Testing General Game Players Against a Simplified Boardgames Player Using Temporal-difference Learning

Jakub Kowalski, Andrzej Kisielewicz

27 may 2015

General game Playing

What is General Game Playing?

General game playing is the design of artificial intelligence programs to be able to play more than one game successfully.

Wikipedia, The Free Encyclopedia

A General Game Playing System is one that can accept a formal description of a game and play the game effectively without human intervention.

Michael Genesereth, Nathaniel Love; Stanford University

General Game Playing is about playing games that you've never seen before.

Sam Schreiber; ggp.org

Game Description Language (GDL)

- Strictly declarative, based on the *Knowledge Interchange Format*;
- Finite, deterministic games with full information and simultaneous moves;
- Pure first-order logic: no built-in assumptions, no arithmetic, no physics, no domain specific concepts;
- Prefix notation rules with ? preceded variables.

GDL Keywords

`(role ?r)`	`?r` is a player
`(init ?f)`	fact `?f` is true in the initial state
`(true ?f)`	fact `?f` is true in the current state
`(legal ?r ?a)`	in the current state `?r` can perform action `?a`
`(does ?r ?a)`	`?r` performed action `?a` in the previous state
`(next ?f)`	`?f` will be true in the next state
`terminal`	current state is terminal
`(goal ?r ?n)`	player `?r` score is `?n`

State of the Art

Research

- Knowledge extraction (heuristic evaluation functions, subgame detection, . . .)
- MCTS heuristics (Distance Features, PAST, FAST, . . .)
- GDL Reasoning (compilation, propositional networks, . . .)

Domain comparison

- International General Game Playing Competition
- GGP.org Tiltyard Server
- various other competitions

Outside comparison

None

State of the Art

Research

- Knowledge extraction (heuristic evaluation functions, subgame detection, . . .)
- MCTS heuristics (Distance Features, PAST, FAST, . . .)
- GDL Reasoning (compilation, propositional networks, . . .)

Domain comparison

- International General Game Playing Competition
- GGP.org Tiltyard Server
- various other competitions

Outside comparison

None

Idea

Create less general comparison class.

Simplified Boardgames

Definition (Björnsson 2012)

- Rectangular board $n \times m$ squares;
- Two players: black and white, controlling an army of pieces;
- At most one piece per square;
- Fixed initial position;
- Turn based, white starting;
- On its turn, a player moves one of its pieces from its current square to a different one;
- Any piece on the destination square is *captured*;
- Every game is zero-sum;
- Winning conditions:
  - reaching a *goal* square using a piece of a certain type,
  - opponent has no legal moves,
- *Draw* occurs when a preset maximum game length is reached.

Definition (Björnsson 2012)

- Rectangular board $n \times m$ squares;
- Two players: black and white, controlling an army of pieces;
- At most one piece per square;
- Fixed initial position;
- Turn based, white starting;
- On its turn, a player moves one of its pieces from its current square to a different one;
- Any piece on the destination square is *captured*;
- Every game is zero-sum;
- Winning conditions:
  - reaching a *goal* square using a piece of a certain type,
  - opponent has no legal moves,
  - opponent has less pieces of a certain type than some fixed constant;
- *Draw* occurs when a preset maximum game length is reached.

Legal moves

Set of piece's move rules

Regular language over an alphabet Σ containing triplets $(\Delta x, \Delta y, on)$:

- Δx, Δy are relative column/row distances;
- $on \in \{e, p, w\}$ describes a content of a destination square:
  - e – an empty square,
  - p – a square occupied by an o**p**ponent piece,
  - w – a square occupied by an o**w**n piece.

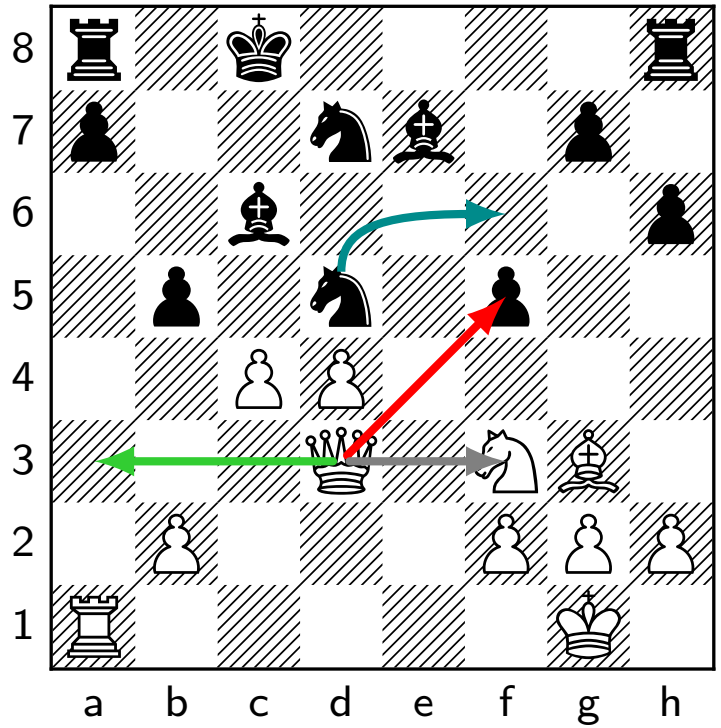
Legal move rules

Consider a *rule* $w \in \Sigma^*$, such that $w = a_1 a_2 \ldots a_k$, each $a_i = (\Delta x_i, \Delta y_i, on_i)$, and a piece standing on a square $\langle x, y \rangle$.

Rule w is applicable if and only if, for every $i \leqslant k$, the content condition on_i is fulfilled by the square $\langle x + \sum_{j=1}^{i} \Delta x_j, y + \sum_{j=1}^{i} \Delta y_j \rangle$.

If a move rule w is applicable, then the *move* transferring a piece from $\langle x, y \rangle$ to $\langle x + \sum_{i=1}^{k} \Delta x_k, y + \sum_{k=1}^{k} \Delta y_i \rangle$ is *legal*.

Move examples



- ♕d3-a3: $(-1, 0, e)(-1, 0, e)(-1, 0, e)$
- ♕d3-f5: $(1, 1, e)(1, 1, p)$
- ♞d5-f6: $(2, 1, e)$
- ♕d3-f3: ~~$(1, 0, e)(1, 0, w)$~~

Our goal

Simplified (Regular) Boardgames as a comparison class for GGP

1. Fit Simplified Boardgames into the GGP framework.

Our goal

Simplified (Regular) Boardgames as a comparison class for GGP

1. Fit Simplified Boardgames into the GGP framework.
2. Create a simple, reasonable Simplified Boardgames agent.

Our goal

Simplified (Regular) Boardgames as a comparison class for GGP

1. Fit Simplified Boardgames into the GGP framework.
2. Create a simple, reasonable Simplified Boardgames agent.
3. Run comparison tests between GGP players and sample Simplified Boardgames agent.

RBgPlayer

How to play an unknown boardgame?

Apply standard methods

- Alpha-beta Min-Max algorithm
- Evaluation function?

How to play an unknown boardgame?

Apply standard methods

- Alpha-beta Min-Max algorithm
- Evaluation function:
  Linear evaluation based on some set of features $\mathcal{F}$:

$$E(p) = \sum_{f \in \mathcal{F}} w_f \cdot f(p) \tag{1}$$

- Features set?

How to play an unknown boardgame?

Apply standard methods

- Alpha-beta Min-Max algorithm
- Evaluation function:
  Linear evaluation based on some set of features $\mathcal{F}$:

$$E(p) = \sum_{f \in \mathcal{F}} w_f \cdot f(p) \tag{1}$$

- Features set:
  - Material (piece values):
    ♙ 100, ♘ 320, ♗ 330, ♖ 500, ♕ 900, ♔ 20000
  - Position (piece-square values):

♙

0	0	0	0	0	0	0	0
50	50	50	50	50	50	50	50
10	10	20	30	30	20	10	10
5	5	10	25	25	10	5	5
0	0	0	20	20	0	0	0
5	-5	-10	0	0	-10	-5	5
5	10	10	-20	-20	10	10	5
0	0	0	0	0	0	0	0

♗

-20	-10	-10	-10	-10	-10	-10	-20
-10	0	0	0	0	0	0	-10
-10	0	5	10	10	5	0	-10
-10	5	5	10	10	5	5	-10
-10	0	10	10	10	10	0	-10
-10	10	10	10	10	10	10	-10
-10	5	0	0	0	0	5	-10
-20	-10	-10	-10	-10	-10	-10	-20

How to find features weights?

Precomputed heuristic

- mobility,
- ability to capturing opponent's pieces,
- pieces importance (based on the winning conditions),
- distance to goal squares.

Learn

Use *Temporal-difference learning.*

$$\Delta w_{t,f} = \alpha(P_{t+1} - P_t) \sum_{k=1}^{t} \lambda^{t-k} \nabla_{w_f} P_k, \tag{2}$$

- P_t is a position evaluation at time t,
- α is a global learning rate,
- $\nabla_{w_f} P_k$ is the partial derivative of P_k with respect to w_f,
- λ is a weight, exponentially decaying for more distant outcomes.

Learning details

Weight update

1. Transformation of the raw evaluation E into a sigmoid function (to reflect probability):
$$P(p) = \frac{1}{1 + e^{-\omega E(p)}}, \tag{3}$$
where ω influence the curve steepness,
2. The weight update $\nabla_{w_f} P_k$ for the feature f is computed using the following formula:
$$\nabla_{w_f} P_k = f \cdot P_k \cdot (1 - P_k). \tag{4}$$

Algorithm

- TDLeaf(λ) algorithm (uses leaf scores to evaluate positions);
- α value updated using Temporal-Coherence algorithm;
- λ value set to 0.991;
- Separate features for black and white pieces;
- Weight updates computed for both sides, average taken.

Algorithm overview

During RBgPlayer's turn

Searching phase:

- iterative-deepening.
- parallelization,
- quiescence search,
- killer-move heuristic,
- non-persistent transposition tables (updates after capture).

During startclock and opponents' turn

Learning phase:

- shallow search playouts,
- no transposition tables and quiescence search,
- feature weights update after every simulation,
- α value update after every tenth simulation.

Experiments

Experimental setup

- 2 reference GGP players,
- 4 test boargames,
- 2 RBgPlayer settings: precomputed initial heuristic and learning only,
- 2 clock settings,
- 20 games per setting (10 per side).

Reference GGP Players

Sancho (Steve Draper, Andrew Rose)

Winner of 2014 IGGPC.

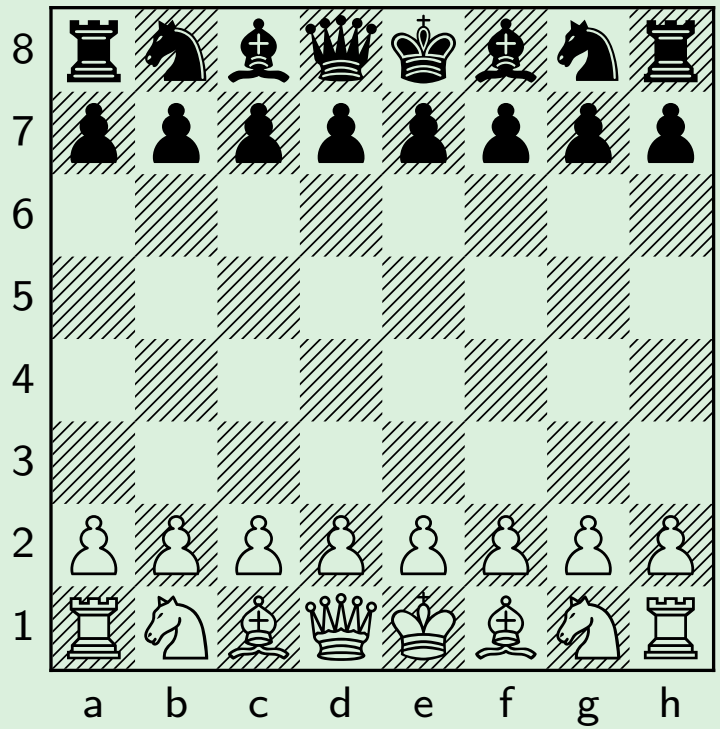
- Monte Carlo (UCT);
- Propositional networks;
- **piece detection mechanism**.

Dumalion (Jakub Kowalski, Marek Szykuła)

Top eight during 2014 IGGPC.

- Monte Carlo (UCT);
- GDL to C++ compiler;
- Parallelization over a cluster of computers.

Regular Chess



Regularity restrictions:

- no pawn double-move,
- no en-passant capture,
- no castling.

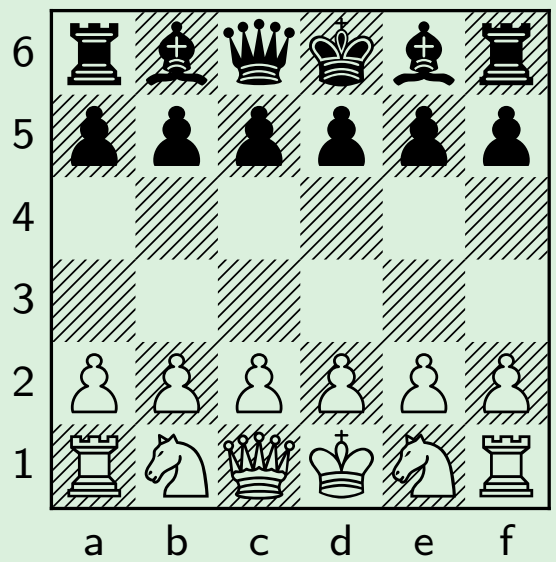
Winning conditions:

- capture opponent's king,
- move pawn to opponent's backrank.

Turn limit:

- 200

Regular Asymmetric Los Alamos



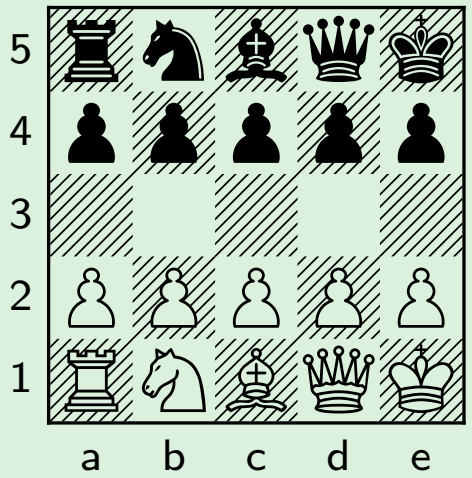
Winning conditions:

- capture opponent's king,
- move pawn to opponent's backrank.

Turn limit:

- 120

Regular Gardner



Regularity restrictions:

- no pawn double-move,
- no en-passant capture,
- no castling.

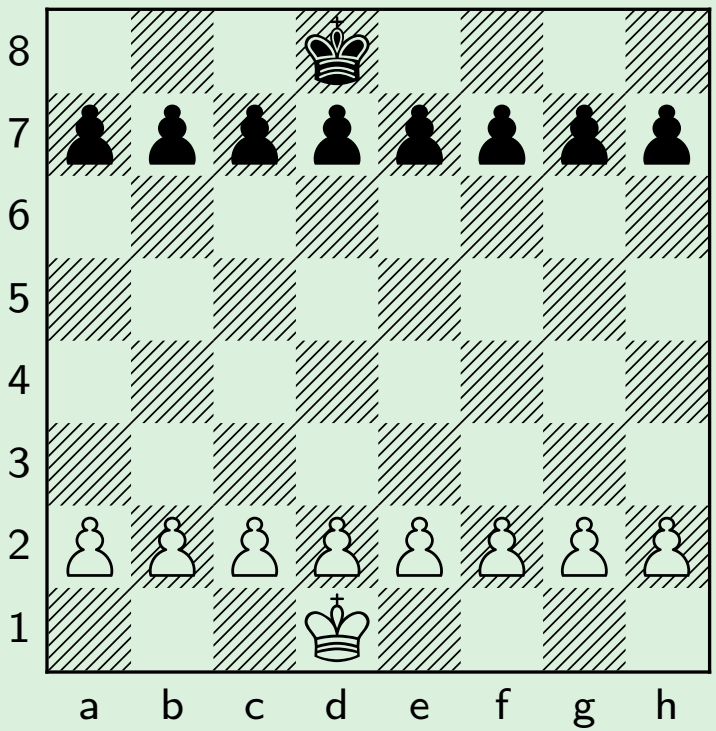
Winning conditions:

- capture opponent's king,
- move pawn to opponent's backrank.

Turn limit:

- 100

Regular Escort Breakthrough



Moves of a pawn:

Moves of a king:

Winning conditions:

- move king to opponent's backrank.

Turn limit:

- 60

Results

Precomputed initial weights

Game	startclock, playclock	Dumalion	Sancho	
			det. off	det. on
Chess	180, 60	100 : 0 : 0	100 : 0 : 0	15 : 5 : 80
	60, 20	100 : 0 : 0	95 : 5 : 0	25 : 5 : 70
Asymmetric Los Alamos	180, 60	100 : 0 : 0	100 : 0 : 0	50 : 0 : 50
	60, 20	100 : 0 : 0	100 : 0 : 0	75 : 0 : 25
Gardner	180, 60	100 : 0 : 0	40 : 5 : 55	20 : 15 : 65
	60, 20	100 : 0 : 0	45 : 10 : 45	25 : 5 : 70
Escort Breakthrough	180, 60	55 : 20 : 25	75 : 10 : 15	35 : 30 : 35
	60, 20	35 : 15 : 50	85 : 10 : 5	35 : 5 : 60

(wins:draws:loses percent)

Precomputed initial weights

Game	startclock, playclock	Dumalion	Sancho det. off	Sancho det. on
Chess	180, 60	100 : 0 : 0	100 : 0 : 0	15 : 5 : 80
	60, 20	100 : 0 : 0	95 : 5 : 0	25 : 5 : 70
Asymmetric Los Alamos	180, 60	100 : 0 : 0	100 : 0 : 0	50 : 0 : 50
	60, 20	100 : 0 : 0	100 : 0 : 0	75 : 0 : 25
Gardner	180, 60	100 : 0 : 0	40 : 5 : 55	20 : 15 : 65
	60, 20	100 : 0 : 0	45 : 10 : 45	25 : 5 : 70
Escort Breakthrough	180, 60	55 : 20 : 25	75 : 10 : 15	35 : 30 : 35
	60, 20	35 : 15 : 50	85 : 10 : 5	35 : 5 : 60

(wins:draws:loses percent)

- Piece detection mechanism drastically increases Sancho's results.
- Equal strength without pieces detection: Gardner.
- Smaller game tree depth favors GGP players.
- Smaller timeouts usually favor RBgPlayer.

Pure learning

Game	startclock, playclock	Dumalion	Sancho	
			det. off	det. on
Chess	180, 60	100 : 0 : 0	95 : 0 : 5	0 : 0 : 100
Asymmetric Los Alamos	180, 60	100 : 0 : 0	80 : 0 : 20	5 : 0 : 95
Gardner	180, 60	95 : 0 : 5	10 : 0 : 90	15 : 10 : 75
Escort Breakthrough	180, 60	15 : 55 : 30	0 : 40 : 60	5 : 20 : 75

(wins:draws:loses percent)

Pure learning

Game	startclock, playclock	Dumalion	Sancho	
			det. off	det. on
Chess	180, 60	100 : 0 : 0	95 : 0 : 5	0 : 0 : 100
Asymmetric Los Alamos	180, 60	100 : 0 : 0	80 : 0 : 20	5 : 0 : 95
Gardner	180, 60	95 : 0 : 5	10 : 0 : 90	15 : 10 : 75
Escort Breakthrough	180, 60	15 : 55 : 30	0 : 40 : 60	5 : 20 : 75

(wins:draws:loses percent)

- Knowledge of the game structure itself is often enough to obtain good results.
- Learning proper weights takes a lot of time.
- Without precomputed base weights, all values are too dependent on the local situation.

Pieces values

Piece	Chess		Assymetric Los Alamos		Gardner	
	RBg	Sancho	RBg	Sancho	RBg	Sancho
♙	1 (0.38)	1.00	1 (0.47)	1.00	1 (0.54)	1.00
♘	2.50	3.69	1.97	2.80	1.57	4.34
♗	3.00	2.69	2.01	1.40	1.51	1.24
♖	4.18	2.59	2.88	1.5/1.2	2.20	1.25
♕	7.16	5.73	4.90	3.30	3.70	2.16
♔	161.60 (158.48)	2.99	150.46 (147.71)	2.00	138.59 (136.24)	1.12

Pieces values

Piece	Chess		Assymetric Los Alamos		Gardner	
	RBg	Sancho	RBg	Sancho	RBg	Sancho
♙	1 (0.38)	1.00	1 (0.47)	1.00	1 (0.54)	1.00
♘	2.50	3.69	1.97	2.80	1.57	4.34
♗	3.00	2.69	2.01	1.40	1.51	1.24
♖	4.18	2.59	2.88	1.5/1.2	2.20	1.25
♕	7.16	5.73	4.90	3.30	3.70	2.16
♔	161.60 (158.48)	2.99	150.46 (147.71)	2.00	138.59 (136.24)	1.12

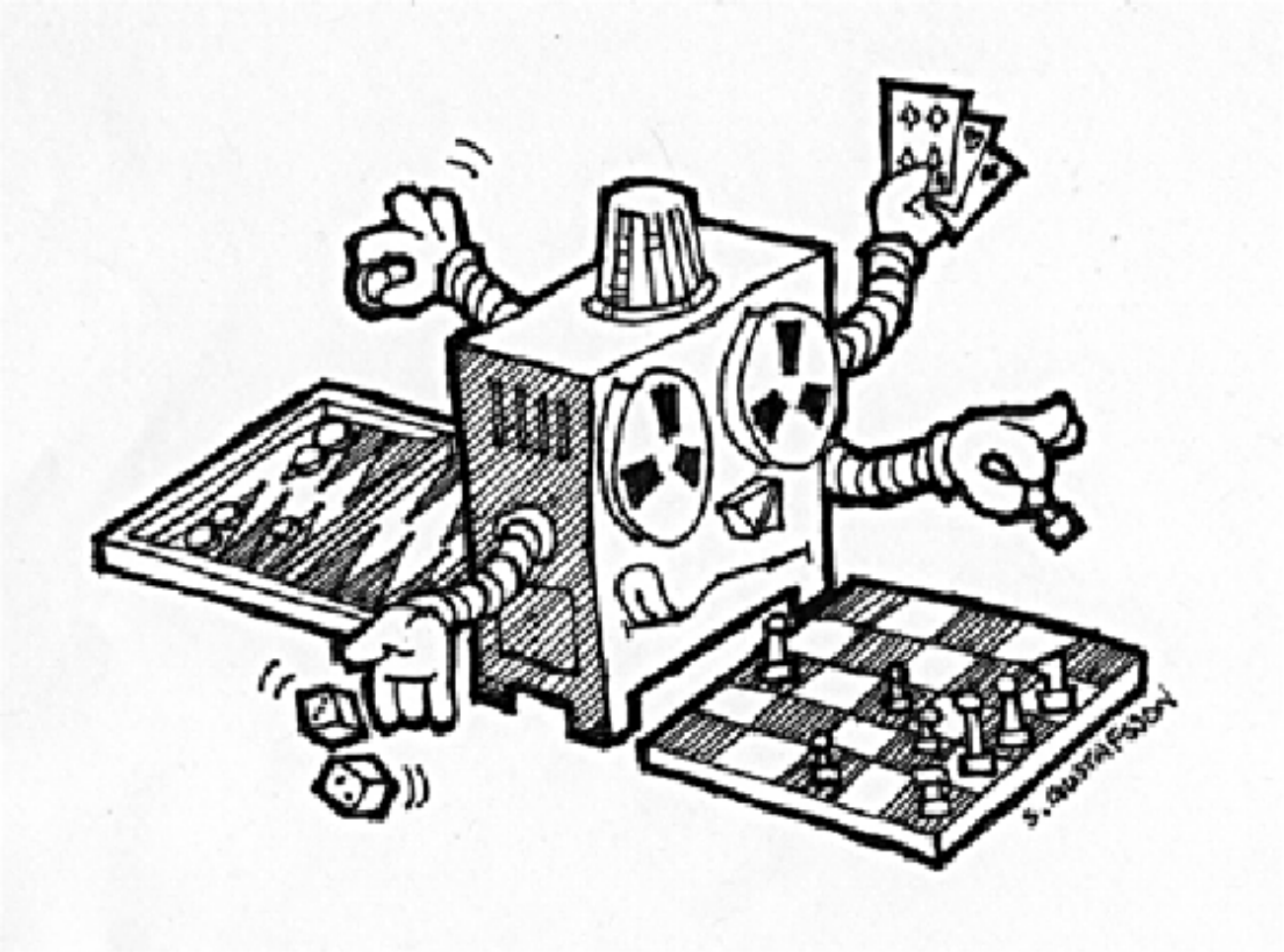
- Obtained weights usually coincide with human judgment.
- Winning conditions have great influence on piece weights.
- Sancho's algorithm has a tendency to overrate knights.

Conclusions

- GGP programs are able to play small boardgames at an acceptable level.
- Results of Sancho with piece detection show the importance of GDL domain knowledge retrieval.
- This should lead to the task of defining GGP as a union of specialized subclasses.

Future work

- Improved piece/boardgame detection and usage in MCTS
- Extending Simplified Boardgames class
- Evolving/generating boardgames
- Provide an automatic translation mechanism from Simplified Boardgames code into an efficient GDL code
- RBgPlayer improvements (opening/endgame libraries, better evaluation function)



Thank you