Computer Chinese Chess

Tsan-sheng Hsu

徐讎昇

tshsu@iis.sinica.edu.tw

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

An introduction to research problems and opportunities in Computer Games.
- Using Computer Chinese chess (象棋) as examples.
- Show how theoretical research can help in solving the problems.
  - Data-intensive computing: tradeoff between computing on the spot and using pre-stored knowledge.

Phases of games
- Open game (開局): database
- Middle game (中局): Search
- End game (殘局): knowledge

Topics:
- Introduction
- Construction of a huge knowledge base that is consistent
- Playing rules for repetition of positions
- Construction of huge endgame databases
- Benchmark
Introduction

- Why study Computer Games:
  - Intelligence requires *knowledge*.
  - Games hold an inexplicable fascination for many people, and the notion that computers might play games has existed at least as long as computers.
  - Reasons why games appeared to be a good domain in which to explore machine intelligence.
    - They provide a structured task in which it is very easy to measure success or failure.
    - They did not obviously require large amount of knowledge.

- A course on teaching computers to play games was introduced at NTU in 2007.

  電腦對局理論
# Predictions for 2010 – Status


<table>
<thead>
<tr>
<th>solved</th>
<th>over champion</th>
<th>world champion</th>
<th>grand master</th>
<th>amateur</th>
</tr>
</thead>
<tbody>
<tr>
<td>Awari</td>
<td>Chess</td>
<td>Go (9 × 9)</td>
<td>Bridge</td>
<td>Go (19 × 19)</td>
</tr>
<tr>
<td>Othello</td>
<td>Draughts (10 × 10)</td>
<td>Chinese chess</td>
<td>Shogi</td>
<td></td>
</tr>
<tr>
<td>Checkers (8 × 8)</td>
<td>Scrabble</td>
<td>Hex</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Backgammon</td>
<td>Amazons</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Lines of Action</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Over champion means definitely over the best human player.
- World champion means equaling to the best human player.
- Grand master means beating most human players.

- **color code**
  - **Red**: right on the target.
  - **Blue**: not so.
  - **Black**: have some progress towards the target.
Introduction

- **Western chess (西洋棋) programs.**
  - One of the important areas since the dawn of computing research.
    - C.E. Shannon, 1950, Computer Chess paper
  - Beat the human champion at 1997.
  - Many techniques can be used in computer Chinese chess programs.

- **Computer Chinese chess programs.**
  - About 7-dan.
  - Computing research history: more than 30 years late.
    - Started from about 1981.
Chess Related Researches

- Chess related research:
  - Open game.
    - Databases.
    - Many pseudo theories with heavy human involvements.
  - Middle game searching.
    - Traditional game tree searching.
    - In search of a good evaluating function.
  - Endgame.
    - Heuristics and knowledge.
    - Computer constructed databases.
Properties of Chinese Chess

- Several unique characteristics about Chinese chess.
  - The usage of Cannon.
    - *It is possible to attack or protect a piece at a longer range.*
  - Categories of defending and attacking pieces.
  - The positions of Pawns.
    - *The mobility of the pawn is limited before crossing the river.*
    - *It cannot move backward.*
  - Complex Chinese chess rules (棋規).
  - Palace and the protection of kings.
  - The value of each piece (子力價值). is highly dynamic:
    - *Although Knight 马 is roughly equal to Cannon 炮,*
    - *Rook 车 + Knight 马 + Cannon 炮 is better than Rook 车 + 2 Cannons 炮 炮.*
    - *Knowledge inferencing among material combinations [Chen et al. 2007,2011].*
Research Opportunities

- Some research opportunities.
  - Open game theories.
    - Learning form a vast amount of prior human knowledge.
    - In great need of some breakthrough.
  - Much larger searching space:
    - Western chess: $10^{123}$
    - Chinese chess: $10^{150}$
    - Deeper searching depth and longer game.
  - Game tree searching.
    - The usage of materials.
    - Knowledge inferencing among material combinations.
  - Endgame: contains lots of pieces.
    - The size of useful endgames is huge compared to that of Western chess.
  - Rules in facing of repetitions.
Construction of a huge knowledge base that is consistent
Motivations

- Computing of the material values is a crucial part of a good evaluating function for Chinese chess.

  - **Static material values:**
    - King: 100
    - Guard/Minister: 2
    - Rook: 10
    - Knight/Cannon: 5
    - Pawn: 1

  - **Meanings:**
    - A knight is about equal to a cannon.
    - A rook is about equal to two knights, two cannons, or a cannon plus a knight.
    - Three defending pieces are better than a knight, but two of them are as good.
Dynamic piece value

- Values of pieces are dynamic depending on the combination.
  - It is better to have different types of attacking pieces.
    - Cannons can “jump” over pieces, rooks can attack in straight-lines, and knights can attack in a very different way.
    - Guards are better in protecting the king in facing a rook attack.
    - Guards are not good in protecting the king in facing a cannon attack.

- Examples:
  - Example 1:
    - KCPGMMKGGMM is a red-win endgame.
    - KNPGMMKGGMM is a draw endgame.
  - Example 2:
    - KPPKGG and KPPKMM are red-win endgames.
    - KPPKGM is a draw endgame.
  - Example 3:
    - KNPKGM and KNPKGG are red-win endgames.
    - KNPKKMM is a difficult endgame for red to win.
Usage of Endgame Knowledge

- Computer constructed endgame databases are too large to be loaded into the main memory during searching.
  - only useful at the very end of games.

- Human experts:
  - Studies the degree of “advantageous” by considering only positions of pawns and material combinations.
  - Lots of endgame books exist.
Granularity: 12 different levels by considering material combinations (子力組合) only.

- 紅必勝 (0): The red side is almost sure to win.
- 紅大優 (1): The red side is almost sure to win, but may be draw if the black side takes a very good position.
- 紅佔優 (2): The red side has advantage, but has a chance to lose if the black side is in a very good position.
- 紅巧勝 (3): The red side may win in some good positions, but in most cases it is a draw.
- 紅難勝 (4): The red side has an advantage, but is very difficult to win.
- 均勢 (5): Either side has a chance to win, i.e., tie.
- 必和 (6): No side can win, i.e., draw.
- 黒難勝 (7): The black side has an advantage, but is very difficult to win.
- 黒巧勝 (8): The black side may win in some good positions, but in most cases it is a draw.
- 黒佔優 (9): The black side has advantage, but has a chance to lose if the red side is in a very good position.
- 黒大優 (10): The black side is almost sure to win, but may be draw if the red side takes a very good position.
- 黒必勝 (11): The black side is almost sure to win.
Motivations

- There are many existing heuristics about Chinese Chess endgames.
  - Books.
  - Computer records.
  - Annotations from human experts.
  - ...

- Previously, efforts are spent to collect heuristics.

- Now, our problem is to compile a consistent set of heuristics.
  - Granuality.
  - Errors and contradictions.
    - Input error.
    - Cognition error.
    - Approximation and conversion error.

- Questions:
  - How to compile a consistent set of heuristics?
  - How can you choose the “right” one when you have two different selections?
  - How can you easily detect a potential conflict?
    - It is difficult to be 100% sure that there is no conflict.
Comments

- Numerical scale.

- We do not assume every endgame has a fixed value by simply considering its material combination.
  - Many critical endgames have different values according to their positions.

- It is an art to integrate the values from material combinations into the evaluating function.
Books: about 10,000 combinations

- 象棋實用殘局
- 新殘棋例典1, 2, 3, 4, 5, 6
- 象棋基本實用殘局詳解
- 圖說象棋殘局
- 象棋殘局基礎
- 巧勢殘局
- 馬兵專集
- 馬兵專集增補
- 炮兵專集
- ...

Sources (I)
Sources (II)

- Computer constructed endgames: about 2,500.
- Endgames input by a human expert: about 17,000.
  - Using a web interface to manually input results of endgames with very few total number of attacking pieces.
- Using expert systems and rules to do inference: about 110,000.
  - Differ from collected endgames by one piece after removing some meaningless ones.
- Total: 140,320 out of the possible 2,125,764 feasible combinations.
Problems

- **Human mistakes.**
  * Different conclusions from different sources, e.g., books.
    - Different conclusions were made in different eras.
    - Different conclusions were made by different authors.
    - Some books discuss an endgame extensively with detailed positions, but have no general conclusions.

- **Algorithmic mistakes.**
  * Our algorithm for computer inferred endgame values has a roughly 90% of correctness.

- **Granularity.**
  * Some books only record results using a win-loss-draw format, not in 10 levels as we do.
  * Perfect endgame databases obtained by retrograde analysis contain winning rates, not a 12-level value.
    - How to convert rates to levels?
How to detect conflicts – Basics

- Piece additive rule:
  - The result of an endgame cannot get worse by
    - gaining extra pieces on your side;
    - losing pieces on your opponent’s side.
  - The result of an endgame cannot get better
    - by losing pieces on your side;
    - if your opponent gains piece.

- Rule of defensive pieces, i.e., Elephant and Guard.
  - The result of an endgame cannot normally be greatly changed by gaining/losing an extra defensive piece.

- Rule of draw and tie:
  - It is a draw if no side can win.
  - It is a tie if either side can win.
  - An endgame cannot usually be turned from tie into draw by using the piece additive rule.
How to detect conflicts – Process

- **Procedure:** check rules for endgames that we have already collected.
  - Piece additive rule.
  - Rule of defensive pieces, i.e., Elephant and Guard.
  - Rule of draw and tie.

- **Using relations between endgames, not just endgames themselves** to check for potential conflicts.
  - Similar activities applied for human cognitive process.
A graphic view

- A graph theoretical model.
  - vertex: an endgame
  - edge: between two vertices \( u \) and \( v \) if they follow the piece additive rule.

  - the direction from \( u \) to \( v \) if the value of \( u \) must be no worse than that of \( v \).

![Diagram of a graph theoretical model with vertices and edges indicating advantage, tie, slight advantage, and great advantage to red or black.](image-url)
High level ideas

- Assume the major part of the heuristics are correct.
- A **conflict** is an edge such that the values between them does not follow the piece additive rule.

\[ \geq \]

- The vertex who has a large percentage of conflicts is more likely to be incorrect.
Enhanced ideas

- **A potential conflict** is an edge such that the difference in values between the endpoints is more than a threshold, say 3, and the two connected endgames follow the rule of the defensive pieces.

- Original relation.

- Enhanced relation.

```
>=

<= +threshold
```
Algorithm

- For each endgame, compute the percentage of conflicts, which is the ratio between the number of “corrected” relations and the number of total relations.
- Identify the ones with the large percentage of conflicts and either use human or an automatic procedure to re-assign its value.
Potential problems for our approach

- A cluster of endgames all with consistent errors.
- A sparse or isolated cluster where inter-relation is few.
- A vertex can have a wide range of possible values due to the fact the values of its neighbors are much higher or lower than it.
- Solution: randomly select endgames in different clusters and verify them by human experts.
Remarks

- Out of about 140,320 endgames, there are about 1/12 severe errors (ones whose corrected values differ from the original values by at least 3).
- A total of 1/3 endgames are revised.
- A period of 1 year is spent to obtain a consistent set of heuristics where it is almost impossible to manually check the consistency of the collection of endgames previously.
Ongoing work

- More testing and analysis are needed.
- Using graph theoretical techniques to further process the data.
  - Assign a different weight to a different type of relations, and use the weight to find the ones that are most likely to be incorrect.
  - More inferencing rules that are not just between direct neighbors.
    - Piece exchanges: you cannot get better by exchanging a stronger piece with a weaker piece.
    - Depending on the piece involved, assign a confidence factor, e.g., adding a rook and adding a knight have different levels of confidence.
- Use expert system to do a better job in self-correcting.
- Test how much it can improve the performance of a Chinese chess program.
- Further usage:
  - Tutoring
  - E-learning
  - Knowledge abstraction
Rules for repetitions
Chinese Chess Special Rules (1/2)

- A player cannot avoid the losing of the game or important pieces by forcing the opponent to do repeated counter-moves.
  - Checking the opponent’s king repetitively without chance of checkmate.
    - Asia rule example #2.
  - Chasing an unprotected opponent’s piece repetitively without chance of capturing it.
    - Asia rule example #19.
  - Threatening (to checkmate) repetitively without chance of realizing the threat.
    - Asia rule example #31.

- Not a problem for Western chess.
  - Cycles mean draw.
  - It is difficult to force a repetition when one side is not willing to do it.

  - The king cannot move out of the palace.
Chinese Chess Special Rules (2/2)

- You can always “chase” a protected opponent piece because the opponent can decide to exchange it with your attacking piece.
- Sometimes it is difficult to check whether a piece is truly or falsely protected.
  - Asia rule example #39.
  - Asia rule example #105.
Asia Rule Example #2

- Checking the opponent’s king repetitively with no hope of checkmate.
  - $R4=5, K5=6, R5=4, K6=5,...$
  - Red Rook checks Black King.
Asia Rule Example #19

- Chasing an unprotected opponent’s piece repetitively with no hope of capturing it.
  - $C_2 - 1, R_4 - 2, C_2 + 2, R_4 + 2, \ldots$
  - Red Cannon at the 2nd column chases Black Rook.
Asia Rule Example #31

- Threatening (to checkmate) repetitively with no hope of realizing the threat.
  - $R2=1, C9=8, R1=2, C8=9,...$
  - Black Cannon at the 9th column threatens to checkmate.
Asia Rule Example #39

- Sometimes it is difficult to check whether a piece is *truly* or *falsely* protected: the definition of a protector is complicated.
  - \( R8+2, G6+5, R8-3, G5-6, \ldots \)
  - *Red Knight at the 2nd column is not protected.*
  - *Black Rook at the 6th column cannot threaten.*
Sometimes it is difficult to check whether a piece is truly or falsely protected: you can block a protector.

- $P7=6, M1+3, P6=7, M3-1,...$

- The protector of Black Knight at the 7th column is blocked.
Types of rules

- **Two main categories:**
  - **Asian version (2003)**
    - Supported by Asian Chinese Chess Association.
    - Simple and effective.
    - Is not really “fair” in certain complex cases.
    - Taiwan version (2007) is based on Asian version.
  - **PRC version (1999)**
    - Supported by the PRC Chinese Chess Association.
    - A national standard.
    - Try to be as complete and ”fair” as possible.

- **Problems in computer implementation:**
  - “Rules” are vague.
  - Often illustrated with examples.
Rules: Taiwan Version

Rules: Asian Version

Rules: PRC Version

Rules: Problems About the PRC Version

Current solutions

- Current treatment of special rules:
  - Avoid them at all: do not play repeated positions.
    - May lose advantage.
    - Must allow loops in endgame construction.
  - Special cases:
    - Only one side has attacking pieces: all are implemented.
    - One side has only a pawn and some defending pieces: can be affected by special rules.
  - Partial treatment:
    - Implement only the rules related to “checking.”
    - Implement some “chasing” rules.
    - Verify whether special rules can affect an endgame.

- We need a throughout understanding of special rules to build larger endgame databases.
Special Rules: Previous Results

- Partial treatment may build imperfect databases.
  - [Fang, Hsu & Hsu 2000].
  - Upto 17.3% for the checking rule in KRKNMM (⿰ vs. ♂ 象 象) [Fang, Hsu & Hsu 2002].
  - Jih-tung Pai [Private communication 2003] implemented a variation of [Fang, Hsu & Hsu 2002].

- Look for necessary conditions when databases can be stained by special rules.
  - Selected 50+ databases are verified [Fang 2004].
Special Rules: Work in Progress

- May affect the correctness of evaluation functions.
  - Xie Xie vs. Contemplation in the first WCCCC (Year 2004).
    ▶ Less than 3% of the games played.
  - About 5% of the games played in the 10th Computer Olympiad (October 2005) need to utilize special rules.

- Usage of logic and graph theory in an algorithmic context to describe the Asian version.
  - To explain all examples.
  - To abstract hidden experts’ knowledge.
  - To obtain fast computer implementations.

- Still a long way to go for the PRC version.
Xie Xie vs. Contemplation at WCCCC 2004

- Red: Contemplation.
- N3+4,R7–6,N4–3,R6–7,...
  - Red Knight at 3rd column is protected.
  - The game ended in a draw.
any pieces in between; otherwise it is FALSE.

- $is\_stallmate(P)$ (因鈴) = ($move\_gen(P) = ∅$).

7. Let $R = (P_1, P_2)$ be a ply. Let $P_1 = (B_1, player)$.

- $is\_suicide(R)$
  (自殺步：走之前沒有因鈴、王見王或被將，但走之後對手可以一步之內使你因鈴、王見王或被將，因此必輸)
  = ($\neg is\_stallmate(P_1) \land \exists m \in move\_gen(P_2), is\_stallmate((P_2, m))$) ∨
    ($\neg is\_KFK(P_1) \land is\_KFK(P_2)$) ∨
    ($\exists a$ piece $p$, $p \notin pieces\_attacking\_king((B_1, \neg player)) \land p \in pieces\_attacking\_king(P_2)$).

8. Let $R = (P_1 = (B_1, player), P_2 = (B_2, \neg player))$ be a ply. Let $p$ and $kind(q) \neq KING$ be two pieces in $B_1$ where $capturing\_move(P_1, p, q) \in move\_gen(P_1)$.

- $is\_forced\_capture(R)$
  (將軍抽子：走之前沒有將到對方的王，但走子此步之後將軍)
  = ($pieces\_attacking\_king(P_1) = ∅$) ∧ ($is\_in\_check(P_2)$).

9. Let $B_1$ be a board configuration. Let $p$, $q$ and $r$ be three pieces in $B_1$,
    $capturing\_move(P_1, p, q) \in move\_gen(P_1)$, and $capturing\_move(P_2, r, p) \in move\_gen(P_2)$,
    where $owner(p) = player$, $P_1 = (B_1, player)$, and $P_2 = position\_after(capturing\_move(P_1, p, q))$. Let $P_2 = (B_2, \neg player)$ and let $P_3 = position\_after(capturing\_move(P_2, r, p))$.

- $can\_be\_protected(B_1, p, q, r)$
  (在$p$可以吃到$q$的盤面組合$B_1$中，若$p$吃$q$，則$r$可以自由離線回吃$p$，目前自由離線的定義為：不能是自殺步。因此，若對手是將軍抽子，代表是可以自由離線。)
Construction of huge endgame databases
Chinese chess endgame database:

- Indexed by a sublist of pieces $S$, including both Kings.

A position in a database $S$: A legal arrangement of pieces in $S$ on the board and an indication of who the next player is.

Perfect information of a position:

- What is the best possible outcome, i.e. win/loss/draw, that the player can achieve starting from this position?
- What is a strategy to achieve the best possible outcome?

Given $S$, to be able to give the perfect information of all legal positions formed by placing pieces in $S$ on the board.

Partial information of a position:

- win/loss/draw; DTC; DTZ; DTR.
Usage of Endgame Databases

- Improve the “skill” of Chinese chess computer programs.
  - KNPKGGMM ( 马 兵 vs. 士 士 象 象 )
- Educational:
  - Teach people to master endgames.
- Recreational.
An Endgame Book
Books
Definitions

- **State graph for an endgame** \( H \):
  - **Vertex**: each legal placement of pieces in \( H \) and the indication of who the current player (Red/Black) is.
    - Each vertex is called a position.
    - May want to remove symmetry positions.
  - **Edge**: directed, from a position \( x \) to a position \( y \) if \( x \) can reach \( y \) in one ply.
  - **Characteristics**:
    - Bipartite.
    - Huge number of vertices and edges for non-trivial endgames.
    - Example: KCPGGMMKGGMM has \( 1.5 \times 10^{10} \) positions and about \( 3.2 \times 10^{11} \) edges.
Overview of Algorithms

- **Forward searching:** doesn’t work for non-trivial endgames.
  - AND-OR game tree search.
  - Need to search to the terminal positions to reach a conclusion.
  - Runs in exponential time not to mention the amount of main memory.
  - Heuristics: $A^*$, transposition table, move ordering, iterative deepening...

---

TCG: Computer Chinese Chess, 20131226, Tsan-sheng Hsu ©
Retrograde Analysis (1/2)

- First systematic study by Ken Thompson in 1986 for Western chess.
  - Retrograde analysis (回溯分析)

- Algorithm:
  - List all positions.
  - Find all positions that are initially “stable”, i.e., solved.
  - Propagate the values of stable positions backward to the positions that can reach the stable positions in one ply.
    - Watch out the and-or rules.
  - Repeat this process until no more changes is found.
Critical issues: time and space trade off.

- Information stored in each vertex can be compressed.
- Store only vertices, generate the edges on demand.
- Try not to propagate the same information.
Stable Positions

- Another critical issue: how to find stable positions?
  - Checkmate, stalemate, King facing King.
  - It maybe the case the best move is to capture an opponent’s piece and then win.
    - so called “distance-to-capture” \((DTC)\); 
    - the traditional metric is “distance-to-mate” \((DTM)\).

- Need to access values of positions in other endgames. For example,
  - \(KCPKGGM\) needs to access 
    - \(KCKGGMM\)
    - \(KPKGGMM\)
    - \(KCPKGMM, KCPKGGM\)
  - A lattice structure for endgame accesses.
  - Need to access lots of huge databases at the same time.

- [Hsu & Liu, 2002] uses a simple graph partitioning scheme to solve this problem with good practical results.
An Example of the Lattice Structure
Yet another critical issue: cycles in the state graph.

- Can never be stable.
- In terms of graph theory,
  - a stable position is a pendant in the current state graph;
  - a propagated position is removed from the state graph;
  - no vertex in a cycle can be a pendant.
Cycles in the State Graph (2/2)

- For most games, a cyclic sequence of moves means draw.
  - Positions in cycles are stable.
  - Only need to propagate positions in cycles once.
- For Chinese chess, a cyclic sequence of moves can mean win/loss/draw.
  - Special cases: only one side has attacking pieces.
    - Threaten the opponent and fall into a repeated sequence is illegal.
    - You can threaten the opponent only if you have attacking pieces.
    - The stronger side does not need to threaten an opponent without attacking pieces.
    - All positions in cycles are draws.
  - General cases: very complicated.
Western chess: general approach.
- Complete 3- to 5-piece, pawn-less 6-piece endgames are built.
- Selected 6-piece endgames, e.g., KQQKQP.
  - Roughly $7.75 \times 10^9$ positions per endgame.
  - Perfect information.
  - $1.5 - 3 \times 10^{12}$ bytes for all 3- to 6-piece endgames.

Awari: machine and game dependent approach.
- Solved in the year 2002.
- $2.04 \times 10^{11}$ positions in an endgame.
  - Using parallel machines.
  - Win/loss/draw.

Checkers: game dependent approach.
- $1.7 \times 10^{11}$ positions in an endgame.
  - Currently the largest endgame database of any games using a sequential machine.
  - Win/loss/draw.
  - Solved in the year 2007 with a total endgame size of $3.9 \times 10^{13}$.

Many other games.
Results — Chinese Chess

- Earlier work by Prof. S. C. Hsu (許舜欽) and his students, and some other researchers in Taiwan.
  - KRKGGMM (陣 vs. 士 士 象 象) [Fang 1997; master thesis]
    - About $4 \times 10^6$ positions; Perfect information.
  - Memory-efficient implementation: general approach.
  - KCPGMKGGMM (炮 兵 仕 仕 相 相 vs. 士 士 象 象) [Wu & Beal 2001]
    - About $2 \times 10^9$ positions; Perfect information.
  - KCPGGMMKGGMM (炮 兵 兵 仕 仕 相 相 相 vs. 士 士 象 象 象) [Wu, Liu & Hsu 2006]
    - About $8.8 \times 10^9$ positions; $2.6 \times 10^{-5}$ seconds per position; Perfect information.
    - The largest single endgame database and the largest collection reported.
  - Verification [Hsu & Liu 2002]
  - Special rules: more likely to be affected when endgames get larger.
Problems and Solutions

- Need to solve the problem of finding cycles in a graph.
  - Modeling using graph theory.
  - Using previous knowledge from graph theory.

- Need to solve the problem of requiring a huge space to store the database being constructed.
  - Carefully partition the database into disjoint portions so that only the needed parts are loaded into the memory.
  - Using combinatorial properties to do the partition.
Benchmarking
Introduction

- It is inevitable to revise your Chinese chess program during development.
- There are many stable versions, $P_{t_1}, P_{t_2}, \ldots$, indexed by time stamp $t_i$.
  - Stable means bug-free, namely the code does exactly what you intend to do as far as you know.
- Question:
  - How can you be sure that $P_{t_i}$ is better than one of its previous versions $P_{t_j}$ for some $j < i$?
  - Notation:
    - $P_i > P_j$ means $P_i$ is better than $P_j$.
    - $P_i = P_j$ means $P_i$ is about the same as $P_j$.
    - $P_i \geq P_j$ means $P_i$ is no worse than $P_j$.
- Easy solution:
  - Conduct some games between the two versions and see who wins the most games.
- Is the solution satisfactory?
Potential problems

- **Is the relation transitive?**
  - That is, if $P_i > P_j$ and $P_j > P_k$, then can you be sure that $P_i > P_k$?

- **Is the relation platform independent?**
  - CPU speed, memory bandwidth, …
  - Both versions use the same platform.
  - Both versions use different platforms.
  - Platform may differ over time.

- **Is the relation rule independent?**
  - Fast game (5 seconds per ply), Normal game (60 seconds per ply), …
  - Chinese chess rules for repetitions.

- **Without knowing answers to these potential problems, we do not know which version is ”the” best.**
Motivation and related work

Scenario:
- Suppose you have a “great idea” $M$ and a current version $P_i$.
- By patching $P_i$ with $M$, you obtain the next version $P_{i+1}$.
  $\triangleright$ Let $P_{i+1} = P_i + M$.

Questions:
- Is $M$ really a “great idea”?
- What version shall I use during a competition?
- How to compare the “strength” of $P_i$ and $P_{i+1}$?
Related work

- The **Swiss system for a tournament.**
- Rating systems, such as ELO, over a period of time.
  - *It has been estimated by Levy and Newborn that doubling the computer speed gains approximately fifty to seventy ELO points in playing strength for chess.*
  - *However, this applies mainly to computer-vs-computer matches, and not to computer-vs-human matches.*
  - *Remark: cited from Wiki.*

...
Our little experiment

- Two versions of Contemplation.
  - Version 814:
    ▶ *The version we used for Computer Olympiad 11 (May 2006).*
    ▶ *Searching speed: 5.5 M nodes per second during middle game.*
  - Version 903:
    ▶ *The latest version (upto May 2007).*
    ▶ *Adding 6000+ lines of code (for the evaluation function) to version 814.*
    ▶ *Searching speed: 2.8 M nodes per second during middle game.*

- Version 814 is faster, but version 903 has more domain knowledge.

- Platform:
  - Free BSD 6.2 + GCC 4.1.2
  - Xeon 5160 3.0 GHZ CPU + 4GB DDR II 677 memory
    ▶ *This CPU has two cores.*
    ▶ *Each version uses one core.*
Testing setup

- **Games played:**
  - 22 fixed openings.
  - No endgame databases.
  - Testing of middle games.
  - Each program plays first alternatively for each opening.
  - A total of 44 games played in each round.

- **Each round:**
  - With pondering.
  - Fix the time used in each ply.
    - From 5 seconds, 10 seconds, ..., 60 seconds.
  - Time that is not used in a ply can be saved to be used in the plys thereafter.

- A total of 12 rounds, i.e., 528 games, were played in four weeks during May 2007.
### Results

<table>
<thead>
<tr>
<th>seconds</th>
<th>v903 vs v814</th>
<th>scores</th>
<th>difference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>win</td>
<td>draw</td>
<td>loss</td>
</tr>
<tr>
<td>5</td>
<td>19</td>
<td>7</td>
<td>18</td>
</tr>
<tr>
<td>10</td>
<td>19</td>
<td>7</td>
<td>18</td>
</tr>
<tr>
<td>15</td>
<td>15</td>
<td>13</td>
<td>16</td>
</tr>
<tr>
<td>20</td>
<td>16</td>
<td>8</td>
<td>20</td>
</tr>
<tr>
<td>25</td>
<td>21</td>
<td>10</td>
<td>13</td>
</tr>
<tr>
<td>30</td>
<td>21</td>
<td>9</td>
<td>14</td>
</tr>
<tr>
<td>35</td>
<td>21</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>7</td>
<td>17</td>
</tr>
<tr>
<td>45</td>
<td>19</td>
<td>8</td>
<td>17</td>
</tr>
<tr>
<td>50</td>
<td>20</td>
<td>11</td>
<td>13</td>
</tr>
<tr>
<td>55</td>
<td>18</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>60</td>
<td>22</td>
<td>12</td>
<td>10</td>
</tr>
</tbody>
</table>

- **Each win:** 2 points.
- **Each draw:** 1 point.
- **Each loss:** 0 point.
Observation 1

- Observation I:
  - v814 roughly equals to v903 at rounds that took time less than 20 seconds.
  - v814 beats v903 at the 15-second and the 20-second rounds, but loses all other rounds.

- Comments:
  - When time is limited, searching deeper and searching smarter is about the same for Contemplation.
  - 20-second round is the best “competing environment” for v814 as compared to v903.

- Conjecture:
  - Expect a similar behavior on a 40-second round for a platform that is twice slower.
Observation II

- Observation II:
  - The performance of v814 improves on rounds of 40 and 45 seconds.

- Comments:
  - The performance curse of v903 vs. v814 is stair-cased.
  - The average branching factor $B_{avg}(v814)$ is about 2 to 3.
    - Need 2 to 3 more times of running time in order for the search depth to increase 1.
  - It is only useful to search the entire layer of nodes of the same depth.
Observation II – conjecture

Conjecture:

- Relatively speaking, the performance of v903 is more stable.
- If the additional computational power is less than $B_{avg}$, then it is better to spend the extra power on the evaluation function.
- Extra power may come from
  - multi-threading;
  - faster architecture, CPU or RAM;
  - . . .
Observation III

- **Observation III:**
  - v903 out-scores v814 at a large margin for rounds of 25, 30, 35, 50, 55 and 60 seconds.

- **Comments:**
  - Searching depth may not be a dominating factor for playing strange when the depth is “enough.”
  - The law of diminishing returns (效益遞減) in searching.

- **Conjecture:**
  - Need to balance the time used in searching deeper and the time used in searching smarter.
Remarks

- Need to fix the “competing environment” \( E \) to define \( P_{i+1} > P_i \).
  - Revised definition: \( P_{i+1} >_E P_i \).

- Whether a newly coded \( M \) is a great idea or not depends on the platform and “competing environment” you are using.
  - If \( M \) slows you down too much at the current platform, then your testing score may be worse.
  - Some times later, say next year, if a faster platform is available, then your testing score may become better.
  - You may observe a zig-zag effect on a series of versions.
    - It becomes bad for a while, then it becomes good again.

- Do not abandon \( M \) easily!

- You are a winner of a competition does not guarantee you will be the winner later even if all programs stay unchanged.
  - Faster platforms may be invented.
  - Rules may change.
  - Luck is always part of a game with fun, such as Chinese chess.
Future work

- More testing and analysis are needed.
  - The growth of games win/draw/loss.
  - The behavior when one plays first/last.
  - The length, time elapsed or plys played, of games.
  - What kinds of games are easily win by a particular version?
  - Are the results stable?
  - …

- Compare results across different platforms.
  - For example, AMD v.s. INTEL.

- With opening databases.
- With endgame databases.
Concluding Remarks

- Many open problems.
- Research opportunities:
  - Algorithm and complexity.
  - Algorithmic engineering.
  - External memory algorithms.
  - System implementation.
  - Parallel computing.
  - A.I.
    - Knowledge extracting.
    - Data mining.
    - . . .
  - Discrete Math., e.g., Graph theory.
- Commercial opportunities.
- Fun.
References and further readings (1/3)


References and further readings (3/3)
