Evolving Chess-like Games Using Relative Algorithm Performance Profiles

Jakub Kowalski, Marek Szykuła

EvoApplications

31 March 2016

Procedural Content Generation

Procedural Content Generation of complete games

Task

- Generate complete rules of high quality games

Applications

- Games for humans
- Games for AI (competitions)

Problems

- Game rules are usually complex
- How to measure quality of a game?

Relative Algorithm Performance Profiles

RAPP (Nielsen et al. 2015)

Idea

In good games,
better algorithms should play better than worse algorithms.

Application

Evaluate game quality
by comparing performance of different algorithms.

RAPP (Nielsen et al. 2015)

Idea

In good games,
better algorithms should play better than worse algorithms.

Application

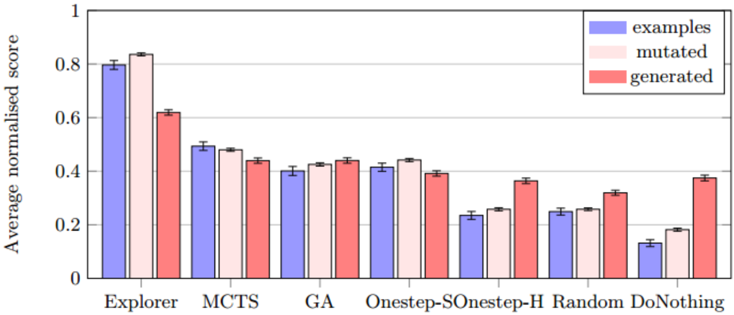
Evaluate game quality
by comparing performance of different algorithms.

GVG-AI tests

- Atari-like games (VGDL)
- Simulation-based framework (Java)
- Real-time responses.
- One player games only (puzzles).

Research (Nielsen et al., EvoGAMES 2015)

Performance comparison



Research, cont. (Nielsen et al., IEEE CIG 2015)

Evolution

- Based on two controllers: *DeepSearch* and *DoNothing*
- Fitness function:

$$\frac{RD(score) + RD(wins) + win\_50 + win\_lose}{4}, \text{where:}$$

  - *RD* means relative difference
  - win_50 is -1 if win in fewer than 50 frames
  - win_lose is 1 if the game can be both won and lost

RAPP extension

- Generalization of RAPP;
- Formalization of the method;
- Application for two player, zero-sum games;
- Generation and evolution tests.

Simplified Boardgames

Simplified Boardgames (Björnsson, ECAI 2012)

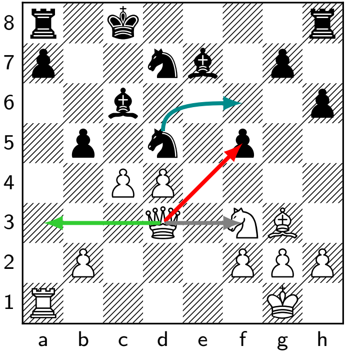
- Turn based; two players; zero-sum games;
- Rectangular board; fixed initial position; max one piece per square;
- One piece movement per turn;
- *Capturing* only at destination square;
- Winning conditions:
  - reaching a *goal* square using a certain piece,
  - captured some number of opponent's pieces;
- *Draw* occurs when the preset maximum game length is reached.

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 the content of the destination square:
  - e – **e**mpty square,
  - p – square occupied by an o**pp**onent piece,
  - w – square occupied by an o**w**n piece.

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: ~~$(1, 0, e)(1, 0, w)$~~

*Deep Blue vs Garry Kasparov, 1997, Game 6, Move 19 (last)

Simplified Chess example

```
<--BOARD-->
8 8
rnbqkbnr
pppppppp
........
........
........
........
PPPPPPPP
RNBQKBNR
```

```
<--GOALS-->
200 &
@P 0 7, 1 7, 2 7, 3 7, 4 7, 5 7, 6 7, 7 7 &
@p 0 0, 1 0, 2 0, 3 0, 4 0, 5 0, 6 0, 7 0 &
#K 1 &
#k 1 &
```

```
<--PIECES-->
P (0,1,e) + (-1,1,p) + (1,1,p) +
  (0,1,e)(0,5,e)(0,-4,e) + ... &
N (2,1,e) + (2,-1,e) + (-2,1,e) + (-2,-1,e) + (1,2,e)... &
R (0,1,e)^* + (0,1,e)^*(0,1,p) + (0,-1,e)^* + ... &
```

Generalized RAPP

Model selection

1. Defining the set of *example games*
2. Defining the set of *example algorithms* (player profiles)
3. (Selecting *model games*)
4. Selecting *evaluation algorithms*
5. Generation and evolution

Example sets

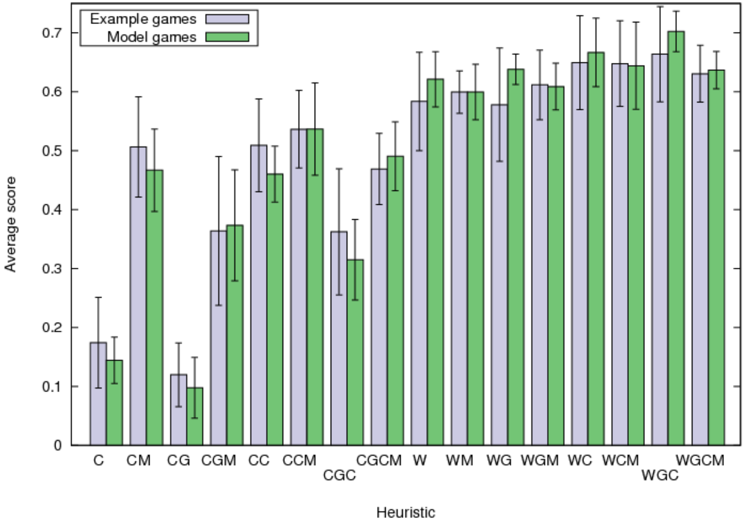
Example games

- Gardner,
- Action Man's Chess,
- Petty Chess,
- Half Chess,
- Demi-chess,
- Los Alamos Chess,
- Cannons and Crabs,
- Small-Deacon Chess,
- Shatranj,
- Chess.

Example algorithms (heuristic functions)

- **C**onstant/**W**eighted
- + **M**obility
- + **C**ontrol
- + **G**oal

Average score of algorithms



Extracting model

Model games

Division into two sets with minimal sum of distances

$$\mathrm{dist}(\mathcal{G},\mathcal{H}) = \frac{\Sigma_{i=1}^{n}\Sigma_{j=i+1}^{n}(P_{\mathcal{G}}[i,j] - P_{\mathcal{H}}[i,j])^2}{n(n-1)/2}. \quad (1)$$

Result: *Action Man's Chess*, *Cannons And Crabs*, *Chess*, *Los Alamos*, *Shatranj*, *Small-Deacon*.

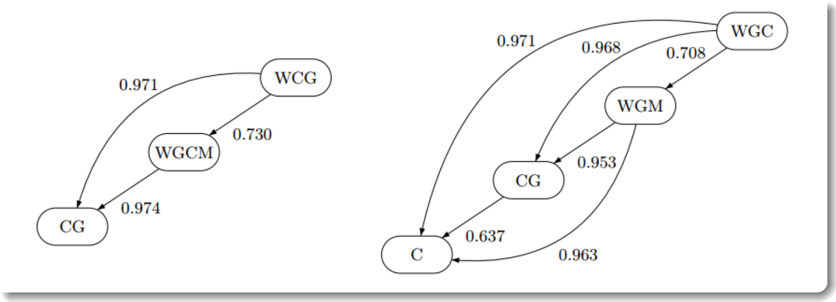
Model algorithms

Take k algorithms which maximize spread

$$\mathrm{spread}(P_{model}) = \Sigma_{i=1}^{k}\Sigma_{j=i+1}^{k}|P_{model}[i,j] - 0.5|^2. \quad (2)$$

Result ($k = 3$): *CG*, *WGC*, *WGCM*
Result ($k = 4$): *C*, *CG*, *WGM*, *WGC*
Result ($k = 5$): *C*, *CG*, *WGM*, *WGC*, *WGCM*.

Representation graph of the relative performance models



Generation

Restrictions (chess-like games)

- Symmetric initial position
- Two rows of pieces per player.
- Front row contains only *pawns* (or empty squares)
- Back row contain one piece of the *king* type.
- Win by capturing the enemy king.
- Win by reaching the opponent's back row using a pawn.

Parameters

- Board *width*, *height* $\in \{6, 7, 8\}$,
- Number of *non-winning* figures $\in \{3, 4, 5\}$.
- $\mathit{turnlimit} = 3 \times \mathit{width} \times \mathit{height} + \mathit{random}(\{0, \ldots, 19\})$.

Fitness function

Features

- $B = \frac{|score_w - score_b|}{n}$ is a *balance*, where
  - n is the number of plays,
  - $score_w$ be the percent of points scored by white player,
  - $score_b$ be the percent of points scored by black player.
- $Q = \frac{s}{n}$ is a game's *quickness*, where
  - game is *too short* if it ends in 10 turns,
  - s plays were qualified as *too short*.
- D is distance to the model using modified formula (1),

$$\mathrm{dist}(\mathcal{G}, P_{model}) = \frac{\sum_{i=1}^{n}\sum_{j=i+1}^{n}(|P_{\mathcal{G}}[i,j] - P_{\mathcal{H}}[i,j]|)^{1}}{n(n-1)/2}. \tag{1a}$$

Fitness value

$$f = \begin{cases} (1-D)(1-B)(1-Q) & \text{if game is } \textit{playable}, \\ -1 & \text{otherwise.} \end{cases} \tag{3}$$

Genetic operators

Crossover

- Roulette wheel parents selection
- Uniform crossover (except king squares)

Mutation

- *Piece mutation* regenerates the rules of a random piece.
- *Position mutation* changes the content of a random square.

Selection

- Best n games from parents and children sets.

Experiment setup

Generation

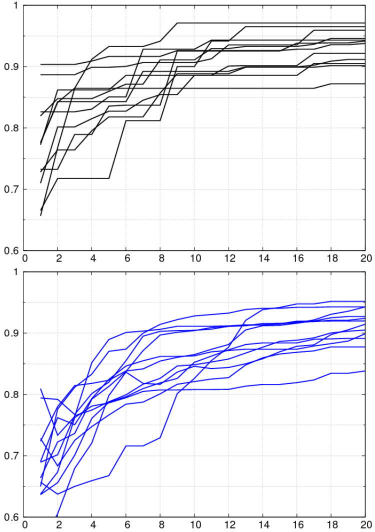
- 4 parameter sets (for piece rules generation);
- 200 generated games (50 per set);
- evaluated using 3 and 4 model algorithms.

Evolution

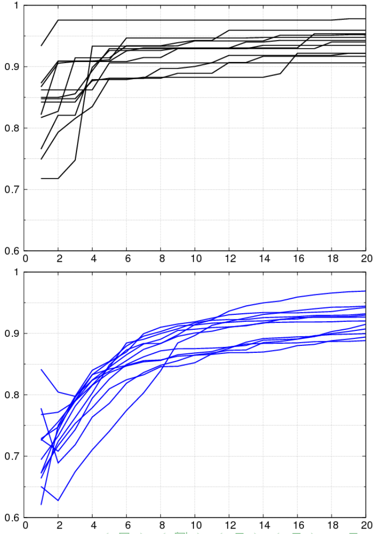
- same 4 parameter sets;
- 20 generations;
- 3 and 4 model algorithms: 12 runs with population size 10;
- 4 algorithms: 12 runs, population size 16, increased mutation rate.

Evolution results

3 algs.; population size 12



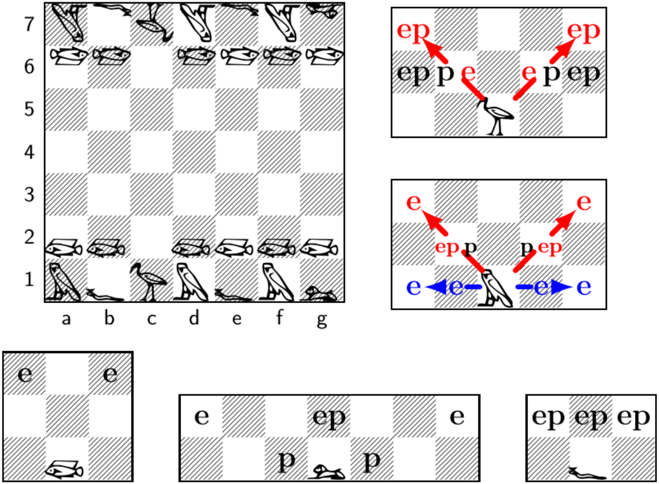
4 algs.; population size 16



Results comparison

Variant	3 algs.			4 algs.		
	Max.	Avg.	Promising	Max.	Avg.	Promising
Generated	0.907	0.671	29%	0.942	0.704	34%
Evolved(12)	0.971	0.911	100%	0.959	0.916	100%
Evolved(16)				0.978	0.922	100%
Example	0.858	0.811	90%	0.959	0.859	100%

Example of evolved game (fitness 0.9538)



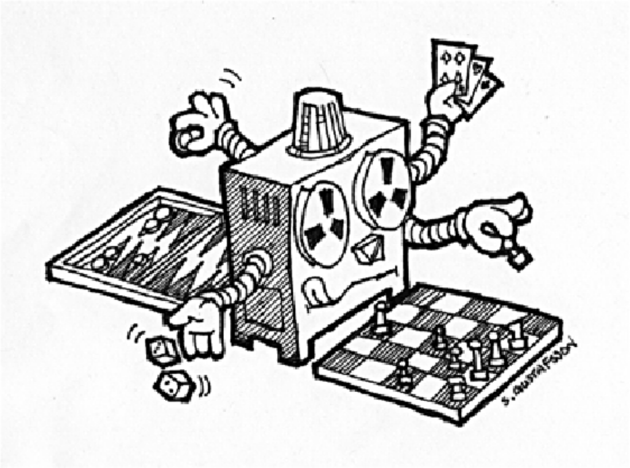
Summary

Contribution

- Generalization and formalization of RAPP approach for games evaluation.
- Application in Simplified Boardgames class.

Conclusions

- Works best as a sieve with human intervention in the last stage.
- Time consuming (requires many expensive simulations).
- By using MCTS with different time limits can be made knowledge-free.



Thank you