Theory of Computer Games: Concluding Remarks

Tsan-sheng Hsu

http://www.iis.sinica.edu.tw/~tshsu
Abstract

- Introducing practical issues.
  - The open book.
  - The graph history interaction (GHI) problem.
  - Smart usage of resources.
    - time during searching
    - memory
    - coding efforts
    - debugging efforts
  - Opponent models

- How to combine what we have learned in class together to get a working game program.
During the open game, it is frequently the case
- branching factor is huge;
- it is difficult to write a good evaluating function;
- the number of possible distinct positions up to a limited length is small as compared to the number of possible positions encountered during middle game search.

Acquire game logs from
- books;
- games between masters;
- games between computers;

Use off-line computation to find out the value of a position for a given depth that cannot be computed online during a game due to resource constraints.

...
Assume you have collected $r$ games.
- For each position in the $r$ games, compute the following 3 values:
  - $\text{win}$: the number of games reaching this position and then wins.
  - $\text{loss}$: the number of games reaching this position and then loss.
  - $\text{draw}$: the number of games reaching this position and then draw.

When $r$ is large and the games are trustful, then use the 3 values to compute a value and use this value as the value of this position.

Comments:
- Pure statistically
- You program may not be able to take over when the open book is over.
- It is difficult to acquire large amount of “trustful” game logs.
- Automatically analysis of game logs written by human experts. [Chen et. al. 2006]
- Using high-level meta-knowledge to guide the way in searching:
  - $\text{Dark chess}$: adjacent attack of the opponent’s Cannon. [Chen and Hsu 2013]
The graph history interaction (GHI) problem [Campbell 1985]:
- In a game graph, a position can be visited by more than one paths.
- The value of the position depends on the path visiting it.
- In the transposition table, you record the value of a position, but not the path leading to it.
  - Values computed from rules on repetition cannot be used later on.
  - It takes a huge amount of storage to store the path visiting it.
GHI problem – example

• $A \rightarrow B \rightarrow E \rightarrow I \rightarrow J \rightarrow H \rightarrow E$ is loss because of rules of repetition.
  ▶ Memorized $H$ is loss.

• $A \rightarrow B \rightarrow D$ is a loss.

• $A \rightarrow C \rightarrow F \rightarrow H$ is loss because $H$ is recorded as loss.

• $A$ is loss because both branches lead to loss.

• However, $A \rightarrow C \rightarrow F \rightarrow H \rightarrow E \rightarrow G$ is win.
Using resources

- **Time** [Hyatt 1984] [Šolak and Vučković 2009]
  - For human:
    - More time is spent in the beginning when the game just starts.
    - Stop searching a path further when you think the position is stable.
  - Pondering:
    - Use the time when your opponent is thinking.
    - Guessing and then pondering.

- **Memory**
  - Using a large transposition table occupies a large space and thus slows down the program.
    - A large number of positions are not visited too often.
  - Using no transposition table makes you to search a position more than once.

- **Other resources.**
Opponent models

- In a normal alpha-beta search, it is assumed that you and the opponent use the same strategy.
  - What is good to you is bad to the opponent and vice versa!
  - Hence we can reduce a minimax search to a NegaMax search.
  - This is normally true when the game ends, but may not be true in the middle of the game.

- What will happen when there are two strategies or evaluating functions $f_1$ and $f_2$ so that
  - for some positions $p$, $f_1(p)$ is better than $f_2(p)$
  - “better” means closer to the real value $f(p)$
  - for some positions $q$, $f_2(q)$ is better than $f_1(q)$

- If you are using $f_1$ and you know your opponent is using $f_2$, what can be done to take advantage of this information?
  - This is called OM (opponent model) search [Carmel and Markovitch 1996].
  - In a MAX node, use $f_1$.
  - In a MIN node, use $f_2$. 
Opponent models – comments

Comments:
- Need to know your opponent model precisely.
- How to learn the opponent on-line or off-line?
- When there are more than 2 possible opponent strategies, use a probability model (PrOM search) to form a strategy.
Putting everything together

- **Game playing system**
  - Use some sorts of open book.
  - Middle-game searching: usage of a search engine.
    - Main search algorithm
    - Enhancements
    - Evaluating function: knowledge
  - Use some sorts of endgame databases.
Assume during a selfplay experiment, two copies of the same program are playing against each other.

- Since two copies of the same program are playing against each other, the outcome of each game is an independent random trial and can be modeled as a trinomial random variable.
- Assume for a copy playing first,

\[
Pr(game_{first}) = \begin{cases} 
    p & \text{if won the game} \\
    q & \text{if draw the game} \\
    1 - p - q & \text{if lose the game}
\end{cases}
\]

- Hence for a copy playing second,

\[
Pr(game_{last}) = \begin{cases} 
    1 - p - q & \text{if won the game} \\
    q & \text{if draw the game} \\
    p & \text{if lose the game}
\end{cases}
\]
Outcome of selfplay games

- Assume 2n games, \( g_1, g_2, \ldots, g_{2n} \) are played.
  - In order to offset the initiative, namely first player’s advantage, each copy plays first for \( n \) games.
  - We also assume each copy alternatives in playing first.
  - Let \( g_{2i-1} \) and \( g_{2i} \) be the \( i \)th pair of games.

- Let the outcome of the \( i \)th pair of games be a random variable \( X_i \) from the prospective of the copy who plays \( g_{2i-1} \).
  - Assume we assign a score of \( x \) for a game won, a score of 0 for a game drawn and a score of \(-x\) for a game lost.

- The outcome of \( X_i \) and its occurrence probability is thus

\[
Pr(X_i) = \begin{cases} 
  p(1 - p - q) & \text{if } X_i = 2x \\
  pq + (1 - p - q)q & \text{if } X_i = x \\
  p^2 + (1 - p - q)^2 + q^2 & \text{if } X_i = 0 \\
  pq + (1 - p - q)q & \text{if } X_i = -x \\
  (1 - p - q)p & \text{if } X_i = -2x 
\end{cases}
\]
How good we are against the baseline?

### Properties of $X_i$.

- The mean $\mathbb{E}(X_i) = 0$.
- The standard deviation of $X_i$ is

$$\sqrt{\mathbb{E}(X_i^2)} = \sqrt{2pq + (2q + 8p)(1 - p - q)},$$

and it is a multi-nominally distributed random variable.

### When you have played $n$ pairs of games, what is the probability of getting a score of $s$, $s > 0$?

- Let $X[n] = \sum_{i=1}^{n} X_i$.
  - **The mean of $X[n]$, $\mathbb{E}(X[n])$, is 0.**
  - **The standard deviation of $X[n]$, $\sigma_n$, is**

$$x\sqrt{n}\sqrt{2pq + (2q + 8p)(1 - p - q)},$$

- If $s > 0$, we can calculate the probability of $Pr(|X[n]| \leq s)$ using well known techniques from calculating multi-nominal distributions.
Practical setup

- Parameters that are usually used.
  - $x = 1$.
  - For Chinese chess, $q$ is about $0.3161$, $p = 0.3918$ and $1 - p - q$ is 0.2920.

  - This means the first player has a better chance of winning.

- The mean of $X[n]$, $E(X[n])$, is 0.
- The standard deviation of $X[n]$, $\sigma_n$, is

\[
x \sqrt{n} \sqrt{2pq + (2q + 8p)(1 - p - q)} = \sqrt{1.16n}.
\]
## Results (1/3)

| $Pr(|X[n]| \leq s)$ | $s = 0$ | $s = 1$ | $s = 2$ | $s = 3$ | $s = 4$ | $s = 5$ | $s = 6$ |
|---------------------|--------|--------|--------|--------|--------|--------|--------|
| $n = 10, \sigma_{10} = 3.67$ | 0.108  | 0.315  | 0.502  | 0.658  | 0.779  | 0.866  | 0.924  |
| $n = 20, \sigma_{20} = 5.19$ | 0.076  | 0.227  | 0.369  | 0.499  | 0.613  | 0.710  | 0.789  |
| $n = 30, \sigma_{30} = 6.36$ | 0.063  | 0.186  | 0.305  | 0.417  | 0.520  | 0.612  | 0.693  |
| $n = 40, \sigma_{40} = 7.34$ | 0.054  | 0.162  | 0.266  | 0.366  | 0.460  | 0.546  | 0.624  |
| $n = 50, \sigma_{50} = 8.21$ | 0.049  | 0.145  | 0.239  | 0.330  | 0.416  | 0.497  | 0.571  |
### Results (2/3)

| \( P_r(|X[n]| \leq s) \) | \( s = 7 \) | \( s = 8 \) | \( s = 9 \) | \( s = 10 \) | \( s = 11 \) | \( s = 12 \) | \( s = 13 \) |
|--------------------------|------------|------------|------------|------------|------------|------------|------------|
| \( n = 10, \sigma_{10} = 3.67 \) | 0.960      | 0.981      | 0.991      | 0.997      | 0.999      | 1.000      | 1.000      |
| \( n = 20, \sigma_{20} = 5.19 \) | 0.851      | 0.899      | 0.933      | \textbf{0.958} | 0.974      | 0.985      | 0.991      |
| \( n = 30, \sigma_{30} = 6.36 \) | 0.761      | 0.819      | 0.865      | 0.902      | 0.930      | \textbf{0.951} | 0.967      |
| \( n = 40, \sigma_{40} = 7.34 \) | 0.693      | 0.753      | 0.804      | 0.847      | 0.883      | 0.912      | 0.934      |
| \( n = 50, \sigma_{50} = 8.21 \) | 0.639      | 0.699      | 0.753      | 0.799      | 0.839      | 0.872      | 0.900      |
### Results (3/3)

| $P_r(|X[n]| \leq s)$ | $s = 14$ | $s = 15$ | $s = 16$ | $s = 17$ | $s = 18$ | $s = 19$ | $s = 20$ |
|----------------------|----------|----------|----------|----------|----------|----------|----------|
| $n = 10, \sigma_{10} = 3.67$ | 1.000    | 1.000    | 1.000    | 1.000    | 1.000    | 1.000    | 1.000    |
| $n = 20, \sigma_{20} = 5.19$ | 0.995    | 0.997    | 0.999    | 0.999    | 1.000    | 1.000    | 1.000    |
| $n = 30, \sigma_{30} = 6.36$ | 0.978    | 0.986    | 0.991    | 0.994    | 0.997    | 0.998    | 0.999    |
| $n = 40, \sigma_{40} = 7.34$ | **0.952** | 0.966    | 0.976    | 0.983    | 0.989    | 0.992    | 0.995    |
| $n = 50, \sigma_{50} = 8.21$ | 0.923    | 0.941    | **0.956** | 0.967    | 0.976    | 0.983    | 0.988    |

TCG: Putting everything together, 20131224, Tsan-sheng Hsu ©
Statistical behaviors

- Hence assume you have two programs that are playing against each other and have obtained a score of $s + 1$, $s > 0$, after trying $n$ pairs of games.
  - Assume $Pr(|X[n]| \leq s)$ is say 0.95.
    - Then this result is meaningful, that is a program is better than the other, because it only happens with a low probability of 0.05.
  - Assume $Pr(|X[n]| \leq s)$ is say 0.05.
    - Then this result is not very meaningful, because it happens with a high probability of 0.95.

- In general, it is a very rare case, e.g., less than 5% of chance that it will happen, that your score is more than $2\sigma_n$.
  - For our setting, if you perform $n$ pairs of games, and your net score is more than $2 \times \sqrt{1.16} \times \sqrt{n} \simeq 2.154 \sqrt{n}$, then it means something statistically.

- You can also decide your “definition” of “a rare case”.

TCG: Putting everything together, 20131224, Tsan-sheng Hsu ©
Concluding remarks

Consider your purpose of studying a game:

- It is good to solve a game completely.
  - You can only solve a game once!

- It is better to acquire the knowledge about why the game wins, draws or loses.
  - You can learn lots of knowledge.

- It is even better to discover knowledge in the game and then use it to make the world a better place.
  - Fun!
References and further readings (1/2)
