Chess Evolution Visualization

Wei-Li Lu, Yu-Shuen Wang, and Wen-Chieh Lin

Abstract—We present a chess visualization to convey the changes in a game over successive generations. It contains a score chart, an evolution graph and a chess board, such that users can understand a game from global to local viewpoints. Unlike current graphical chess tools, which focus only on highlighting pieces that are under attack and require sequential investigation, our visualization shows potential outcomes after a piece is moved and indicates how much tactical advantage the player can have over the opponent. Users can first glance at the score chart to roughly obtain the growth and decline of advantages from both sides, and then examine the position relations and the piece placements, to know how the pieces are controlled and how the strategy works. To achieve this visualization, we compute the decision tree using artificial intelligence to analyze a game, in which each node represents a chess position and each edge connects two positions that are one-move different. We then merge nodes representing the same chess position, and shorten branches where nodes on them contain only two neighbors, in order to achieve readability. During the graph rendering, the nodes containing events such as draws, effective checks and checkmates, are highlighted because they show how a game is ended. As a result, our visualization helps players understand a chess game so that they can efficiently learn strategies and tactics. The presented results, evaluations, and the conducted user studies demonstrate the feasibility of our visualization design.

Index Terms—Chess visualization, graph

1 Introduction

Chess is one of the most popular two-player strategy board games in the world. Players usually polish their chess skills, including strategy and tactics, by reading experts' comments and play books. Given that computers have recently surpassed humans in playing chess, modern artificial intelligence (AI) has emerged as a helpful tool for analyzing the interaction of pieces and the evolution of a game. AI's thorough and deep searching enables the depiction of chess positions after more than ten moves. Therefore, it is capable of analyzing long-term positional advantages of a game. However, all the current chess tools [1], [2] assist users in analyzing games by visualizing spatial information. For the example shown in Figure 1, Arena indicates that the opposing black knight can be captured (background in green); the white pawn may be captured while it is protected by a castle (background in yellow); and the white knight will be captured if nothing is done to prevent it (background in red). The visualization also indicates the best two moves for White to win in the current position using arrows. Purple and light blue stand for the best and the second best moves, respectively. As for the game evolution, it simply describes a sequence of chess moves using algebraic chess notation [3]. Hence, users have to sequentially investigate the game and build the global strategies in mind to answer the questions such as "Why does the player give up the game?", "Is there a way to escape from king hunt?", and "How does the player turn defeat into victory?"

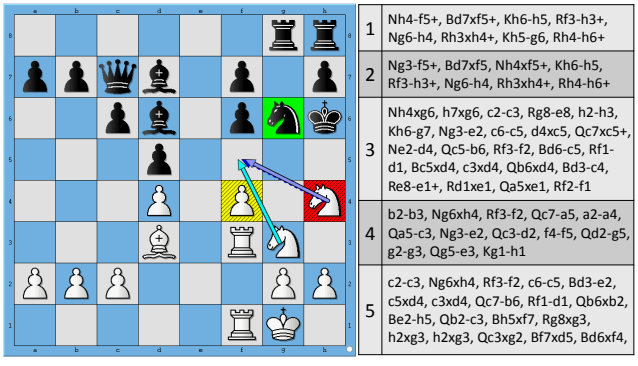

Fig. 1. (left) Current chess visualization tool (Arena [1]) focuses only on highlighting pieces that are under attack and step-by-step moves. (right) Long-term move sequences containing strategies can only be represented using algebraic chess notation [3], which is very difficult to understand.

We present an evolution visualization system to convey subtle trends of a chess game. Thus, instead of investigating chess positions sequentially, users are allowed to examine a chess game from global to local viewpoints. The analysis of a game is obtained from recording and linking successive chess positions made by players, followed by utilizing AI to grow a decision tree from this recorded structure. Essentially, a decision tree is a directed graph, where each node represents a chess position and edges are the legal plies between two positions. An intuitive way to visualize the chess evolution is to render the whole

• *W. L. Lu, Y. S. Wang, and W. C. Lin are with the Department of Computer Science, National Chiao Tung University, Taiwan*

E-mail: willy-78831@hotmail.com
E-mail: yushuen@cs.nctu.edu.tw
E-mail: wclin@cs.nctu.edu.tw

decision tree. Yet numerous nodes and complex edges can cause serious visual clutter. Furthermore, different nodes in the tree may represent the same chess position, thus resulting in perceptual misleading. These readability problems should be well handled in chess evolution visualization.

We conditionally simplify a game decision tree by merging nodes that represent the same chess position and then discarding nodes that have little correlation to the game evolution. The node merging reveals chess position relations that are invisible in a decision tree, while the node discarding achieves readability. Our system then renders the simplified graph in 2D space, where node locations are determined using GraphViz [4], and edge thicknesses are computed based on how good the chess move is. By highlighting the nodes containing critical events, such as draws, effective checks and checkmates, this evolution graph shows the circumstance of each chess position, including potential advantages and disadvantages after a number of moves. To convey the global trend throughout a game, we also align a score chart, which contains the current score and the potential score fluctuation with our evolution graph. This is done to quantize the two players' tactical advantages at each move. By integrating these two global visual clues, our evolution visualization reveals potential end positions of a game and answers why a player would consider giving up a game although there are still many pieces left on the board.

Our technique leverages AI to analyze a chess game and is capable of conveying the subtle trends to the users. This global-to-local visualization enables experts to quickly obtain the overview and critical events that lead to winning or losing a chess game. It also allows novice players to preview potential chess positions in future moves to know how an opponent may respond and learn how the strategy works. To evaluate our technique, we compared our chess evolution graph to professional comments, existing graph visualization designs, and current chess tools. We also conducted a user study with 21 participants, including novice players, experienced players, and an expert. Experimental results show that our visualization fits the tactical points raised by experts and is capable of helping players understand chess games efficiently.

2 Related Work

Chess AI. AI long has been developed to play chess. Existing techniques focus on board evaluation and game tree pruning. The former evaluates how good a position is based on piece locations, tactics, king safety, board control, and passed pawns, etc. [5]. The latter strives to determine the best move by searching the minimal tree because exploring all possible moves is impractical. Therefore, alpha beta algorithm [6], [7], quiescence search [8], null-move pruning [9], [10], [11], [12], and other selective enhancements [13] have been presented to avoid searching sub-trees that have minimal tactical advantages. We refer the readers to [14] for more details, because our work focuses on evolution visualization.

Chess visualization. All current chess visualizations focus on an instant instead of an overall evolution of a game, even though the AI engines behind the systems contain long-term tactical knowledge. Arena [1] and Fritz [2] show pieces that are under attack and step-by-step piece moves to guide players in beating their opponents. The thinking machine [15] sketches invisible thoughts of an AI to convey how the program searches for the best move based on the current chess position. In contrast, our chess evolution graph shows all position relations throughout a game and helps players predict future moves that an opponent may use in respond, in order to enhance their chess skills.

Tree and Graph visualization. Trees and Graphs are often used to represent relations between entities. The visualization of these structures allows users to discover intrinsic, hidden, non-trivial, and potential valuable knowledge [16]. The main difference of these two structures is the existence of cycles. Generally, tree visualizations can be categorized according to edge representation, alignment, and dimensionality [17]. That is, explicit representations clearly show connectivity using edges [18], [19] while implicit ones focus on data hierarchy [20]. The layout could be radial [18], [19], [21] or axis-parallel [22], [23]; the former places the root at the center and grows children nodes outward, whereas the latter maps tree levels parallel to one axis and span the structure to another. Finally, trees could be rendered in 2D or 3D spaces. In contrast to tree visualizations, where cycles do not exist, algorithms developed to graph visualizations strive to minimize crossovers and edge wiggles [24], [25], [26] with the purpose of enhancing readability and aesthetics. Moreover, plenty of navigation and clustering techniques are presented to improve space efficiency due to visual clutter in high complexity graphs. On the one hand, navigation approaches [27], [28] magnify regions of interest while shrinking or discarding the rest to show graph details. On the other hand, clustering methods [29] show the main trunk of a graph by hiding nodes and edges that are less important. Both categories of methods reveal only partial information due to the limitations of human cognitive ability.

Time-varying data visualization. A large amount of research works have been presented that focus on time-varying data analysis and visualization. These works attempted to reveal temporal trends of the underlying data by transforming the data into visual means for exploring complex relations. Some studies were presented to handle scientific imaging [30], [31]. Some of them applied dynamic graphs to information

visualization [32], [33], [34]. They showed changes of state over time while preserving the mental map and achieving good aesthetic quality. Although these works can be utilized to visualize relations over successive generations, they were not designed to convey decisions and the induced events in a chess game. Therefore, they are not sufficient in visualizing chess evolutions and helping players polish their chess skills.

Storyline visualization. Storyline visualization is a technique that is often used to convey sequential relational information. Each entity is represented as a narrow line going from left to right as time goes by and are grouped together if the entities have interactions. Kim et al. [35] developed a visualization system for genealogical data, which facilitates the identification of marriages, parent-child relations, siblings, and cousins over time. Ogawa and Ma [36] presented a technique to visualize software project evolution and to depict the scale and revisions of development to viewers who are not familiar with the project. Cui et al. [37] introduced a TextFlow method to convey complex relations among topic evolution trends, critical events, and keyword correlations so that people can know well current affairs. Reda et al. [38] presented a social network visualization method to reveal gradual changes of community structures over a period of time, thus allowing analysts to understand the formation, evolution, and dissolution of communities.

Our chess evolution visualization can also be considered as a kind of storyline visualization. However, its complexity is much higher than that of a general storyline visualization, because many potential chess positions after successive moves can be identified. In addition, our system conveys high level tactics and strategies computed by AI, which is very helpful in facilitating chess learning.

3 Design Methodology

The goal of our graph design is to reveal chess game evolutions. It ought to show potential chess positions after successive moves and highlight events such as *draws*, *effective checks*, and *checkmates*. The checks that lead to a worse position are neglected. For clarity, we use *position* to represent a game instant and *location* to indicate where a node is placed in the latter part of this paper.

Given that a position contains many piece locations, it is challenging to simultaneously show both spatial and sequential information of a game. This problem becomes even worse when offense and defense in a game is complex. Thus, we focus on showing sequential relations of chess positions and pointing out the successive moves achieving either tangible gain or loss. Apart from the evolution graph, our system additionally provides a chess board for the reference of piece placements. It shows how the pieces are controlled after a graph node is clicked. Our interactive interface also allows the examination of the chess board and the evolution graph on the fly so that users can realize game strategies easily.

Our system leverages an AI engine called Stockfish[1] to analyze a chess game and generate its evolution graph. That is, we record the sequence of chess positions made by players throughout a game and feed them into the chess engine to compute for the decision tree. This is followed by branch pruning, node merging, and branch shortening. To achieve an insightful analysis, the depth of the decision tree in our system is set to 20, which has the grandmaster chess level and is able to predict the chess positions after 10 moves[2]. Once the decision tree is determined, we pick up 4 to 9 move sequences that take larger advantages from the opponent at each position and discard the remains because we assume that the goal of a player is to beat the opponent. Specifically, suppose the player's move appear in the n^{th} best sequence from a position. We pick the 1^{st} to 4^{th} best sequences if $n \leq 4$, the 1^{st} to n^{th} best sequences if $4 < n \leq 8$, and the 1^{st} to 8^{th} best sequences plus the sequence that contains the player's move if $n \geq 9$. The consideration to only eight best sequences is based on more than 250 game play analyses in the GBR-ch tournament 2007[3], where the eight best sequences contain 93.4% of the piece moves made by players; taking the remaining 6.6% moves would lead to much worse positions and eventually lose the game. Note that all the real chess positions made by players are retained in the tree. Mistakes leading to failure are presented in our evolution graph.

3.1 Evolution graph generation

Our system computes a directed graph that reveals tactical knowledge from a decision tree and conveys a chess game evolution. We focus on the chess positions made by players and visualize potential positions they may achieve. The tree usually contains duplicated notes because a decision tree only spans potential moves at each node but neglects the fact that different move sequences may lead to an identical chess position. In addition, considering general chess games have 40 moves on average, rendering all the selected move sequences at each chess position would result in serious visual clutter (Figure 2.a). To prevent this problem, we merge the duplicated nodes (Figure 2.a→b) and shorten the branches (Figure 2.b→c) where the nodes on them have no events and contain only a parent and a child. We also compute node locations using GraphVis [4] to minimize edge crossings and achieve readability in our visualization. Hence,

1. http://stockfishchess.org/
2. Each move contains two plies, one for White and one for Black.
3. http://www.angelfire.com/games3/smartbridge/

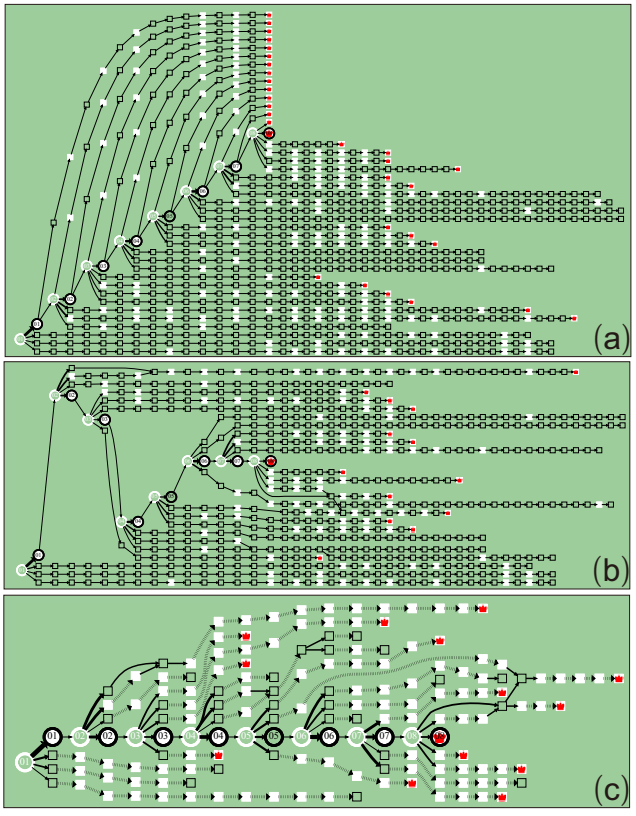

Fig. 2. (a) The evolution of a chess endgame is computed using Stockfish engine and stored in a decision tree. Chess positions and legal plies are represented in nodes and edges, respectively. Chess positions played by players and containing events are highlighted using our visual encodings (Section 3.2). (a)→(b) Our system merges the nodes representing the same chess position to reveal position relations. (b)→(c) We also shorten branches where nodes on them have only two neighbors to achieve readability. Notice that the nodes that are highlighted or adjacent to those highlighted persist during our graph simplification.

this evolution graph allows users to easily trace nodes via directed edges and foresee potential chess positions in the future. For example, in Figure 2.c, the nodes filled with white are the chess positions at which White can check Black, whereas the nodes filled with white and a red crown are the chess positions at which White can checkmate Black. Detailed visual encodings of nodes and edges are described in Section 3.2 and Figure 3. By tracing the evolution graph, one can easily identify that White will dominate the game eventually because many nodes derived from the first move show the Black's king is checked or checkmated.

Graph simplification. An intuitive way to merge duplicated nodes is to compare all pairs of arbitrary nodes in a tree and merge them together if piece locations on the nodes are identical. The process repeats until nodes in the tree are all different (Figure 2.b). Given that the cost of this brute force strategy is expensive, we apply the hash table technique presented by Hyatt and Cozzie [39] to reduce the number of comparison and speed up the merging process. The edges are re-directed whenever nodes are merged to form the evolution graph. The position relations of a chess game are then revealed. However, in Figure 2.b, there are some move sequences in the graph, where nodes have only a parent and a child. These sequences show that the subsequent game evolution is simple and can be shortened if there are no events on them (Figure 2.c). Namely, only nodes that have events or have more than two children will persist. Given that our visualization focuses on the chess positions made by players, their moves are also retained in the graph.

Graph layout. Our system computes the 2D location of each node to visualize the evolution graph. This graph is expected to extend from left to right as the move number increases and to spread potential locations in the vertical direction. The edges are then rendered according to the determined node locations. Specifically, we apply GraphVis [4] to compute an evolution graph $\mathbf{G} = \{\mathbf{V}, \mathbf{E}\}$, where $\mathbf{v}_i = \{x_i, y_i\} \in \mathbf{V}$ is the node location and $\{i, j\} \in \mathbf{E}$ is a directed edge from node i to node j. This tool first determines the order of each node in the horizontal direction $f_h(\mathbf{v})$ and then the order in the vertical direction $f_v(\mathbf{v})$. For the connected nodes i and j, if their horizontal orders are not adjacent, i.e., $|f_h(\mathbf{v}_i) - f_h(\mathbf{v}_j)| > 1$, it inserts a *virtual node* from order $min\{f_h(\mathbf{v}_i), f_h(\mathbf{v}_j)\}+1$ to order $max\{f_h(\mathbf{v}_i), f_h(\mathbf{v}_j)\}-1$ and connects the inserted and the original nodes to form a chain. As a result, the straight arrow from $\mathbf{v}_i$ to $\mathbf{v}_j$ can be replaced by an arrow on a Bézier curve, in which the virtual nodes serve as the control points, thus enhancing flexibility and reducing edge crossing. After that, the objective function is optimized to determine node locations. That is,

$$\sum_{\{i,j\}\in\mathbf{E}} k_{ij}\omega_{ij}|\mathbf{v}_i - \mathbf{v}_j|, \qquad \text{subject to}$$

$$|\mathbf{v}_a - \mathbf{v}_b| \geq \frac{s_a + s_b}{2} + d, \qquad \text{where}$$

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

$\omega_{ij} = 5 \times 10^4$ if both node i and node j are chess positions made by players and $\omega_{ij} = 10^4$ otherwise, a and b are nodes with $f_h(\mathbf{v}_a) = f_h(\mathbf{v}_b)$ and $|f_v(\mathbf{v}_a) - f_v(\mathbf{v}_b)| = 1$, s_a and s_b are the radii of node a and node b, respectively, and d is a constant parameter denoting the shortest distance between nodes. Clearly, the edge with a larger $k_{ij}\omega_{ij}$ would be shorter after the optimization because the accumulated energy grows rapidly if the corresponding edge length $|\mathbf{v}_i - \mathbf{v}_j|$ increases. Considering short and straight edges are preferable in achieving readability, we set k_{ij} to a larger value if either of its connecting node is visible. We also set ω_{ij} to a larger value if the move from i to

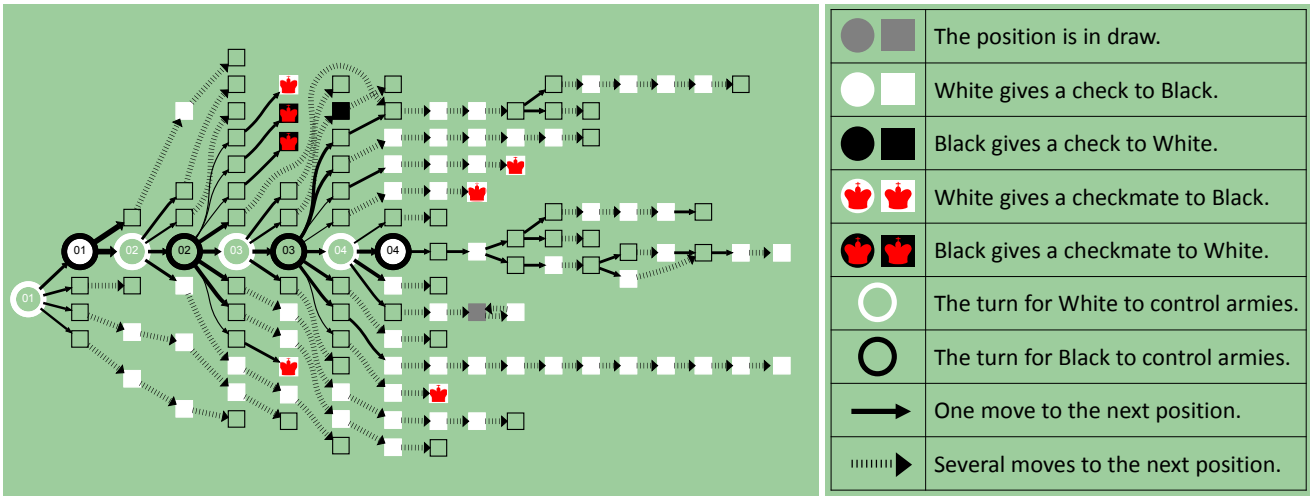

Fig. 3. Our system renders an evolution graph to convey potential positions throughout a chess game (left). In this graph, the actual moves made by users and the virtual moves estimated by AI are represented using circles and squares, respectively. We show the detailed meaning of each primitive at the right table. Notice that the primitives could be combined for additional meanings. For example, a white solid circle with a black border means White gives a check to Black and it is the turn for Black to react. In addition, directed edges connecting nodes are rendered in either solid or dotted arrows to indicate a single move or multiple moves, respectively. Furthermore, the thicker the arrow is, the better position the move achieves. To better understand a game evolution, we also label a move number in those nodes made by players (the main trunk) to represent the game progression.

j is made by players so that people can trace players' moves sequentially without frequently switching attention. Finally, parameter d controls the spaciousness of nodes, which is set to 0.5 in all our experiments.

We utilize GraphViz to minimize Equation 1, which considers node distances and edge chain straightness during optimization. For more details, we refer the readers to the directed graphs drawing technique presented by Gansner et al. [40].

3.2 Evolution graph rendering

We render the evolution graph to convey the subtle trend of a game, where its x coordinate shows the move number, while the y coordinate indicates the spread of potential chess positions. The nodes containing events, such as draws, effective checks and checkmates, are highlighted. We also label a move number in the nodes that represent chess positions made by players, in order to show the progression of a game. Finally, darkseagreen is selected as background color to achieve a harmonic viewing experience and enhance the contrast in the evolution graph.

Visual encoding of a node. Fundamentally, there are three design principles in our node appearance to achieve readability. 1) Circles and squares are used to represent positions made by players and computed by the chess engine, respectively. 2) Black and white represents the side that has advantage. Specifically, the boundary color shows the turn to control armies. The solid color shows the side that enjoys the event. In the case of a tie position, the node is filled with gray. 3) The chess position indicating that the king will be captured is additionally filled with a red crown icon. Therefore, based on the composition of the mentioned principles, we come up with a set of nodes representing different kinds of meanings as shown in Figure 3 (right). The two examples shown below illustrate how the composition works.

The node filled with white and a red crown icon means White gives a checkmate to Black. The black boundary shows that it is Black's turn to respond.

The node filled with black and a red crown icon means Black gives a checkmate to White. Given that this node is computed using AI and does not actually happen in a game, for a clear visualization, our system does not indicate the side that can control armies, as explained in the next paragraph.

We highlight draws, effective checks, and checkmates because the ultimate goal in chess is to capture the opposing king and prevent one's own king from being captured. These events reveal potential endings of a game and should be emphasized. Specifically, a draw or a checkmate shows a game result and an effective check takes the advantage by enforcing the opponent trades either a piece or a worse position for the king. Our system highlights all graph nodes containing events. However, the side that can control armies is indicated only on the nodes made by players. All the predicted chess positions are rendered with a black hallow square unless these positions have an event. We do so because the nodes computed by the chess engine do not actually happen in a game. They can be abstracted to enhance the visual saliency

of real positions in a game.

Visual encoding of an edge. We render directed graph edges using solid and dotted arrows to connect two chess positions achieved by a move or a sequence of moves, respectively. We also render solid arrows with different thicknesses to indicate the relative gained advantage of a move compared with the other moves starting from the same position. That is, good moves are rendered with thick arrows while bad moves are rendered with thin ones. Let δ_{ij} be the gained advantage from position i to position j, and min_i be the minimal value of δ_{ij} among edges coming out of node i, we compute the thickness of each edge using

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

where $\alpha = 1$ in all our experiments, followed by normalizing w_{ij} to a value within 1 and 30. A logarithm is used here because the gained values are typically among $[-30000, 30000]$. Without this transformation, a serious mistake or a genius attack would result in other moves that have smaller gained values incomparable. Note that each normalization considers only a small set of edges starting from node i, and the thickness comparison between edges starting from different nodes are meaningless. We show an example in Figure 3 for illustration.

3.3 Score chart

Given that the evolution graph represents potential moves at each chess position, we show a score chart at the bottom of the graph to reveal the quantized positional advantages of two players throughout a game. In this chart, the color of each primitive is used to indicate the side; its x coordinate is aligned with the move number of the evolution graph; and its y coordinate shows the log value of the score computed by the chess engine. Considering our goal is to convey a game evolution, the score chart contains not only the actual advantage of players in each move but also potential advantages they could obtain if they chose the other moves predicted by the engine. Here, we plot a circle on the chart to represent the actual score at the n^{th} move made by a player. To represent the potential scores at the n^{th} move that the player may obtain, we check the computed move sequences from the $n-1^{th}$ to the n^{th} move and render a score band to show the fluctuation of the estimated player's potential advantages. These score bands are also rendered with translucency to prevent occlusion. As a result, knowing how much loss a mistake would cause and how likely to turn defeat into victory becomes intuitive with the use of our score chart, as shown in Figures 6, 5, 7, and 8.

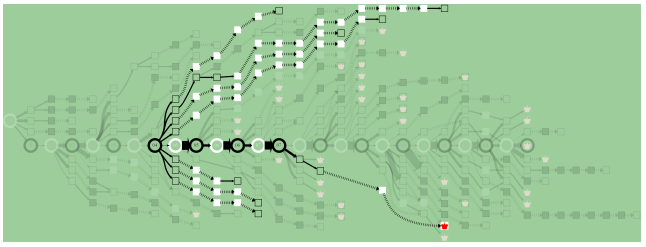

Fig. 4. Our system highlights the branches derived from the 4^{th} Black's move so that users can trace them easily.

4 Results and Discussions

We implemented and tested our visualization system on a desktop PC with Core i7 3.0 GHz CPU. The most expensive part of this visualization process is analysis, which may take up to 40 minutes to compute the decision tree. Fortunately, this step can be precomputed in advance. Meanwhile, our graph generation requires 3 seconds on average, including node merging, branch shortening and node position determination, which can also be precomputed. Finally, the graph rendering is in real time. Our system allows users to interactively explore the changes in a game over successive generations to help them learn tactics and strategies.

Our system allows users to investigate a chess game evolution from global to local viewpoints. It conveys information from growth and decline of tactical advantages throughout a game globally, to position relations and potential events locally, which may occur whenever a piece is moved. To trace position relations, users can click on the node of interest and our system would then enhance visual difference by reducing the color saturation of the nodes and edges that are not derived from this interested node (Figure 4). Besides, as some nodes are degenerated to dotted arrows after the branch shortening process, our system also allows the user to click on a dotted arrow, which would smoothly transit the arrow to a sequence of nodes (see our accompanying video), thus enabling users to observe detailed piece movements. Given that spatial piece placements are also provided, users can go back and forth to examine the chess board and the evolution graph to know how the tactics work in a game.

We tested our chess evolution visualization on a variety of real-world examples, including chess puzzles, chess played by both humans, by both computers, and by a human and a computer. This interactive visualization helps players gain a better understanding of a game because they can go back and forth to examine the potential chess positions and the successive moves responded by an opponent after the tactics are applied. Thus, users can understand why the critical move leads a game to a loss or turns defeat into victory. To demonstrate the feasibility of our visualization, we listed the professional comments

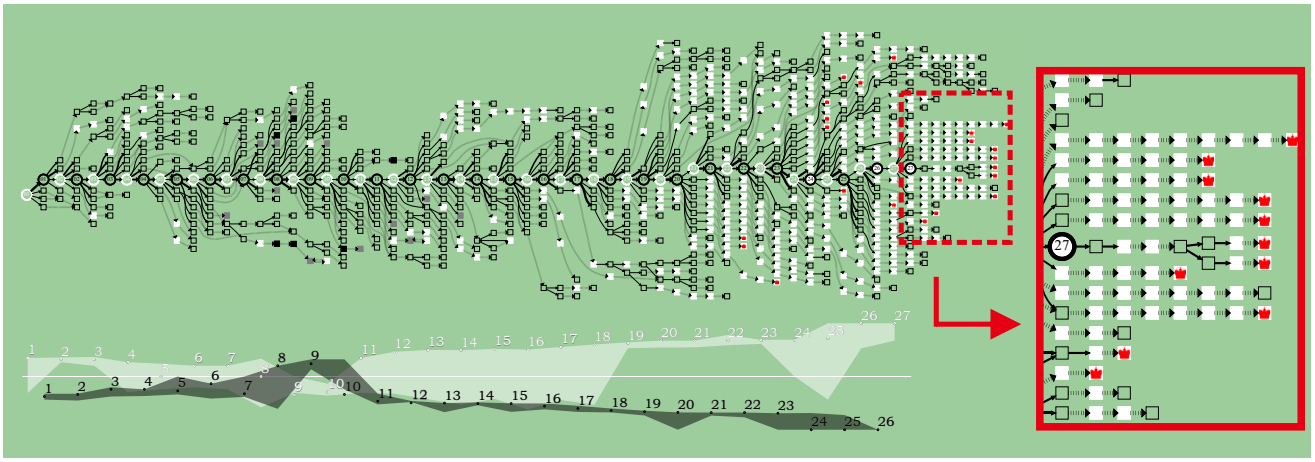

Fig. 5. Our chess visualization shows the evolution of a game, wherein the top evolution graph conveys position relations and highlights critical events, while the bottom score chart reveals the advantages of the Black and White players. In this contest, Black once had a better position even though he is at the defense position. However, he eventually lost the game after making a number of mistakes in moves 9, 10, 12, 24, and 26.

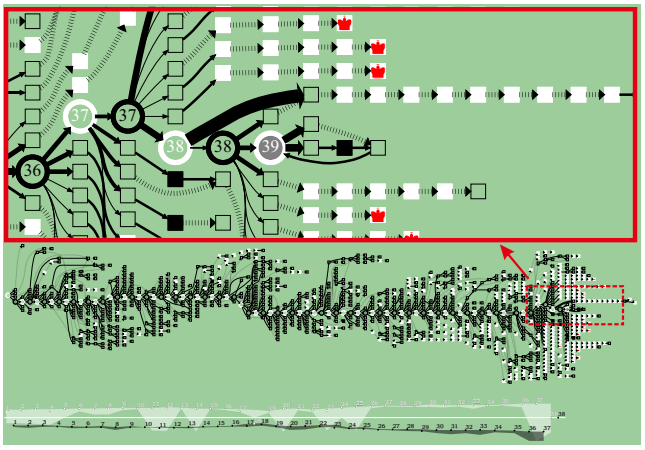

Fig. 6. Our evolution visualization shows that White keeps the offense position until the 38^{th} move. However, he made a mistake and the game eventually ended in a tie.

downloaded from a famous chess database[4], which verify that our system successfully highlights the tactical points. Please refer to the results shown in Figures 3–9, 7, 8, our supplemental materials and our accompanying video, especially when the piece movements are difficult to visualize in still images.

4.1 Case Studies

Case 1: Moderate Player vs. Moderate Player. Figure 5 shows a game played in World Open U2200. The White and Black players are James Plaskett and Sergei Shipov, respectively. Both are moderate players. In this game, our visualization reveals that White did not play very well at the 9^{th} and 10^{th} moves because the passed arrows are relatively thinner than others. Hence, the player lost his first hand advantage. The aligned score chart also shows this fact. However, as Black kept making faults at the 9^{th}, 10^{th}, 12^{th}, 24^{th}, and the 26^{th} moves, White eventually dominated the game. Notice that the passed arrows starting from the mentioned positions are relatively thinner than the computed ones. Our evolution graph also reveals that the checkmate to Black is unavoidable after the 27^{th} move. The estimated positions with highlighted events after this move explain the reason why Black gave up the game. We show the original comments from the annotator in our supplemental materials for the reference.

Case 2: Expert vs. Expert. Figure 6 shows a game played in Buenos Aires WCh. The White and Black players are Jose Raul Capablanca and Alexander Alekhine, respectively. Both players are experts. In this game, both players chose almost the best moves throughout the game. Given that White is on the offense, his tactical advantage is slightly over his opponent's before the 38^{th} move. However, he made a mistake at the 38^{th} move, and Black also caught the opportunity immediately to reach a draw, as we highlight in the top image of Figure 6. The only comment from the annotator in this game is "A huge missed opportunity and possible turning point of the match. Blowing Alekhine off the board as White, Capablanca walks into a perpetual check on his final move. Simply 38. Ke2 instead of 38. Kf2 would have won easily.". This comment supports our highlights.

Case 3: Deep Blue vs. Kasparov. Figure 7 shows a world-famous game played by Deep Blue and Garry Kasparov, wherein the computer played as White and the chess grandmaster played as Black. This was the second game in the 1997 rematch. In this game, the 37^{th} move played by Deep Blue was so ingenious and considered counterintuitive. IBM

4. http://www.chessbase.com/

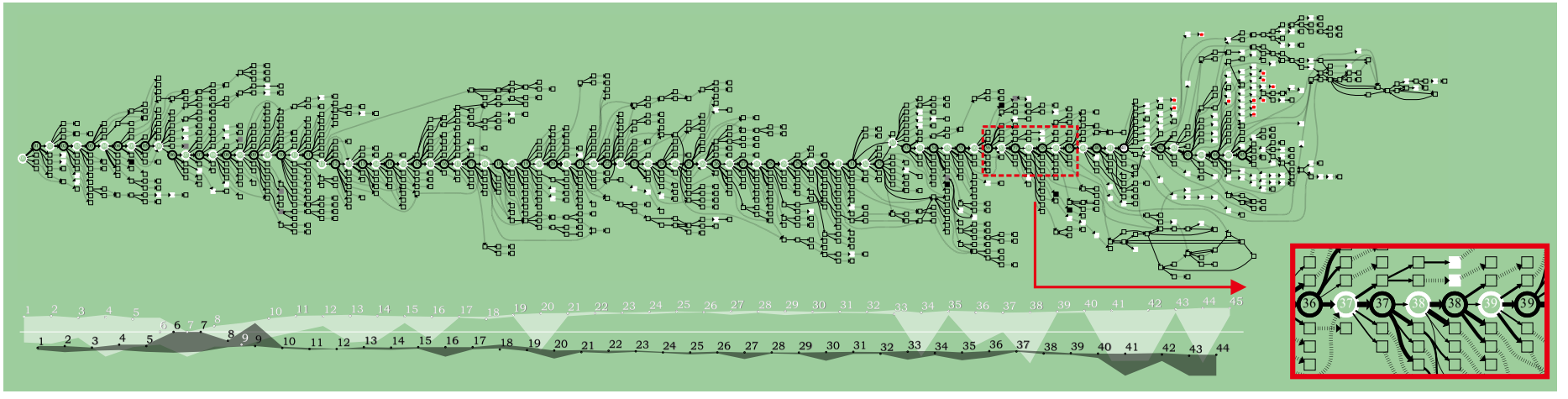

Fig. 7. A world-famous game played by Deep Blue (White) and Garry Kasparov (Black). The 37^{th} move played by Deep Blue was so ingenious and Kasparov's anxiety made him commit an unprecedented mistake. As the numbers of effective checks and checkmates increase rapidly after the 38^{th} move, the grandmaster eventually resigned.

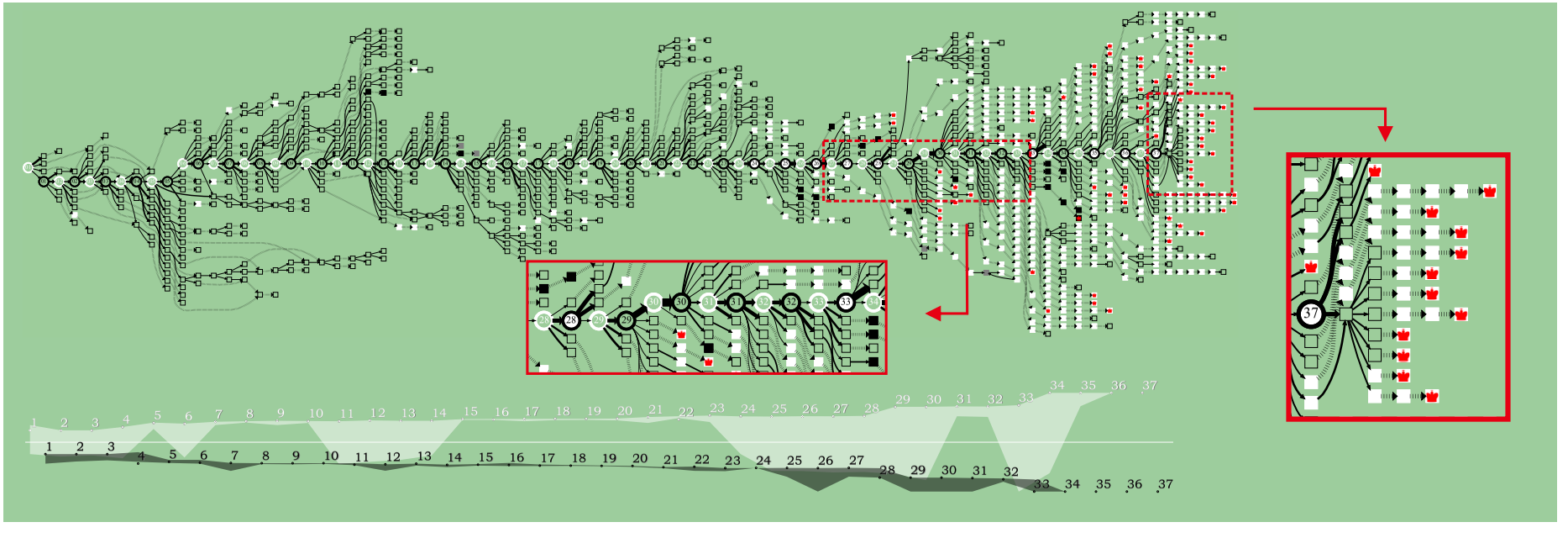

Fig. 8. A blitz chess played by Stockfish (White) and AnMon (Black) engines. Our evolution graph shows that AnMon did not play very well when its king was under attack at the 28^{th} and 33^{rd} moves. AnMon resigned at the 37^{th} move because the checkmate to its king was inevitable.

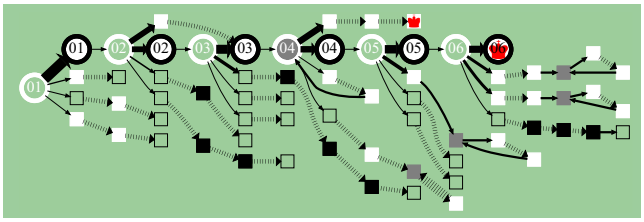

Fig. 9. Our visualization points out the solution that can solve the chess puzzle even though it has to take six successive moves to checkmate the Black's king. Notice the chess positions represented using circles.

Corporation was then accused of cheating because Kasparov considered that there must be superior intelligence behind Deep Blue. However, as shown in Figure 7, today's improved chess engine is capable of demonstrating the move to be the best during that game. Kasparov's anxiety also made him commit an unprecedented mistake after the move and forfeit a position that could have ended the match in a tie. One can also observe that the numbers of effective checks and checkmates to Black grew rapidly after the 38^{th} move, indicating that Deep Blue gradually controlled the game. Hence, it came as no surprise that the chess grandmaster eventually resigned.

Case 4: AI vs. AI. Figure 8 shows a blitz chess played by Stockfish and AnMon[5] engines, where both of them had only 5 seconds to respond and they were not able to perform deep and thorough searches during a game. In this game, Stockfish played as White and AnMon played as Black. Our post analysis revealed that computers did not make serious mistakes even in such a time-constrained scenario because they iteratively increased the spread and depth of the searching space while humans might miss some important pieces. As Stockfish was at the offense position and its chess level was higher than that of AnMon, it was not surprising that White dominated the game from the first move and Black eventually surrendered. Besides, by zooming to the graph, one can observe that Black did not play well when its king was under attack at the 28^{th} and 33^{rd} moves, which accelerated the game's conclusion.

Case 5: Chess Puzzle. Figure 9 demonstrates the evolution graph of the chess puzzle shown in Figure 1. The goal of this puzzle is to play as White and give a checkmate to Black within six moves. One can observe that in Figure 1, there are so many pieces left on the chess board and White does not dominate the game at that position. Considering there are so many potential chess positions, finding six successive moves to checkmate Black is very difficult. However, by observing our evolution graph, it is now easier to know that there is a way to checkmate Black's king by successive checks. Suppose White makes a correct decision at each position (white circular nodes), Black has only one legal move in return and eventually the king is checkmated. Although it is difficult to find the checkmate sequence, our visualization points out the solution clearly.

5. http://wbec-ridderkerk.nl/html/details1/AnMon.html

4.2 Limitations

Our chess visualization focuses on the evolution of a game. It shows the potential positions and tactical advantages after successive moves. Users, however, must still examine the piece placements to understand how an event occurs and how a strategy works. Thus, we claim that our system is not presented to replace existing chess GUIs but to complement them. We believe that fusing our evolution graph and piece placements seamlessly would greatly help users understand chess tactics and strategies. In addition, this framework visualizes present chess games, where the potential chess positions are precomputed. Thus, we plan to extend our chess evolution visualization system, which can dynamically update the graph structure based on user's query while minimizing unpleasant visual artifacts.

5 Evaluations

5.1 Comparison to visualization designs

An important key to learn chess tactics is to forecast potential positions and to predict how an opponent may respond in successive generations. Therefore, a good chess evolution visualization should be able to help players understand what would happen in successive moves and how a certain position could be reached. Given that current chess tools such as Arena [1] and Fritz [2] focus on spatial piece placements, using this kind of visualization, it is difficult to interpret chess position relations after successive generations because users have to memorize all piece placements in each step. In contrast, using graph visualization, in which nodes with events are highlighted and sequential node relations are indicated, users are allowed to trace connecting arrows and then obtain the results when making decisions. Generally, there may be some graph designs that can be used to show chess evolutions. For example, the graph could have: 1) a simplified or full structure, 2) an implicit or explicit representation, 3) a radial or axis parallel alignment, and 4) a 2D or 3D appearance. In the presented system, we adopt a simplified graph that enhances visual clearance, an explicit representation with edges that depicts position relations, an axis parallel alignment that allows obtaining game stages easily, and a 2D appearance that is free from occlusion. Furthermore, our visualization is carefully designed to fit the demands of chess evolution visualization. The theoretical analysis below shows the advantages.

Radial vs. Axis-Parallel. We choose the axis-parallel design to visualize the chess evolution because the

evolution is nearly sequential and it seldom reverses. For example, only few players move back the pieces and the captured pieces cannot be taken back. This characteristic is verified in our results, which indicate that almost all edges follow a left to right direction. The only exception in our experimental results is the 39^{th} move in Figure 6 because the game falls into the state of threefold repetition. Given that the axis-parallel layout matches the concept of most chess evolutions, we confirm that our design is intuitive and is sufficient in revealing the changes in a game over successive generations. Although a radial graph layout may also achieve the aim, this design inevitably introduces many edge intersections because chess positions that have similar move numbers are potentially connected. While branches extend outward but in different directions, those connected nodes are placed far away from each other. Therefore, it becomes more difficult to traverse sequential moves and foresee the potential chess positions after a certain position using this layout. In addition, due to lengthy distances between the connected nodes, users have to switch their attention frequently when investigating the graph structure. Our supplemental material (*ModerateRadial.pdf*), which shows the radial layout of the graph in Figure 5, verifies that the design is not capable of conveying game evolutions.

Full vs. Simplified. Rendering the full tree would result in serious visual clutter because of the complex structure. Specifically, a chess game has 40 actual moves on average, and a chess position in a decision tree usually has eight potential move sequences to predict future tactical advantages (Section 3.1). To verify the difficulty of discovering a game evolution from an unsimplified structure, we show the full decision tree of Figure 7 in our supplemental material (*DeepBlueFull.pdf*) for comparison.

Implicit vs. Explicit. Given that our goal is to present relations among chess positions, it is important to have an explicit representation with edges that clearly shows how a chess position is achieved. Implicit ones, such as a treemap that focuses on highlighting hierarchy, is not appropriate.

2D vs. 3D Although 3D visualization provides an additional dimension for data exhibition, most information visualizations are achieved in 2D due to flat displays. While the graph structure might be complex, this strategy also prevents node and edge occlusions, thereby allowing users to trace chess positions easily.

5.2 Comparison to current chess visualization

Some chess visualization tools have been presented to help players better understand a chess game and polish their chess skills. The most popular tools are Arena [1] and Fritz [2], both of which focus on spatial piece locations, while their main difference is the use of different AI engines. Our system has the same rationale but cuts into the point from sequential relations of chess positions. Considering that both Arena and Fritz visualize a chess game in a similar way, and Arena is an open source project but Fritz is a commercial one, we mainly compared Arena to our system.

To understand a chess game using Arena, users could click forward, backward, and input a move number to inspect the position with corresponding piece locations, move indications, and highlights to those pieces that are under attack. Clearly, the visualization in this tool focuses on the spatial relations of pieces. The long-term evolutions are conveyed using algebraic chess notation [3], which are difficult to understand. Under this circumstance, users have to investigate chess positions sequentially and memorize what has happened previously in each step, in order to understand the changes in a game over successive generations. Users also have to pay attention to multiple paths from a junction that lead to different results upon examination. Therefore, getting wise to a chess strategy using Arena is difficult and time consuming.

In contrast, our visualization is designed to convey the sequential relations of chess positions. Users can pan and zoom into the evolution graph so as to observe its topology while examining events, advantages, and piece placements of the interesting nodes. To understand a chess game, users can also trace connecting arrows of our graph to obtain the following: 1) the success or failure, 2) the involved chess positions, and 3) the potential weak points of a strategy based on the graph connectivity. They can then investigate piece locations by clicking a small number of nodes to obtain piece placements and fully interpret the strategy. Given that not all potential chess positions after the start of a strategy is involved, which is not necessarily investigated, and the sequential relations of chess positions are shown, the global-to-local viewpoints allows users to become more familiar with a chess strategy easily.

5.3 User study

We conducted a user study to verify the feasibility of our chess evolution visualization. We designed a multiple-choice questionnaire according to the suggestions of a chess expert. A total of 21 participants were asked to answer the evolutional questions by investigating our visualization system and Arena [1]. These participants came from diverse backgrounds and their ages ranged from 23 to 32 years. About 67% were novice players. They were aware of chess rules but had only little game experience. Given that no participants had used our system and only few of them were familiar with Arena, they were trained to use both systems before the study. Each training took about 10 minutes, i.e., 5 minutes for instruction and 5 minutes for practice.

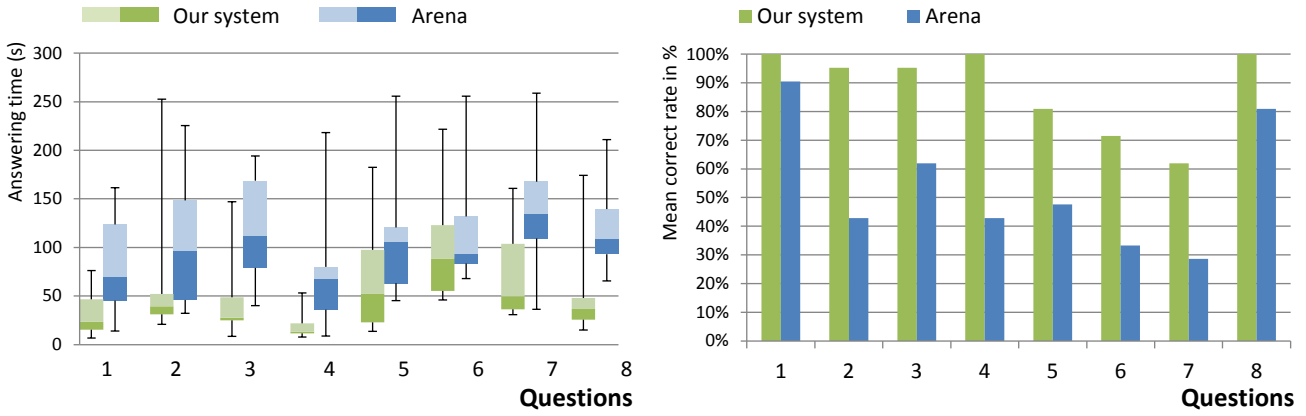

Fig. 10. (Left) The box and whisker plot represents the answering time of the questionnaires. The five-number summaries from top to bottom represent the 5^{th}, 25^{th}, 50^{th}, 75^{th}, and 95^{th} percentile. (Right) The mean correct rate of the questionnaires. Clearly, participants take less time but get higher correct rate when answering questions related to long term evolutions because our graph provides a global to local investigation interface.

Questionnaire. The participants were asked to answer two sets of questions, where each set contained 8 questions; each set of questions were answered based on the help from our system and Arena, respectively. The questionnaire included the move number to complete a chess puzzle, the judgment of tactical advantages among players, the critical moves leading to a win or a loss, and the chance to turn defeat into victory. Some of the questions related to local events and can be answered by observing highlights on a single chess position; some were about long-term tactics and required participants to investigate successive chess positions before providing their answer. We refer the readers to the user study questionnaire attached in our supplemental materials for more details.

Environment. To achieve a fair study, both the interfaces were displayed on a 27-inch screen with the resolution of 1920×1080; the chess engines behind the two GUIs were identical; two sets of the questions were the same but with different orders and phrasings. Furthermore, answering the sets of questions assisted by different graphical interfaces was done in a random order. We also asked the participants to take a 10-minute break before answering the second set of questions to avoid bias. All the user studies were performed in a quiet room, and each questionnaire took a participant about 50 minutes on average. We also found that most participants did not know the two sets of questions were the same until we told them the truth.

Results. We collected the answering time, error, and preference of each user during the study. Figure 10 shows the box and whisker plot to depict the answering time, and the bar chart to represent the mean correct rate. As can be seen, by using our visualization system, users took less time to answer questions but achieved higher correct rates. This phenomenon was more obvious in the 2^{nd} to 5^{th} questions because the questions related to global evolution. While using Arena requires sequential investigation, answering these questions was difficult and time consuming. In contrast, our visualization shows the entire trend of a game in a 2D space, thus making it easier to find out the key moves that result in winning or losing a game. Regarding the 6^{th} to 8^{th} questions, the participants were asked to examine a number of chess games, but the questions were about local events, the advantage of our system was slightly weakened in this scenario because the answers could be figured out in less than five moves. Finally, as for the 1^{st} question, the goal was to find sequences that can end the game in two moves, our evolution visualization only slightly outperforms Arena because the question was very simple. However, the mean correct rate and the answering time still showed that our visualization was much better than Arena even in terms of finding local events and strategies.

The hypothesis in the user study is that our chess evolution graph outperforms Arena in terms of both correct rate and answering time. We apply the Repeated Measures Analysis of Variance (RM-ANOVA) to analyze these two factors and verify the hypothesis after the study. As shown in Table 1, the reported post-hoc comparisons are significant at the $p < 0.08$ level.

Preference. Participants were asked to fill out satisfaction surveys after completing the questionnaire. The survey contained multiple areas, including intuitive interface, enjoyment, easy of use, efficiency, and helpfulness in teaching chess. The ratings ranged from 1 to 5, where 1 indicated totally disagree and 5 indicated totally agree. Figure 11 shows the ratings for both techniques. As can be seen, our visualization system is preferable compared with the traditional chess interface due to its global-to-local investigation manner.

We also asked for comments from the participants.

question	1	2	3	4	5	6	7	8
p-value (time)	1.3×10^{-3}	1.7×10^{-3}	4.7×10^{-7}	1.0×10^{-3}	4.1×10^{-3}	7.1×10^{-2}	2.9×10^{-5}	6.0×10^{-6}
p-value (correct rate)	7.7×10^{-2}	4.6×10^{-5}	3.8×10^{-3}	3.5×10^{-6}	1.2×10^{-2}	6.4×10^{-3}	1.5×10^{-2}	1.8×10^{-2}

TABLE 1
The significance values with respect to time and correct rate in our user study

They confirmed the quality and the feasibility of this new visualization system. Some participants did not get used to our design in the beginning because they had used traditional graphical interface for years. Nevertheless, in the end, they were very happy to see this new tool as they found that the global perspective of conveyance provided to be very useful. They then preferred our visualization because investigating the entire chess game step by step was no longer necessary, and the graph structure provided a much more intuitive way to point out successive moves compared with the algebraic notation sequences. For example, they could browse the bottom score chart to obtain the overall growth and decline of players' advantages with respect to the move number, and then check the thickness of each arrow to determine whether or not there was another better move from the position of interest. Finally they were able to trace the branches to predict the outcome of successive moves. The senior players especially liked the highlights provided for effective check events because many checks often result in a worse position, which might be traps setup by the opponent. Generally, after using our system for a while, most players agreed that our new visualization design is useful. Some chess teachers also looked forward to the system suitable for young children because our current evolution graph was still abstract to them. Thus, we consider involving these junior players in the development and the evaluation of a production-level system in the near future.

Summary. Our conducted user study and the collected user preferences highly support the fact that the presented chess evolution visualization is sufficient in revealing long-term strategies in a game. Specifically, participants took less time but got higher correct rate when utilizing our visualization to answer the questions related to global evolution. These results are not surprising because the trend of a chess game is disclosed in our evolution graph and a sequential investigation is not needed. Moreover, users can foresee potential events in the future and examine spatial piece movements to help them easily understand the interested tactic.

6 Conclusions

We have presented a chess evolution visualization system, which enables the user to understand a game from a global-to-local viewpoint. This visualization conveys the change over successive moves using a graph, in which the x and y coordinates represent the move number and the spread of potential positions, respectively. Given that sequential examination of a chess game is not necessary, experts can check our visualization to quickly understand the overview and critical events of a chess game. Furthermore, novice players can preview potential chess positions in future moves to predict how an opponent may respond and to learn chess tactics. The evaluation of our graph designs, the verification of professional comments to our visualization results, and the conducted user study demonstrate the feasibility of our system.

We emphasize that our system focuses on sequential relations among chess positions. To help them be aware of a strategy, users still have to investigate spatial piece movements so as to fully understand how the strategy works. Considering that users have to switch their attention in between the evolution graph and the piece placements, which may interrupt their thoughts, we plan to integrate the two kinds of information seamlessly in our future work.

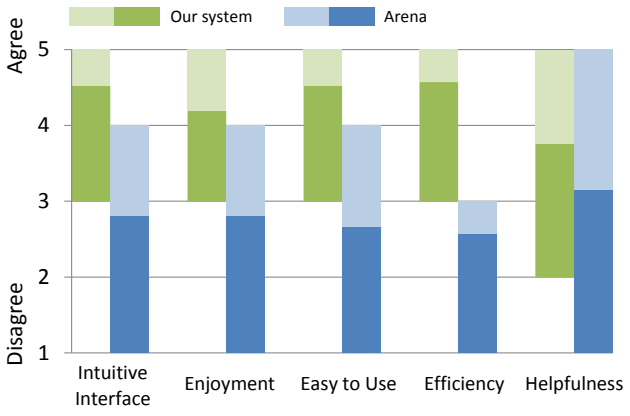

Fig. 11. Subjective ratings from our user study. The values ranging from 1 to 5 mean totally disagree to totally agree. In this chart, the three-number summaries from top to bottom are maximum, mean and minimum, respectively.

Acknowledgements

We thank the anonymous reviewers for their constructive comments. We are also grateful to Benjamin Hildreth for narrating the demo video, to Ke-Jung Chen for helping us edit the demo video, and to all the users who participated in the user study. This work was supported in part by the National

Science Council (101-2628-E-009-020-MY3, 102-2221-E-009-083-MY3, 101-2628-E-009-021-MY3, and 102-2221-E-009-082-MY3) and the UST-UCSD International Center of Excellence in Advanced Bioengineering sponsored by the Taiwan National Science Council IRiCE Program (NSC-101-2911-I-009-101).

REFERENCES

[1] M. Blume, "Arena chess gui," *Available online at http://www.playwitharena.com/*.
[2] "Fritz 13 – the truly great chess program," *Available online at http://en.chessbase.com/home/TabId/211/PostId/4007601*.
[3] E. Schiller, *Official Rules Of Chess*. ISBN: 978-1-58042-092-1, 2003.
[4] J. Ellson, E. Gansner, L. Koutsofios, S. North, G. Woodhull, S. Description, and L. Technologies, "Graphviz – open source graph drawing tools," in *Lecture Notes in Computer Science*. Springer-Verlag, 2001, pp. 483–484.
[5] P. Saariluoma, *Chess Players'thinking: A Cognitive Psychological Approach*. Routledge, 1995.
[6] A. Newell, J. C. Shaw, and H. A. Simon, "Chess-playing programs and the problem of complexity," *IBM J. Res. Dev.*, vol. 2, no. 4, pp. 320–335, 1958.
[7] T. A. Marsland and M. Campbell, "Parallel search of strongly ordered game trees," *ACM Comput. Surv.*, vol. 14, no. 4, pp. 533–551, 1982.
[8] D. F. Beal, "Mating sequences in the quiescence search," *ICCA Journal*, vol. 7, no. 3, pp. 133–137, 1984.
[9] ——, "Experiments with the null move," *Advances in Computer Chess*, vol. 5, pp. 65–79, 1989.
[10] C. Donninger, "Null move and deep search: Selective-search heuristics for obtuse chess programs," *ICCA Journal*, vol. 16, no. 3, pp. 137–143, 1993.
[11] G. Goetsch and M. S. Campbell, "Experiments with the null-move heuristic," *Computers, Chess, and Cognition*, pp. 159–168, 1990.
[12] E. A. Heinz, "Adaptive null-move pruning," *ICCA Journal*, vol. 18, no. 2, pp. 123–132, 1999.
[13] ——, "Extended futility pruning," *ICCA Journal*, vol. 21, no. 2, pp. 75–83, 1998.
[14] T. Marsland, "A review of game-tree pruning," *Int. Comp. Chess Assoc. Journal*, vol. 19, no. 1, pp. 3–19, Mar. 1986.
[15] M. Wattenberg, "Thinking machine," *Available online at http://www.bewitched.com/chess.html*.
[16] Y. Wang, S. T. Teoh, and K.-L. Ma, "Evaluating the effectiveness of tree visualization systems for knowledge discovery," in *Eurographics / IEEE VGTC Conference on Visualization (EUROVIS)*, 2006, pp. 67–74.
[17] H.-J. Schulz, "Treevis.net: A tree visualization reference," *IEEE Comput. Graph. Appl.*, vol. 31, no. 6, pp. 11–15, Nov. 2011.
[18] R. M. Wilson and R. D. Bergeron, "Dynamic hierarchy specification and visualization," in *IEEE Symposium on Information Visualization*, pp. 65–72.
[19] C.-C. Lin and H.-C. Yen, "On balloon drawings of rooted trees," in *International conference on Graph Drawing*.
[20] H.-J. Schulz, S. Hadlak, and H. Schumann, "The design space of implicit hierarchy visualization: A survey," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 4, pp. 393–411, April 2011.
[21] J. Yang, M. O. Ward, and E. A. Rundensteiner, "InterRing: An interactive tool for visually navigating and manipulating hierarchical structures," in *IEEE Symposium on Information Visualization*, pp. 77–84.
[22] T. D. Nguyen, T. B. Ho, and H. Shimodaira, "A visualization tool for interactive learning of large decision trees," in *IEEE International Conference on Tools with Artificial Intelligence*, pp. 28–35.
[23] S. van den Elzen and J. J. van Wijk, "BaobabView: Interactive construction and analysis of decision trees," in *IEEE Conference on Visual Analytics Science and Technology*, pp. 151–160.
[24] R. Tamassia, G. Di Battista, and C. Batini, "Automatic graph drawing and readability of diagrams," *IEEE Trans. Syst. Man Cybern.*, vol. 18, no. 1, pp. 61–79, Jan. 1988.
[25] G. D. Battista, P. Eades, R. Tamassia, and I. G. Tollis, *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, 1998.
[26] I. Herman, G. Melancon, and M. S. Marshall, "Graph visualization and navigation in information visualization: A survey." *IEEE Trans. Vis. Comput. Graph.*, vol. 6, no. 1, pp. 24–43, 2000.
[27] T. A. Keahey and E. L. Robertson, "Techniques for non-linear magnification transformations," in *IEEE Symposium on Information Visualization (INFOVIS)*, 1996, pp. 38–.
[28] G. W. Furnas and X. Zhang, "Muse: a multiscale editor," in *ACM symposium on User interface software and technology*, 1998, pp. 107–116.
[29] D. Kimelman, B. Leban, T. Roth, and D. Zernik, "Reduction of visual complexity in dynamic graphs," in *Graph Drawing*, 1994, pp. 218–225.
[30] H. Akibay and K.-L. May, "A tri-space visualization interface for analyzing time-varying multivariate volume data," in *Eurographics / IEEE VGTC Conference on Visualization (EUROVIS)*, 2007, pp. 115–122.
[31] J. Woodring and H.-W. Shen, "Multiscale time activity data exploration via temporal clustering visualization spreadsheet," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 1, pp. 123–137, 2009.
[32] A. V. Moere, "Time-varying data visualization using information flocking boids," in *IEEE Symposium on Information Visualization (INFOVIS)*, 2004, pp. 97–104.
[33] G. Kumar and M. Garland, "Visual exploration of complex time-varying graphs," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 805–812, 2006.
[34] K.-C. Feng, C. Wang, H.-W. Shen, and T.-Y. Lee, "Coherent time-varying graph drawing with multifocus+context interaction," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 8, pp. 1330–1342, 2012.
[35] N. W. Kim, S. K. Card, and J. Heer, "Tracing genealogical data with timenets," in *International Conference on Advanced Visual Interfaces*, 2010, pp. 241–248.
[36] M. Ogawa and K.-L. Ma, "Software evolution storylines," in *International symposium on Software visualization*, 2010, pp. 35–42.
[37] W. Cui, S. Liu, L. Tan, C. Shi, Y. Song, Z. Gao, H. Qu, and X. Tong, "Textflow: Towards better understanding of evolving topics in text," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, pp. 2412–2421, Dec. 2011.
[38] K. Reda, C. Tantipathananandh, A. Johnson, J. Leigh, and T. Berger-Wolf, "Visualizing the evolution of community structures in dynamic social networks," in *Eurographics / IEEE VGTC conference on Visualization (EuroVis)*, 2011, pp. 1061–1070.
[39] A. C. Robert M. Hyatt, "The effect of hash signature collisions in a chess program."
[40] E. R. Gansner, E. Koutsofios, S. C. North, and K.-P. Vo, "A technique for drawing directed graphs," *IEEE Trans. Softw. Eng.*, vol. 19, no. 3, pp. 214–230, Mar. 1993.

Wei-Li Lu received the BS and MS degrees from the Department of Computer Science and Information Engineering, National Central University, Taiwan, in 2011, and from the Department of Computer Science, National Chiao Tung University, Taiwan, in 2013, respectively. Now, he is working in Foxconn Electronics Inc. His research interests include computer graphics and visualization.

Yu-Shuen Wang received the BS and PhD degrees from the Department of Computer Science and Information Engineering, National Cheng-Kung University, in 2004 and 2010, respectively. He is currently an assistant professor of the Department of Computer Science at National Chiao Tung University (http://people.cs.nctu.edu.tw/ yushuen/). He leads the Computer Graphics and Visualization Lab at the Institute of Multimedia Engineering. His research interests include computer graphics, computational photography, and visualization.

Wen-Chieh Lin is an associate professor at the Department of Computer Science, National Chiao Tung University, Taiwan. He received the BS and MS degrees in control engineering from National Chiao Tung University, Hsinchu, Taiwan, in 1994 and 1996, respectively, and the Ph.D. degree in Robotics from the School of Computer Science at Carnegie Mellon University, Pittsburgh, in 2005. His research interests span several areas of computer graphics and computer vision. In particular, he is interested in physics-based animation, real-time rendering, and texture modeling and manipulation.