GENERIC HEURISTIC APPROACH TO GENERAL GAME PLAYING

Maciej Świechowski (m.swiechowski@ibspan.waw.pl)

Jacek Mańdziuk (mandziuk@mini.pw.edu.pl)

SOFSEM 2012

AGENDA

- Introduction to General Game Playing
- Game Description Language (GDL)
- A brief look at top GGP players
- Our approach – base idea
- Elements discovery
- Evaluation function construction
- Playing results
- Conclusions and future work

GARRY KASPAROV – DEEP BLUE [1997]

GENERAL GAME PLAYING (GGP)

Idea:

To create computer players (artificial inteligence programs) which are able to play a variety of games effectively.

- without human intervention
- even games never encountered before

GGP Competition:

- Tournament for GGP players started by Stanford Logic Group at Stanford University:

  http://games.stanford.edu/

- Held annually since 2005 *(at AAAI National Conference 2005-2010, IJCAI – 2011)*
- Natural, interactive, competitive platform for testing solutions

GENERAL GAME PLAYING

Motivation:

- Escape from heavy specialization
- Programs are expected to perform game analysis and learn how to play strongly
- Programmers develop intelligent mechanisms instead of ultra-fast algorithms dedicated to solve a particular game
- Follows the fundamental mission of AI better

- Possibility to incorporate various aspects of AI working in integrated fashion such as:
  - Knowledge representation
  - Abstract reasoning and decision making
  - Learning
  - Planning and evaluation

    and more

GAME DESCRIPTION LANGUAGE (GDL)

No game specific knowledge should be encoded – players accept declarative description of games at **runtime**.

Game Description Language:

- Used to define complete rules of arbitrary games in GGP scenario
- Based on the first-order logic
- GDL derives from **Datalog**
- **Datalog** is a subset of **Prolog**

Logical rules as more compact representation than finite state machines (FSM)

- Power of games description is equivalent
- FSM representation is too complex e.g. Chess requires 10^{28} states

GAME DESCRIPTION LANGUAGE (GDL)

Available class of games (GDL-I):

- **Finite**
- **Deterministic**
- Synchronous

Available class of games (GDL-II):

- **Finite**
- Synchronous

How all this work:

- Logical rules enable to dynamically compute game states and aspects. Computed (realized) rule produces set of true facts.
- Like in Prolog, the whole world is defined by facts which are true.

- Predefined keywords are „entry points” to particular elements of the game flow
- There also exist constant facts which are always true

GDL

Keywords:

- **role(r)**
- **init(p)**
- **true(p)**
- **does(r, a)**
- **next(p)**
- **goal(r, v ∈ $[0..100]$)**
- **terminal**
- **distinct (v, w)**

```
(role xplayer) (role oplayer)
(init (cell 1 1 b))
(init (cell 1 2 b))
 ...
(init (cell 3 3 b))
(init (control xplayer))

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

(<= (next (cell ?m ?n b))
    (does ?w (mark ?j ?k))
    (true (cell ?m ?n b))
    (or (distinct ?m ?j) (distinct
?n ?k)))
...

(<= (legal white (mark ?x ?y))
    (true (cell ?x ?y b)))
(<= (legal black (mark ?x ?y))
    (true (cell ?x ?y b)))

...

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

```
 (<= (line ?x) (row ?m ?x))
 (<= (line ?x) (column ?m ?x))
 (<= (line ?x) (diagonal ?x))
 (<= open (true (cell ?m ?n b)))

...

(<= (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))
...

(<= terminal
    (line x))
(<= terminal
    (line o))
(<= terminal
    (not open))
```

A BRIEF LOOK AT OTHER GGP PLAYERS

Solutions based on a Min-max tree-search:

- ClunePlayer – ***simple game model (payoff, mobility, termination, p stability, t stability)***
- FluxPlayer – ***syntatic structures, fuzzy logic***
- ...

Solutions based on MonteCarlo simulations with UCT rule:

- CadiaPlayer
- Ary
- …

UCT rule – Upper Confidence Bounds Appied for Trees – balance between exploration and exploitation ratio

Other:

- NEAT – Neuroevolution of Augmenting Topologies

UCT RULE

Variant of UCB

When still within tree:

- **Select each action once and from now on:**
- **Choose action-state pair that maximizes:**

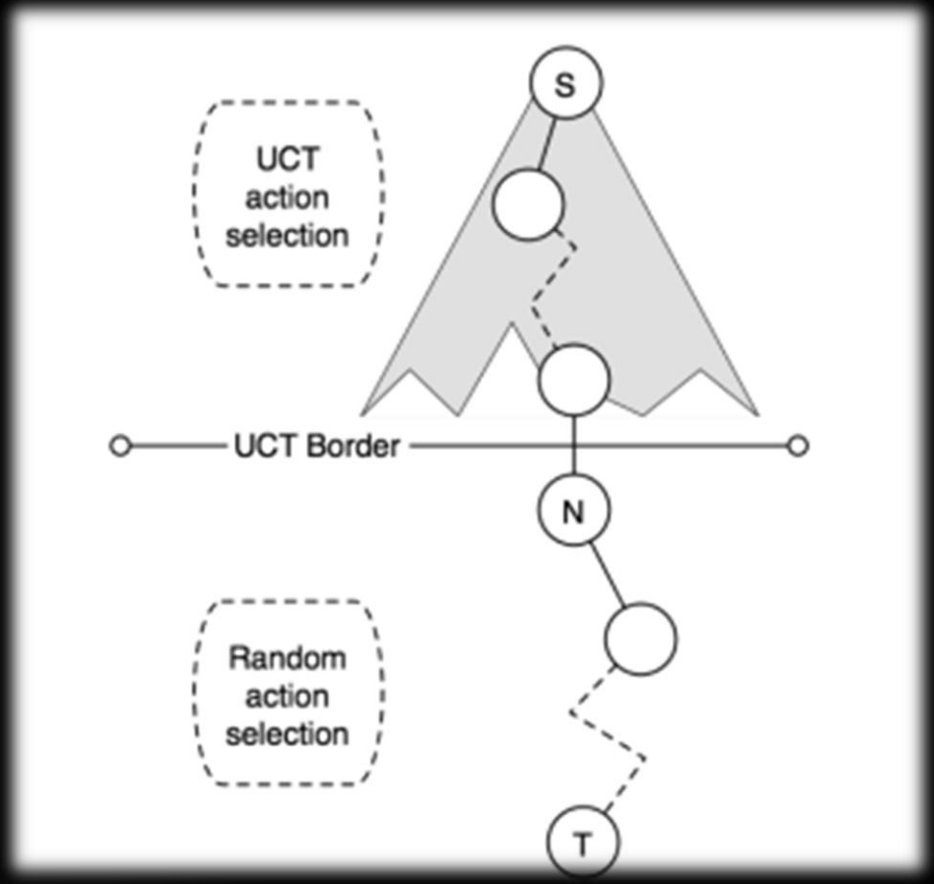
$$a^* = \arg\max_{a \in A(s)} \left\{ Q(s,a) + C \sqrt{\frac{\ln N(s)}{N(s,a)}} \right\}$$

a – action; s - state

Q(s,a) – the average return so far

N(s) – the number of visits to s

N(s,a) – the number of a selections in state s



IDEA

Let us distinguish objects build upon three types of GDL constructions:

- Table Rows (TR)
- Column Symbols (CS)
- Dynamic Symbols (DS)

Suplementary vocabulary:

- Row - ***(cell 1 1 b),..., (control xplayer)***
- Table - ***cell, control***
- Column – ***cell[0], cell[1], cell[2], control[0]***
- Symbol – ***1, 2, 3, b, x, xplayer***

```
(cell  1 1 b)
(cell  1 2 b )
(cell  1 3 b )
(cell  2 2 x )
....
(control  xplayer )
....
```

DISCOVERY PHASE - 1

Discovery of Table Rows (TR) and Column Symbols (CS)

- A few (**3-4**) parallel **random** simulations until only one is unfinished
- Synchronized with each other – common time steps

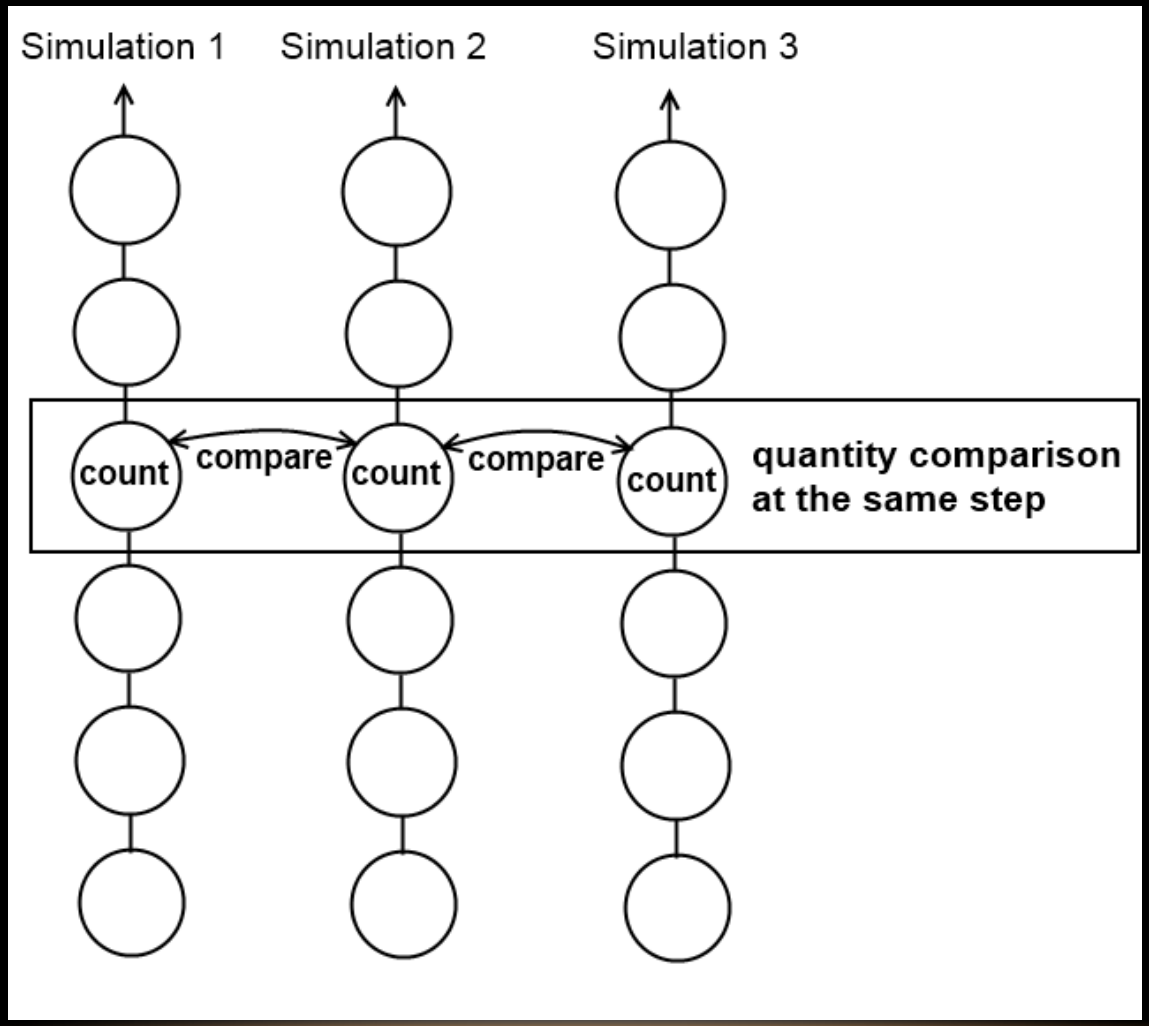
- During each time step: **TR** and **CS** are **counted**

and the quantities are subsequently compared among themselves in different simulations

The aim is to discover which TR and CS quantities vary:

- not as a result of only time step counter advancement
- as a result of performed action

DISCOVERY PHASE - 1



DISCOVERY PHASE - 2

Here, Dynamic Symbols, are selected.

- Dynamic Symbols - symbols which change symbols they appear with in rows most significantly

- One complete random simulation
- After each performed action: two one-move look-aheads.
  - Set of rows with particular candidate DS in state after i-th: action: S_i
  - after the first look-ahead: $S_{(i+1)}$
  - after the second look-ahead: $S_{(i+1)'}$

Measure of dynamism: $$val = 1 - \frac{2*|S_{(i+1)} \cap S_{(i+1)'}|}{|S_{(i+1)}| + |S_{(i+1)'}|}$$

DISCOVERY PHASE - 2

No more than one Dynamic Symbol per step is discovered

- Many have the same dynamism value => select at random
- The most varying has been already selected => no further action

Heuristic interpretation of a change of elements occurences:

- **Table Rows**:

  a novel type of state/relation has occured (e.g. check in chess)

- **Column Symbols:**

  probably some game-specific object (piece, stone, marker...) was created or destroyed

- **Dynamic Symbols:**

  property of game-specific object has been changed (e.g. position)

EVALUATION FUNCTION CONSTRUCTION

Performed during the learning phase before game starts

- play as many random simulations as possible

During each state of ***i-th*** simulation:

- Count occurences of the discovered elements:
  - ❑ Table Rows ; TR[table_name] = ...
  - ❑ Column Symbols ; CS[table_name][column_index][symbol] = ...
  - ❑ Rows containing Dynamic Symbols ; DS_R[table_name][symbol] = ...

- Compute average value AVG_i for each element respectively

EVALUATION FUNCTION CONSTRUCTION

If game result is SUCCESS

- Add AVG_i value to WIN_AVG_TOTAL

Else

- Add AVG_i value to LOSS_AVG_TOTAL

[WIN/LOSS]_AVG_TOTAL stores average values for won and lost games respectively

This values are again averaged i.e.

$$AVG_W = \frac{\sum_{i \in WIN}(AVG_i)}{|WIN|} \quad , \quad AVG_L = \frac{\sum_{i \in LOSS}(AVG_i)}{|LOSS|}$$

For each heuristic object (bucket for counted occurences):

- $weight = C * \frac{AVG_W - AVG_L}{MaxValue}$

C = 1.0 for TR, CS

C = 0.2 for rows with DS

THE PLAYER

Heuristic function:

Linear combination of weights and number of occurences of particular elements in a game state.

- take a state and compute its approximate strength evaluation

The function may be used:

- **to gradually build a game using Iterative-Deepening DFS**

- to guide Monte Carlo simulations
- to terminate MC simulation earlier
- for initial sorting of possible action-state pairs

RESULTS

Tests against reference player using MonteCarlo simulations with UCT

Variety of games:

- classic 2-player board games: chess, checkers, othello, sheep and wolf
- economic 3-player, simultaneous moves game: farmers
- match games: tic-tac-toe, connect four
- path-finding games: breakthrough, wallmaze

Various times for preparation and move

- T – unit proportional to an average simulation length [in seconds] for a particular game

RESULTS

Game	Heuristic Player vs UCT Clocks = [16T, 2T]	Heuristic Player vs UCT Clocks = [32T, 4T]	Heuristic Player vs UCT Clocks = [64T, 8T]
Bomberman **	6% - 94%	11% - 86%	21% - 70%
Breakthrough **	50% - 50%	50% - 50%	48% - 52%
Checkers *	64% - 22%	66% - 20%	84% - 10%
Chess *	14% - 0%	35% - 10%	45% - 9%
Connectfour	48% - 40%	45% - 44%	45% - 46%
Farmers	36% - 64%	32% - 68%	14% - 86%
Othello	50% - 25%	29% - 49%	38% - 49%
Pacman *	78% - 22%	75% - 25%	55% - 45%
Tic-Tac-Toe	55% - 25%	30% - 33%	44% - 46%
Sheep and Wolf *	89% - 11%	76% - 24%	70% - 30%
Wallmaze *	3% - 0%	6% - 0%	29% - 25%

RESULTS

In most games, simple changes in global state are detected such as:

- Change in number of particular pieces:
  - in chess, checkers, othello, breakthrough, sheep and wolf
  - and their estimate influence on game
  - e.g. king twice as strong than a regular piece in checkers
  - e.g queen five times stronger than pawn in chess
- Check event in chess
- In farmers: money, buildings and resources
- Some board positions like (2,2) in Tic-Tac-Toe are detected as rows with dynamics symbols

CONCLUSIONS

- Winning ratio heavily depends on type of the game
- Positive influence on playing => there is hope
- Better results in slower times => need for better/longer learning procedure

- Evaluation function in significant way changes player's strength
  - either positively (if accurate) or negatively (because of overhead or facing wrong objectives)

CONCLUSIONS

Function may be used in various scenarios:

- With MIN-MAX like search
- With MonteCarlo UCT search
- As a main strategy or just search enhancement

- Computational time plays an important role in GGP
- Promising results but inability to discover complex correlations
- GDL lexical description is very difficult to work with
- Method of evaluation function construction might be dependent on type of game
- Still plenty of room for optimizations

CURRENT AND FUTURE WORK

System aimed at GGP Competition:

- Merging various playing strategies
- Dynamic selection and evaluation of available strategies
- Deeper analysis of game rules
- More informed „quasi-blind” search during Monte Carlo Simulations
- Faster logical reasoning

Future work:

- Towards cognitive human-like playing
- Visualization and search for patterns

ANY QUESTIONS?

AFFILIATIONS AND ACKNOWLEDGEMENTS

- **Jacek Mańdziuk** – professor at Faculty of Mathematics and Computer Science (www.mini.pw.edu.pl), Warsaw University of Technology, Warsaw, Poland
- **Maciej Świechowski** – PhD student at Systems Research Institute (www.ibspan.waw.pl/) , Polish Academy of Sciences, Warsaw, Poland.

This contribution is supported by the Foundation for Polish Science
under International PhD Projects in Intelligent Computing.
Project financed from The European Union within the Innovative Economy
Operational Programme 2007-2013 and European Regional Development Fund.