := \beta(s,a) Q_{RAVE}(s,a) + (1 - \beta(s,a)) Q(s,a)$$

$$\beta(s,a) = \sqrt{\frac{k}{3N(s) + k}}$$

It is often assumed that $k = N(s) \Rightarrow \beta(s,a) = \frac{1}{2}$

DEFAULT POLICY ENHANCEMENTS

- Except for the HH (and its variants) it is hard to show a generally improving method unless some domain knowledge is considered
- The use of the „ad-hoc" constructed (**simple and automatically devised**) evaluation function
  - Replacing the whole the playout phase (with some predefined probability)
  - Finishing the playout phase before reaching the leaf node (with some predefined probability in each node → MAGICIAN

CadiaPlayer, Ary, TurboTurtle, Sancho 2007-2016

- All of them use the simulation-based method (UCT), which allows searching and gradually building the game tree
- All enhance UCT with simple heuristics (e.g. the history heuristics)
- Some try to detect „the expected" elements of a game (equipment): e.g., board, pieces, etc.
- Due to annual competitions the state-of-the-art versions remain unrevealed

UCT (GGP) - modeling the opponents

- Modeling the opponents' moves (during simulations):
- Each opponent has its own model assigned [CadiaPlayer]
  - Usually, opponents are assumed to use UCT/MC algorithm
  - Or, each opponent makes random moves
  - Alternatively, each opponent applies the same method as the tested player

Problems with pure UCT/MC

- In the case of insufficient number of simulations:
  - A tendency to overoptimistic play
    [Bjornsson, Finnsson – CadiaPlayer]
  - Too much randomness in moves' assessment

- The information provided through the game rules is not „analytically" taken into account

PART 3

MiNI Player
Strategy switching mechanism

Partly informed search in the MC phase

- In the upper (known) part of the tree: UCT search
- **Outside the UCT tree: a particular playing strategy S is interleaved with MC simulations**
- S is fixed for a given simulation, but not for the whole game
- **Generalization of UCT/MC to UCT/(MC/S)**

Underlying assumptions

- Utilization of the game rules/information
- Ability to use various search strategies – adequate to particular choice of a game
- Adaptability (self-improvement) through dynamic assessment of a current game strategy

Repository of strategies

- History Heuristics (HH)
- Greedy maximization of gaol condition accomplishment (AGE)
- Mobility (M)
- Statistical couting of symbols' occurences in game description forms (SSC)
- Exploration (E)
- Random play (R)
- Other … (e.g. variants of M or E)

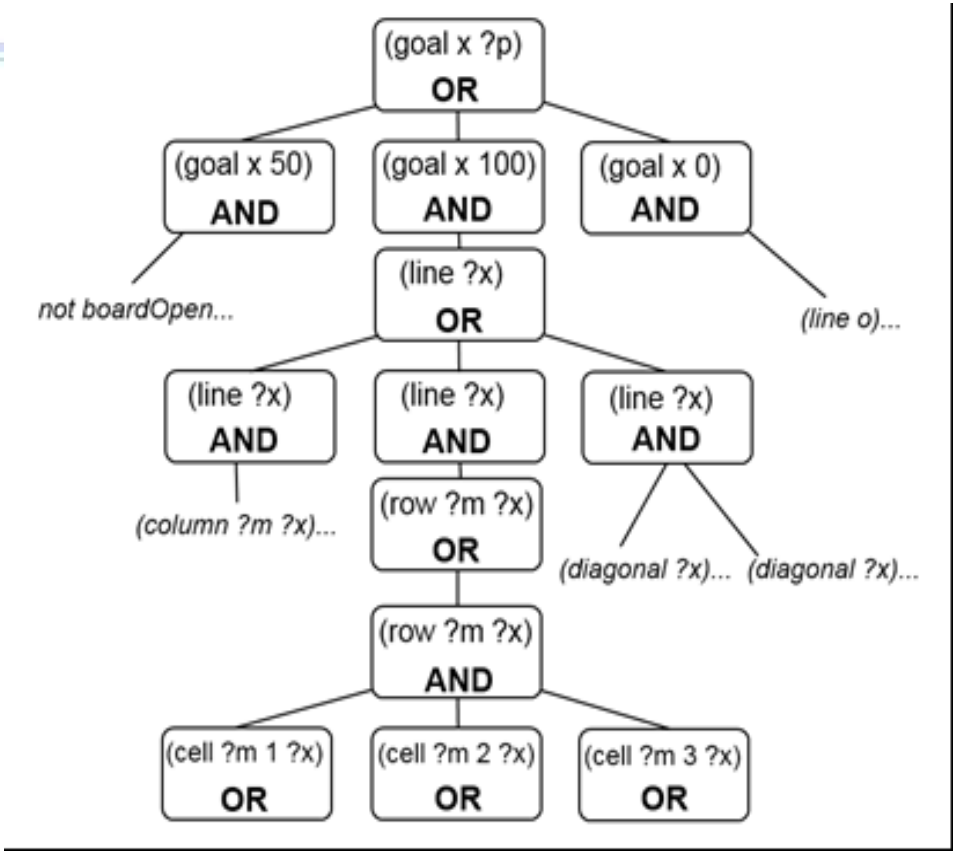
History Heuristics (HH)

- Frequently used in „classical" single-game approaches
- Assumes that and action is strong/weak regardless (in general) of the state it was played in

- Each strategy except for R(andom) keeps track of the moves played in a simulation and update the moves averages accordingly
- The average scores of actions are stored

- The (historically) best move is chosen with e =65% chance. Otherwise a random move is played.

Approximate Goal Evaluation (AGE)



- Approximate degree of a goal rule satisfaction
- OR nodes – set of all rules defining given relation (e.g. goal, line, row, diagonal, cell, etc.)
- AND nodes – particular instances

- AGE is defined recursively, bottom-up
- Two values are propagated $AgVal$ and $TbVal$

Approximate Goal Evaluation (AGE)

In OR leaf nodes:

$AgVal$ =1 if a rule is satisfied;
$AgVal$ = 0, otherwise.

$TbVal = AgVal$

In internal OR nodes

$$AgVal = \max_{i=1,\ldots,N} AgVal_i$$

$$TbVal = \frac{1}{N}\sum_{i=1}^{N} TbVal_i$$

In AND nodes:

$$AgVal = \frac{1}{N}\sum_{i=1}^{N} AgVal_i$$

$$TbVal = AgVal$$

- Move with the highest $AgVal$ is selected
- In the case of even results $TbVal$ is considered

Approximate Goal Evaluation (AGE) - Example of *TbVal* use

- goal(x,100) <= line(x)

- Groundings:
- 1: c11x, c12x, c13x
- 2: c21x, c22x, c23x
- 3: c31x, c32x, c33x
- 4: c11x, c21x, c31x
- 5: c12x, c22x, c32x
- 6: c13x, c23x, c33x
- 7: c11x, c22x, c33x
- 8: c31x, c22x, c13x

mark(2,1,x)	
Groundings:	
1: 0, 0, 0	0
2: 1, 0, 0	1/3
3: 0, 0, 0	0
4: 0, 1, 0	1/3
5: 0, 0, 0	0
6: 0, 0, 0	0
7: 0, 0, 0	0
8: 0, 0, 0	0

mark(2,2,x)	
Groundings:	
1: 0, 0, 0	0
2: 0, 1, 0	1/3
3: 0, 0, 0	0
4: 0, 0, 0	0
5: 0, 1, 0	1/3
6: 0, 0, 0	0
7: 0, 1, 0	1/3
8: 0, 1, 0	1/3

c11x ←→ (cell 1 1 x)

Mobility (M)

- The number of actions available (relatively to other players).
- In majority of the games the greater the mobility the better.
- The state which maximizes the following condition is chosen:

$$M_0 + \sum_{i=1}^{N-1} (M_0 - M_i)$$

Statistical Symbol Counting (SSC)

General idea:

- Identification of all state facts (*cell*, *control*, *line*, etc.) and calculation of their average quantity.

Initial state of checkers:

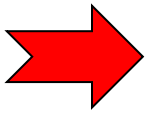
- Quantity(”cell”) = 64
- Quantity(”control”)=1
- Quantity(”piece_count”)=2

etc.

Also for the symbols that appear in a certain place of the argument list

Tic-Tac-Toe

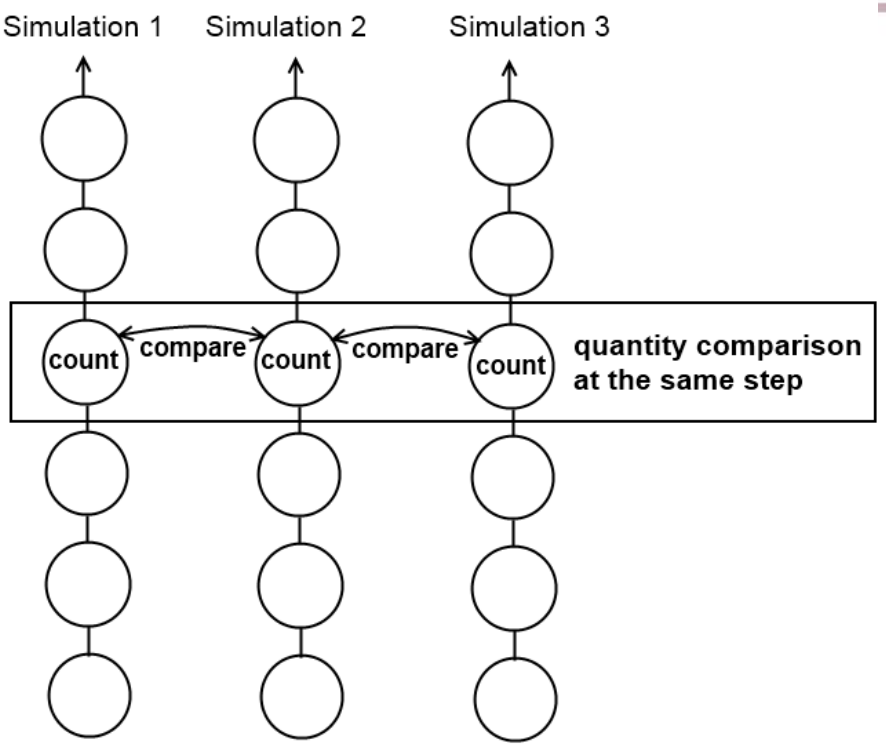
```
[cell 1 1 x]
[cell 1 2 b]
[cell 1 3 x]
[cell 2 1 b]
[cell 2 2 o]
[cell 2 3 b]
[cell 3 1 b]
[cell 3 2 b]
[cell 3 3 b]
```



```
Quantity(”cell”) = 9
Quantity(”cell”, 1, ”1”) = 3
Quantity(”cell”, 1, ”2”) = 3
Quantity(”cell”, 1, ”3”) = 3
Quantity(”cell”, 2, ”1”) = 3
Quantity(”cell”, 2, ”2”) = 3
Quantity(”cell”, 2, ”3”) = 3
Quantity(”cell”, 3, ”x”) = 2
Quantity(”cell”, 3, ”o”) = 1
Quantity(”cell”, 3, ”b”) = 6
```

Statistical Symbol Counting (SSC)

Facts with „stable" or „predictable" quantities are filtered out based on random simulations.



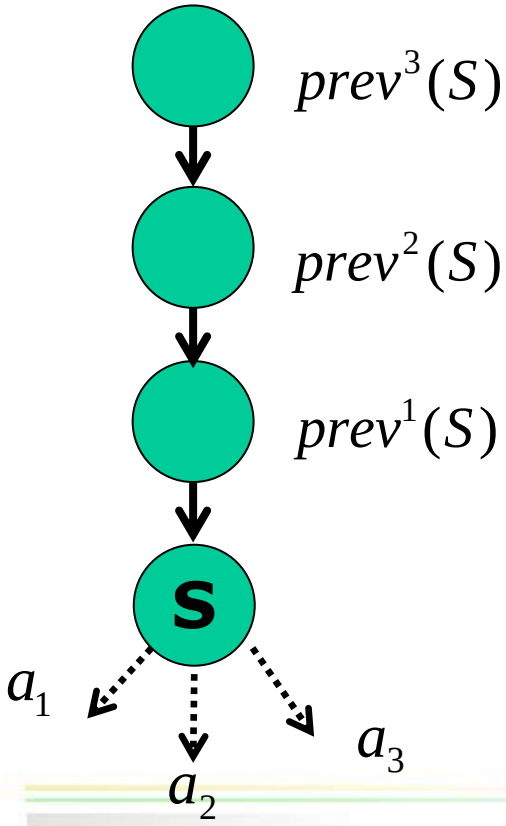
The averages are calculated separately for won/lost games, i.e. (above/below the average score)

$$Eval = \sum w_i q_i$$

$$w_i = \frac{AVG_i^W - AVG_i^L}{MaxValue_i}$$

Exploration (E)

- The idea: explore new states which are different from the previous states, though „linked" to them.
- Difference between states:



$$diff(B,A) = B - A$$

$$diff(B,A) = \left|\{f : f \in B \wedge f \notin A\}\right|$$

$\forall a_i$ in S, calculate $S_i = next(a_i)$

$\forall i$ find „the closest" previous state

$$\arg d_i$$

$$d_i = \min_{j=1,\ldots,N} diff\ (S_i, prev^j(S_i))$$

Choose action $a_i := \arg max_i(min_i d_i)$

$N = 3$

Random play (R)

- All actions in a given state have the same probability
- Effectively, the same as „classical" UCT (out-of-a-tree phase of a simulation)
- Highly increases exploration capabilities of the system
- Computationally efficient ➔ allows for many more simulations in a give time frame, compared to other strategies

Probability of strategy application

P	AGE Checkers	SSC Chess	M Othello	E Farming Quandries	HH Connect-4
0.20	59.17	50.00	54.33	54.83	54.67
0.40	66.00	51.00	55.50	56.50	55.83
0.50	65.67	51.33	68.17	58.17	59.33
0.60	70.67	51.00	73.83	59.50	59.00
0.65	79.17	50.83	73.67	60.67	57.33
0.70	76.33	51.33	81.33	61.17	**66.67**
0.75	**79.33**	49.50	83.67	61.67	62.92
0.80	75.83	53.17	81.17	**62.17**	59.17
0.85	72.67	54.17	**85.17**	58.33	54.83
0.90	69.33	54.64	79.17	42.50	50.67
1.00	55.67	**55.00**	68.17	34.83	44.17
95% Conf.	±4.46	±9.07	±3.90	±5.10	±5.13

Interleaving of strategies

- UCB-LIKE FORMULA

- In simulation n we assign strategy S^* based on the UCB formula:

$$S^* = \arg\max_{i=1,\ldots,6} \left\{ Q_i + C\sqrt{\frac{\ln(n)}{T(S_i, n-1)}} \right\}$$

- $\boldsymbol{Q_i}$ - the average return of strategy S_i
- $\boldsymbol{n}$ – the number of simulations already performed
- $\boldsymbol{T(S_i, n)}$ – number of times S_i has been selected
- $\boldsymbol{C}$ – exploration coefficient ($C = 5$ was used during the championships)

Results

- ***MiNI / MiNI-Player*** – the final version of the player with strategy switching mechanism
- ***MiNI-C*** – a version based on classical MC/UCT approach, enhanced with the HH only
- ***MiNI-1-S*** – a player which uses only one (best suited for a given game) strategy in MC simulation phase
- ***Cadia / CadiaPlayer*** – publicly available version AD 2011

MiNI-Player vs. Cadia – 2 player games

Game	Clock [s]		MINI vs. Cadia
Connect4	40	15	41.67 [5.35]
Cephalopod Micro	60	20	40.00 [5.84]
Free for all 2P	45	15	63.33 [5.14]
Pentago	45	15	29.33 [5.43]
9 Board Tic-Tac-Toe	45	15	70.67 [5.16]
Connect4 Suicide	45	15	53.33 [5.26]
Checkers	60	20	54.33 [5.88]
Farming Quandries	90	30	68.33 [4.21]
Piligrimage	90	30	42.67 [4.32]
Average			**51.40** [5.18]

270 games (with sides swapped)

Game (won, lost, drawn)

Score is normalized to [0,100]

MiNI-Player-C (i.e. UCT/(MC/HH))

Game	MINI vs. MINI-C	Cadia vs. MINI-C
Connect4	61.33 [5.69]	59.67 [5.54]
Cephalopod Micro	59.33 [5.86]	73.33 [5.27]
Free for all 2P	75.33 [4.86]	65.67 [5.11]
Pentago	40.00 [5.84]	55.33 [5.93]
9 Board Tic-Tac-Toe	66.30 [5.42]	50.00 [5.76]
Connect4 Suicide	51.33 [5.72]	43.33 [5.50]
Checkers	79.33 [4.83]	69.33 [5.32]
Farming Quandries	66.67 [5.36]	59.67 [4.90]
Piligrimage	39.33 [5.40]	62.83 [5.39]
Average	**59.88** [5.44]	**59.91** [5.42]

MiNI-1-S (with a single (best) strategy)

Game	MINI-1-S vs. MINI-Player
Connect4	52.41 [5.90]
Cephalopod Micro	45.92 [5.94]
Free for all 2P	48.33 [5.56]
Pentago	50.00 [5.96]
9 Board Tic-Tac-Toe	52.96 [5.75]
Connect4 Suicide	40.30 [5.60]
Checkers	57.78 [5.78]
Farming Quandries	58.89 [5.33]
Piligrimage	35.63 [5.33]
Average	**49.14** [5.68]

MiNI Player vs. Cadia – 1 player games

Game	Clock [s]		MINI-Player	CadiaPlayer
Hanoi	30	20	80.70 [0.55]	99.00 [0.52]
Knight Tour	30	20	100.00 [0.00]	100.00 [0.00]
8-Puzzle	30	20	64.80 [4.82]	3.60 [2.10]
Lights Out	30	20	18.50 [4.63]	0.00 [0.00]
Rubik's Cube	30	20	0.00 [0.00]	13.25 [1.17]
Average			**52.80** [2.00]	**43.17** [0.76]

GGP Competition 2012

- Two phases of the tournament

- PHASE 1: Seven games played against selected opponents
- The top 8 advanced to the final round (MP was the 5th, 4 wins and 3 losses)

- PHASE 2 – until two losses
- Finally MP placed 7th

- Connect-4
- Cephalopod Micro
- Free for All 2P
- Pentago
- 9 Board Tic-Tac-Toe
- Connect-4 Suicide
- Checkers
- Farming Quandries
- Pilgrimage

GGP Competition 2013

- Two phases of the tournament
- PHASE 1: many games including 1,2,4-player ones; 5 advanced to phase 2 (but not MP ☹, and not M ☹)
- MP – a bug in single-player games ☹
- Finally MP placed 11th (out of 16)

GGP Competition 2014

- Three phases of the tournament
- Massive Stanford online course

- PHASE 1: eliminations (various types of games)
- The top 17 advanced to the second PHASE
- PHASE 2 – top 12 advanced to the final phase

- PHASE3 – we ended up at the 7-8th place (out of 49 participants)

Conclusions

- **Multigame playing is besides intuitive playing and creativity one of the hallmarks of human intelligence (in game domain)**
- GGP directly addresses this issue and stimulates research towards accomplishing this goal
- The are some inefficiencies (especially slowness of analysis) which stem from logic description → how to obey/alleviate them?
- GDL (GGP) is potentially extendable to non-deterministic games
- **If we leave out time constraints more established CI learning approaches (neural nets, genetic/memetic methods) can come into play**
- **Applicability of UCT extends beyond game domain (problems with pay-off defined only in terminal states and/or varying in time)**
- Strategy switching mechanism allows for greater flexibility of the method, its self-improvement and adaptability

Thank you for your attention