Testing Game Algorithms

Petr Baudiš ⟨pasky@ucw.cz⟩

MFF UK

Testing (Randomized) Game Algorithms

- The Problem
- Naive Solutions vs. Scientifically Rigorous Testing
- Automatic Tuning — Noisy Blackbox Optimization and CLOP

Outline

Possible Scenarios

- We implemented (feature, bugfix) X
  We changed the parameter vector $\vec{p}$
- What is the effect on performance?
- Solutions: **Position regression testing** or **Play testing**

- Position regression testing: Natural extension on the unit test idea
- Library of positions, does our program find the correct solution?
- Usually, X will improve evaluation of some positions and hinder evaluation of other positions
- Benefits need to be weighed on a case-by-case basis

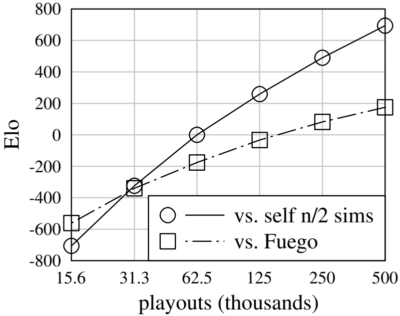
Play Testing

- After modification, we play a number of games and compare performance to the original version
- Games can be against humans (rating change) or against another program
- Games need to be randomized
- *Frequent* play testing is essential for scientifically rigorous development
- Pachi: **autotest** framework for playtesting of different program versions

Outline

Self Play Testing

- **Self-play testing:** Directly match original and new version
- Common practice (sadly sometimes still even in reviewed papers)
- Bad idea! Self-play testing massively amplifies small improvements and does not test for situations disfavored by the program.



Reference Opponent Testing

- With good handicap options in your game, the program can be even relatively weak
- Completely different algorithm is very beneficial to maintain situation diversity

- After a sequence of n games, is the average winrate better for modified or original version, with e.g. $p = 0.95$?
- Each game is a binomial trial; (central limit theorem:) sum of trials (winrate $E[w]$) is normally distributed
- 95% confidence interval is 1.96σ, where $\sigma = \sqrt{n \cdot E[w] \cdot (1 - E[w])}$

Statistical Pitfalls

- Pitfall #1: $p = 0.95$ is a very difficult tradeoff; 1 in 20 trials will still be wrong! But it can take thousands of games to get a statistically significant result.
- Pitfall #2: Number of trials n must be fixed in advance, don't stop as soon as confidence intervals stop touching for a moment!
- Pitfall #3: Winrate difference $\Delta w = 0.1$ means something different with $w_1^a = 0.5$, $w_2^a = 0.6$ and $w_1^b = 0.85$, $w_2^b = 0.95$. Humans are bad at comparing numbers on exponential scale. Convert Δw to Elo point difference, which is linearized.
- $p = 0.63$, $n = 1000$:
  $\Delta w^a = 70$ Elo $(+26 - 25)$
  $\Delta w^b = 200$ Elo $(+200 - 127)$

Outline

Parameter Tuning

- Naive approach: Vary parameter values manually, playtest each $\vec{p}$ instance against reference
- In general: Noisy BlackBox Optimization (many-dimensional)
- Area of active study by itself, many solutions: MCTS, SPSA, CEM, **CLOP**, ...
- CLOP: Specifically developed for game engine tuning, open source framework with GUI available

End of Slideshow

Time for the next topic