Evolving Chess-like Games Using Relative Algorithm Performance Profiles

Jakub Kowalski, Marek Szykuła

EvoApplications
31 March 2016
Procedural Content Generation

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?

Jakub Kowalski, Marek Szykuła
Evolving Chess-like Games Using Relative Algorithm Performance Profiles
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

The graph shows a performance comparison of different algorithms for generating games. The x-axis represents different algorithms: Explorer, MCTS, GA, Onestep-SOnestep-H, Random, and DoNothing. The y-axis represents the average normalized score. The algorithms are compared based on their performance, with examples, mutated, and generated scores. The graph visually illustrates the relative performance of each algorithm.
Research, cont. (Nielsen et al., IEEE CIG 2015)

Evolution

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

$$\frac{RD(score) + RD(wins) + \text{win}_{50} + \text{win}_{lose}}{4},$$

where:

- $RD$ means relative difference
- $\text{win}_{50}$ is -1 if win in fewer than 50 frames
- $\text{win}_{lose}$ 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

Jakub Kowalski, Marek Szykuła
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 $\Sigma$ containing triplets $(\Delta x, \Delta y, on)$:
- $\Delta x, \Delta y$ are relative column/row distances;
- $on \in \{e, p, w\}$ describes the content of the destination square:
  - e – empty square,
  - p – square occupied by an opponent piece,
  - w – square occupied by an own 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)
- Constant/Weighted
- + Mobility
- + Control
- + Goal
Average score of algorithms

![Graph showing average score of algorithms for different heuristics. The x-axis represents different heuristics, including CG, CM, CGM, CC, CCM, CGCM, W, WM, WG, WGM, WC, WCM, and WGC. The y-axis represents the average score, ranging from 0 to 0.7. The graph compares example games and model games, with error bars indicating variability.](image-url)
Evolving Chess-like Games Using Relative Algorithm Performance Profiles

Extracting model

Model games

Division into two sets with minimal sum of distances

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

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

Model algorithms

Take \(k\) algorithms which maximize spread

\[
\text{spread}(P_{\text{model}}) = \sum_{i=1}^{k} \sum_{j=i+1}^{k} |P_{\text{model}}[i,j] - 0.5|^2.
\] (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

Model

PCG

RAPP

Simplified Boardgames

Generalized RAPP

The End
Restrictions (chess-like games)

- Symmetric initial position
- Two rows of pieces per player.
- Front row contains only pawns (or empty squares)
- Back row contains 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 ∈ \{6, 7, 8\},
- Number of non-winning figures ∈ \{3, 4, 5\}.
- turnlimit = 3 × width × height + \text{random}(\{0, \ldots, 19\}).
Fitness function

Features

- \( B = \frac{|\text{score}_w - \text{score}_b|}{n} \) is a balance, where
  - \( n \) is the number of plays,
  - \( \text{score}_w \) be the percent of points scored by white player,
  - \( \text{score}_b \) 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),

\[
\text{dist}(G, P_{\text{model}}) = \frac{\sum_{i=1}^{n} \sum_{j=i+1}^{n} (|P_g[i,j] - P_H[i,j]|)}{n(n-1)/2}.
\]

Fitness value

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

Jakub Kowalski, Marek Szykuła
Evolving Chess-like Games Using Relative Algorithm Performance Profiles
## Results comparison

<table>
<thead>
<tr>
<th>Variant</th>
<th>3 algs.</th>
<th>4 algs.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generated</td>
<td>0.907</td>
<td>0.671</td>
</tr>
<tr>
<td>Evolved(12)</td>
<td>0.971</td>
<td>0.911</td>
</tr>
<tr>
<td>Evolved(16)</td>
<td>0.978</td>
<td>0.922</td>
</tr>
<tr>
<td>Example</td>
<td>0.858</td>
<td>0.811</td>
</tr>
</tbody>
</table>
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