Chess Evolution Visualization

W.-L. Lu et al.
TVCG 2014

Sang-Hyeok Lee
2014. 05. 09
Computer Graphics @ Korea University
Index

1. Introduction
2. Related Work
3. Design Methodology
4. Result and Discussions
5. Evaluation
6. Conclusions
1. Introduction
Chess Visualization?

- Basic concepts
  - Chess AI (Deep Blue)
  - Chess Tool (Analysis & Visualization)
  - Algebraic Chess Notation

Arena Chess GUI

Algebraic Chess Notation
1. Introduction

Feature of our System

- 기존 Tool의 개선
  - 체스 시각화 방법론 재고
    - Global-to-local visualization
  - 새로운 시각화 기법 적용
    - Decision Tree
    - Score graph
- 가시성 향상을 위한 Graph 정제 기법 적용
  - Merging nodes
  - Aligning nodes
  - Highlighting nodes
  - Edge thickness

- 효과 – 다음 질문의 답을 쉽게 얻을 수 있다.
  - 내가 무엇을 실수하였나?
  - 현재 할 수 있는 최선의 수가 무엇인가?
  - 게임의 판도가 어떤식으로 변하는가?
  - 상대는 왜 항복을 하였는가?
2. Related Work

Chess AI

- Chess AI
  - Board evaluation
    - Evaluate how good a position is
    - piece location, tactics, king safety, board control, and passed pawns
  - Game tree pruning
    - Determine the best move by searching the minimal tree
  - In this paper, Chess AI is not main Issue.

- Stockfish
  - World Rank 2 Chess AI (of 272 AIs)
  - Open source chess engine
  - Windows, Mac OS X, or Linux, or on iPhone, iPad, or Android device

<table>
<thead>
<tr>
<th>Rank</th>
<th>Name</th>
<th>Rating</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Houdini 4 64-bit 4CPU</td>
<td>3274</td>
<td>+18</td>
</tr>
<tr>
<td>2</td>
<td>Stockfish DD 64-bit 4CPU</td>
<td>3249</td>
<td>+18</td>
</tr>
<tr>
<td>3</td>
<td>Komodo TCEC 64-bit 4CPU</td>
<td>3235</td>
<td>+19</td>
</tr>
<tr>
<td>4</td>
<td>Guli 2.8b 64-bit 4CPU</td>
<td>3201</td>
<td>+29</td>
</tr>
<tr>
<td>5</td>
<td>Critter 1.6a 64-bit 4CPU</td>
<td>3174</td>
<td>+11</td>
</tr>
</tbody>
</table>

http://www.computerchess.org.uk/
2. Related Work
Chess visualization & Tree and Graph visualization

• Chess visualization
  ▪ All current chess visualizations
    • Focus on an **instant**
      ▪ even though the AI engines contain long-term tactical knowledge
    • Step-by-step piece movement guide
    • AI thought → invisible

• Tree and Graph visualization
  ▪ Represent relations between entities
  ▪ Main Issue
    • Edge
    • Layout
    • Navigation
    • Clustering
2. Related Work

**Time-varying data Visualization**

- Time-varying data visualization
  - Reveal temporal trends of the underlying data

![Coherent Time-Varying Graph Drawing with Multifocus+Context Interaction](image)

K.-C. Feng *et al.* [TVCG 2012]

- 체스게임의 decision tree 에는 적용하기 힘들다
  - 기존 : 미세한 시간으로 연속되는 일련의 다양성의 정보
  - 본 논문 : Decision tree 라는 특정 정보

- 본 논문과 관련이 적다
2. Related Work

Storyline Visualization

- Storyline visualization
  - Chess visualization 에 가장 합당한 시각화 기법

- But, 기존 연구들에 비해 더 복잡한 데이터들을 시각화할 필요
  - AI가 만드는 수많은 잠재적인 경우의 수
  - 선수가 만드는 수에 따라 실시간으로 바뀌는 상황
  - AI가 만드는 전략과 전술의 효과적인 시각화가 가장 큰 이슈

Tracing Genealogical Data with Timenets
N.W. Kim et al. [AVI 2010]
3. Design Methodology

Rough Sketch

- **Goal**
  - Showing *potential* chess position
  - Highlight *event*
  - *Global to Local* visualization

- **Flow**
  
  Chess sequence
  
  → AI Engine (recommendation : 4~9) **WHY?**
  
  → Decision tree (raw data)
  
  → Branch pruning (depth : 20) **WHY?**
  
  → Several refinements... (이후 슬라이드에서 설명)

  → **Evolution Graph**
3. Design Methodology

Evolution Graph Generation

1. 중복된 노드 통합
   [merging nodes]
   → 여기서 중복이라 함은, 완전히
     똑같은 배치의 체스판 상황을
     말함
   → 체스전용 Hashing 기법 적용
     → A. Cozzie and R.M. Hyatt, “The
       Effect of Hash Signature
       Collisions in a Chess Program”
       [ICGA J. 2005]
3. Design Methodology
Evolution Graph Generation (cont.)

2. 의미가 적은 노드 통합
   [shorten the branches]

   → 의미가 적은 노드: 다음 수로 가는 방법이 한가지 뿐인 노드, 즉 부모와 자식을 하나씩 가진 노드

노드의 배치 자체도 달라진 이유?
   → 다음페이지 설명
3. Design Methodology

Evolution Graph Generation (cont.)

3. Edge Crossing 최소화, Node Location 결정
   - GraphVis 툴 이용 \( \rightarrow \) Objective function 을 정의해주면 에너지를 최소화 하면서 자동으로 배치해준다

\[
\sum_{\{i,j\} \in E} k_{ij} \omega_{ij} |v_i - v_j|, \quad \text{subject to}
\]

\[
|v_a - v_b| \geq \frac{s_a + s_b}{2} + d, \quad \text{where}
\]

\[
k_{ij} = \begin{cases} 
1, & \text{if both } i \text{ and } j \text{ are virtual nodes,} \\
2, & \text{if one of } i \text{ and } j \text{ is a virtual node,} \\
8, & \text{if both } i \text{ and } j \text{ are real nodes,}
\end{cases}
\]

\[G = \{V, E\} \quad v_i = \{x_i, y_i\} \in V \quad \{i, j\} \in E\]

\[f_h(v) \quad \text{The order node of each node in the horizontal direction}
\]

\[f_v(v) \quad \text{The order node of each node in the vertical direction}\]
3. Design Methodology

Evolution Graph Generation (cont.)

3. Edge Crossing 최소화, Node Location 결정

1. \[ \sum_{\{i,j\} \in E} k_{ij} \omega_{ij} |v_i - v_j|, \quad \text{subject to} \]

2. \[ |v_a - v_b| \geq \frac{s_a + s_b}{2} + d, \quad \text{where} \]

3. \[ k_{ij} = \begin{cases} 
1, & \text{if both } i \text{ and } j \text{ are virtual nodes,} \\
2, & \text{if one of } i \text{ and } j \text{ is a virtual node,} \\
8, & \text{if both } i \text{ and } j \text{ are real nodes,} 
\end{cases} \]

3. Virtual node definition

연결된 노드 \( v_i \)와 \( v_j \) 가, \( f_h(v) \) 값으로는 근접해있지 않는 경우, 즉 \(|f_h(v_i) - f_h(v_j)| > 1\)
일 경우, \( v_i \)와 \( v_j \) 사이에 Virtual node라고 하는 노드를 삽입해 준다. 그 후, \( v_i \)와 \( v_j \)를 잇는 edge를, Virtual node들을 Control point로 갖는 Bezier curve 형태의 edge로 이어준다.
3. Design Methodology

Evolution Graph Generation (cont.)

3. Edge Crossing 최소화, Node Location 결정

\[ \sum_{\{i,j\} \in E} k_{ij} \omega_{ij} |v_i - v_j|, \quad \text{subject to} \]

\[ |v_i - v_j| \geq \frac{s_a + s_b}{2} + d, \quad \text{where} \]

\[ k_{ij} = \begin{cases} 
1, & \text{if both } i \text{ and } j \text{ are virtual nodes}, \\
2, & \text{if one of } i \text{ and } j \text{ is a virtual node}, \\
8, & \text{if both } i \text{ and } j \text{ are real nodes},
\end{cases} \]

\[ v_i \& v_j \rightarrow \text{선수가 실제로 생성한 노드} \rightarrow \]

\[ v_a \& v_b \rightarrow \text{같은 열에 인접한 노드} \rightarrow \]

\[ s_a \& s_b \rightarrow a, b \text{ 노드의 반지름} \]

\[ d \rightarrow \text{노드들의 여백 조정 값 (본 논문에서는 0.5)} \]
3. Design Methodology

Evolution Graph Rendering

• Rendering factors
  ▪ background
  ▪ node shape
  ▪ edge shape
  ▪ edge thickness
3. Design Methodology

Evolution Graph Rendering (cont.)

- background
  - darkseagreen: harmonic viewing experience, enhance the contrast in the evolution graph

- node shape

<table>
<thead>
<tr>
<th>Color</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>⬤ black</td>
<td>The position is in draw.</td>
</tr>
<tr>
<td>⬤ white</td>
<td>White gives a check to Black.</td>
</tr>
<tr>
<td>⬤ black</td>
<td>Black gives a check to White.</td>
</tr>
<tr>
<td>⬤ white</td>
<td>White gives a checkmate to Black.</td>
</tr>
<tr>
<td>⬤ black</td>
<td>Black gives a checkmate to White.</td>
</tr>
<tr>
<td>⬤ white</td>
<td>The turn for White to control armies.</td>
</tr>
<tr>
<td>⬤ black</td>
<td>The turn for Black to control armies.</td>
</tr>
</tbody>
</table>

- edge shape

<table>
<thead>
<tr>
<th>Shape</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>⬤</td>
<td>One move to the next position.</td>
</tr>
<tr>
<td>⬤ ⬤</td>
<td>Several moves to the next position.</td>
</tr>
</tbody>
</table>

- edge thickness

\[ w_{ij} = \begin{cases} 
(\log(\delta_{ij} - \min_i))^2, & \text{if } \delta_{ij} > \min_i, \\
\alpha, & \text{if } \delta_{ij} = \min_i, 
\end{cases} \]

\[ -30,000 \leq w_{ij} \leq 30,000 \rightarrow [1, 30] \]
3. Design Methodology

Score Chart

- Evolution graph 이외에도, 추가적으로 Score chart 표현
  - Color: to indicate the side
  - X-axis: move number
  - Y-axis: the log value of the score by the chess AI
  - Circle: actual advantage (현재 가지고 있는 지형적 이점)
  - Band: potential advantage (앞으로의 수들이 가지고 있는 지형적 이점)
  - Translucency: to prevent occlusion

- Why Score Chart?
  - 두 선수간의 위치적 이점을 양적으로 표현하기 위해
  - 사용자는 이를 통해 경기 양상과 흐름을 파악할 수 있다
4. Result and Discussion

Rough Sketch

- Graph, Chart 외 추가적인 기능 구현
  - Global to local viewpoint 제공
    - node 클릭 시, 이후 가능한 수와 Checkmate 로 갈 수 있는 방법 표시
  - Dotted edge 변환 제공
    - 기존에 통합시켰던 노드를 클릭 시, 통합되어 생략되었던 노드들 표시
  - Chess board examination 제공
    - 체스 보드판을 추가적으로 제공
4. Result and Discussions

Case 1: Moderate Player vs Moderate Player

- James Plaskett(W) vs Sergei Shipov(B) [World Open U2200]
4. Result and Discussions

Case 2: Expert vs Expert

- Jose Raul Capablanca(W) vs Alexander Alekhine(B)  [Buenos Aires WCh]
4. Result and Discussions

Case 3: Deep Blue vs Kasparov

• Deep Blue(W) vs Garry Kasparov(B) [Second Match]
4. Result and Discussions

Case 4: AI vs AI

- Stockfish AI Engine(W) vs AnMon AI Engine(B)
4. Result and Discussions

Case 5: Chess Puzzle

- White로 플레이, 6번의 움직임으로 Black을 체크메이크하라.
4. Result and Discussions

Limitations

1. Evolution Graph 와 Score Chart 만으로는 부족하다.
   ▪ 두 시각화에선 전략, 전술적 정보가 표현되지 않기 때문이다.
   ▪ 불가피하게 기존의 체스판 GUI를 가져와야 했다.
   ▪ 기존의 체스판 GUI를 대체하는 새로운 기법이 필요.

2. AI engine이 만들어내는 Chess position 들은 precomputing 되어야 한다.
   ▪ 초기 프레임워크 기획상 불가피한 상황
   ▪ 이미 존재하는 체스 기보만을 분석 가능
   ▪ 사용자의 Query 입력을 받아서 동적으로 업데이트하는 방향으로 수정해나갈 계획

3. [개인적인 생각] 단순히 추천만을 해주고, 그 이유는 나타나있지 않다.
   ▪ 한 수, 한 수에 숨겨져 있는 의미를 표현해주면 교육적인 효과가 더욱 커질 것 같다. (예. 이 수는 ‘폰’을 희생하지만 ‘나이트’를 지키면서 동시에 상태 ‘비숍’을 압박할 수 있다. 등의 설명)
5. Evaluations

Comparison to Visualization Design

- Visualization Designs 에 대한 자체적인 당위성 평가
  - Radial vs Axis-Parallel
    - **Axis-Parallel**
      - 체스의 진행은 일방통행이고 일련의 축적되는 과정
      - Radial 채택시, edge 들이 교차되어 가시성 떨어짐
  - Full vs Simplified
    - **Simplified**
      - 40개 이상의 수(手), 최대 8개의 child node → 전부표현하기 너무 복잡
      - 주요한 상황이 눈에 잘 띄기위해선 단순화 과정 불가피
  - Implicit vs Explicit
    - **Explicit**
      - 그래프 하나만으로 전체적인 상황을 파악해야함
      - node 와 edge 들이 전부 특정 정보를 포함해야 함
  - 2D vs 3D
    - **2D**
      - 자료의 겹침으로 인한 폐색(occlusion) 방지
      - 평평한 디스플레이 상에서 구현되므로 2D가 적절
5. Evaluations
Comparison to Current Chess Visualization

- 현재 대표적인 Chess Visualization tool인 ‘Arena’와 비교

<table>
<thead>
<tr>
<th></th>
<th>Local View</th>
<th>Global View</th>
<th>Potential Move</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arena</td>
<td>Chess board</td>
<td>Algebraic chess notation (only future)</td>
<td>Algebraic chess notation &amp; Chess board</td>
</tr>
<tr>
<td>Our System</td>
<td>Evolution graph &amp; Chess board</td>
<td>Evolution graph &amp; Score chart</td>
<td>Evolution graph</td>
</tr>
</tbody>
</table>

![Chess Visualization Comparison Table and Diagrams](image-url)
5. Evaluations

User Study

- 참가자 21명, 23세~32세, 67% 초보자
- 5분 사용설명, 5분 연습시간
- 총 8개의 문제
  - 전체적인 흐름 파악 문제
  - 세부적인 상황 파악 문제

Satisfaction surveys

<table>
<thead>
<tr>
<th>question</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>p-value (time)</td>
<td>1.3 ×10⁻³</td>
<td>1.7 ×10⁻³</td>
<td>4.7×10⁻⁷</td>
<td>1.0 ×10⁻³</td>
<td>4.1×10⁻³</td>
<td>7.1×10⁻²</td>
<td>2.9×10⁻⁵</td>
<td>6.0×10⁻⁶</td>
</tr>
<tr>
<td>p-value (correct rate)</td>
<td>7.7×10⁻²</td>
<td>4.6 ×10⁻⁵</td>
<td>3.8×10⁻³</td>
<td>3.5×10⁻⁶</td>
<td>1.2×10⁻²</td>
<td>6.4×10⁻³</td>
<td>1.5×10⁻²</td>
<td>1.8×10⁻²</td>
</tr>
</tbody>
</table>
6. Conclusions

• 정리 : Chess Evolution Visualization
  ▪ 전체적인 흐름과 세세한 흐름을 동시에 관측 가능
  ▪ 전문가 - 전체 흐름과 중요한 이벤트를 알 수 있게 해준다
  ▪ 초보자 - 상대의 움직임 예측, 체크 전략 공부

• 긍정적 : 수행한 여러 평가를 바탕으로 이 시스템의 효용성 입증됨

• 발전계획 : 전략과 전술을 모두 파악하려면, 아직까지 체스판 GUI가 필요
  ▪ 사용자의 시선이 분산되는 단점이 존재
  ▪ 전략 정보(Graph, Chart)와 전술 정보(Chess board)를 통합할 수 있는 seamless(이음매 없이 부드러운) 한 시스템으로 개선할 예정