Analysis of Networks in Chess

DEREK FARREN, DANIEL TEMPLETON, and MEIJI WANG
Team 23
Stanford University

In this paper we examine the value of the information contained in networks constructed from positions in chess games with respect to game analysis and predictive power. The interrelations of pieces in a chess game represent complex and evolving networks that describe both tactical and strategic elements of the game. Even when a chess game is represented as a network of interrelations among the pieces, abstracting away all of the specifics and details of the game, such as the value of pieces or the notion of the board "center," the network still contains valuable information about the game played, when the critical moments are, and what the likely outcome will be. By representing a game as a series of networks, each modeling a different aspect of the connections among the pieces and the board, even richer information is available. By additionally considering the evolution of the graphs as a time series through the course of the game, still more information can be gathered. With only minimal encoding of game rules and logic, this series of evolving graphs is able to capture with surprising fidelity the evaluation of the game. In this paper we examine the information that can be extracted from the evolution of these networks and show that their predictive power surpasses that of an evaluation function that directly employs the rules and strategies of chess.

1. INTRODUCTION

The game of chess has been studied for over a millennia. Perhaps because of its inherent complexity and the challenge it poses even for human players, chess was one of the earliest topics of interest for the field of artificial intelligence, dating as far back as Wolfgang von Kempelen's Mechanical Turk in 1770 [[Standage]]. Chess poses an interesting problem for artificial intelligence because the state space of the problem is far too large to allow for exhaustive searches. Instead, chess programs typically play ahead as many moves as hardware and time constraints allow and then attempt to summarize the relative value of the candidate positions through an evaluation function. The evaluation function is often a complicated ensemble approach that attempts to capture both the tactical and strategic value of the position. The strategic value, in particular, is difficult to capture mathematically, though modern chess programs have now surpassed the capabilities of human players.

Given the demonstrated capabilities of systems such as Hydra, Rybka, and Deep Fritz, there remains little room for novel research into new chess programs or evaluation functions at this time. Instead, the intent of this paper is to explore the game of chess as a network problem to gain insight into the characteristics of a chess game and how they relate to the final outcome. We explore various ways of formulating a chess game as a network and extracting properties from those networks. We use those properties to predict the outcomes of games and show that network-based game features can be used to effectively predict game results.

2. BACKGROUND

2.1 Chess

The game of chess is played on a board consisting of 64 squares in an 8x8 grid (with the rows labeled as *ranks*, starting with the first rank as the row closest to the white player, and the columns labeled as *files*, starting with the first file as the column of the white player's far left) populated by 16 pieces of each of two colors. Each of two players controls all pieces of a single color. Each piece is of one of 6 types, each with different movement styles. A piece can capture another piece by occupying the captured piece's square on the board through a legal move. Using only this definition of the game plus the specific rules governing the starting positions of the pieces and their movement styles, we can construct networks representing the pieces and their connections with each other and the board, and from these networks, we can predict the outcomes of games and identify critical points in games.

For the purposes of this paper, a position in a chess game is the location of all active pieces on the board at a particular point during the game. Before a move has been made by any player, the board is in the *initial position*, with pawns on the second and seventh ranks and the remaining pieces in their corresponding places on the first and eighth ranks. A position may include anywhere from two (both kings) to thirty-two pieces, depending on the stage of the game.

2.2 Elo Ratings

The Elo rating system is a scoring system that attempts to assign a relative rating to players in chess games. Each player has a numeric score that goes up or down with each game played based on both the results of the game and the relative rating of the player's opponent. The relative rating of two players is intended to predict the expected number of wins by either player. Players with equal ratings who play a series of games are both expected to win an equal number of games. A player whose rating is 100 points higher than her opponent's should expect to win 64% of the games played. [Wikipedia]

The Elo rating system was developed by Arpad Elo in the 1950's for the United States Chess Federation (USCF) as a replacement for the Harkness rating system. Whereas the earlier rating systems relied on competitive rewards, the Elo rating system is based on a statistical estimation of the player's true skill. By awarding or removing points according to the results of games played, the Elo rating system attempts to adjust a player's rating until it reaches a value that represents the player's true skill. The amount of points won or lost depends on the relative rating of the player's opponent.

3. RELATED WORK IN COMPUTER SCIENCE

Chess has been studied exhaustively from a number of perspectives. In his seminal work on chess programs, [Shannon] introduced a lower bound on the game-tree complexity of chess as well as an evaluation function of the form:

$$\begin{aligned} f(P) =& 200(K-K') + 9(Q-Q') + 5(R-R') + \\ & 3(B-B'+N-N') + (P-P') - \\ & 0.5(D-D'+S-S'+I-I') + 0.1(M-M') \end{aligned}$$

in which:

—K,Q,R,B,N,P are the number of white kings, queens, rooks, bishops, knights and pawns.
—D,S,I are doubled, backward and isolated white pawns.
—M is white mobility (measured, say, as the number of legal moves available to white).

—The prime values are the corresponding values for black.

While modern chess programs have advanced significantly beyond the models presented by Shannon, his work sets a useful baseline for the capabilities of a chess program and serves as a yardstick for our approach.

[Atkin and Witten] presents a model for evaluating chess positions that is based on the mathematical relationship between the chess pieces and the squares on the chessboard. In this model, a piece and the squares that it "attacks" are labeled a "simplex," and simplexes are composed into "complexes." By analysis of the simplexes and complexes, Atkin and Witten showed that it is possible to make reasonably effective move predictions using real games as a point of comparison. Their work shares some commonalities with the approach taken in this paper, but their primary focus is on the geometry of the relationships as opposed to the network qualities of those relationships.

[Fernandez and Salmeron] experimented with search tree heuristics based on accepted piece and position values modified by a shallow Bayesian network built to adapt the engine's play to that of its partner. Using the Bayesian network, the engine attempts to classify the opponent's playing style and select a complementary playing style given the current stage of the game.

Outside of the realm of chess, much research has been done in the arena of modeling problems as networks to be analyzed [Wang et al] did it to discover hierarchies in the communication structures of modern corporations. [Yellaboina et al] used network analysis to predict the interaction of genes in E. coli. [Liang et al] experimented with discovering relationships among proteins using network properties. [Anderson et al] used network analysis to find inter- and intra-group relationships among schizophrenia patients.

4. MODEL FOR EVALUATING BOARD POSITIONS

4.1 From a chessboard to a network

As explained by {atkins, a position on a chess board can be viewed as a rich interrelationship among the pieces and the squares of the board that they occupy, defend, and attack. Considering each piece or square to be a node in a network, edges are naturally defined by the movement and potential movement (*reach*) of the pieces.

As an example, consider the position in figure 1. Assuming the pieces are nodes in the network, the most obvious edges are those connecting the white queen to the black queen and king and the black queen to the white queen. Also natural are the edges connecting the black king and queen and the edges connecting the white king and queen to each other and the white pawns. The former are *attacking* edges, whereas the later are *defending* edges. If we allow squares to be represented as nodes in the network, we can easily imagine edges from each of the pieces to the squares where it could move. In this position it is interesting to note that while both the black queen and black king can theoretically reach a large number of squares, because of the positional context, i.e. the black king is in check, the number of squares that can be reached by the black king and queen is actually quite small. There are, in fact, only four moves open to black (*Kf8*, *Kh8*, *Qg7*, and *Qxg6+*).

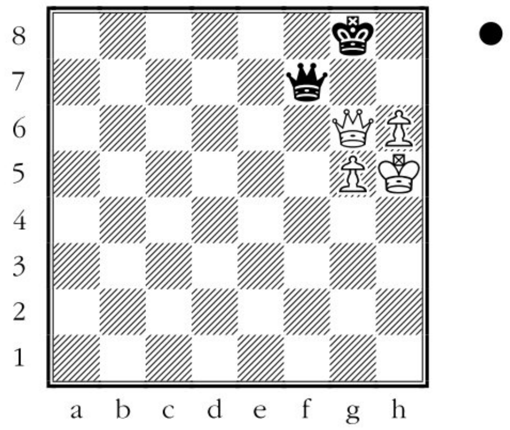

Fig. 1. An example game position

To represent a chess position as a network, we use the concepts from the preceding example to construct four different networks that model different aspects of the game.

The first network type constructed is the *support network*, which represents only relationships among pieces. In the support network, every node represents a piece and has the same *color* as the piece it represents. Whenever a piece has the option of capturing a piece of the opposing color, that potential to capture is represented in the support network as an edge. Whenever a piece could theoretically capture a piece of the same color (which is not actually allowed by the rules), that theoretical potential to capture is represented as an edge. Edges are directed and are colored according to the color of the piece that is represented by the edge's source node. Every edge is also assigned a *type* that is either 'attack' or 'defend', where attacking edges are edges between nodes of different colors, and defending edges are edges between nodes of the same color.

The second network type constructed is the *mobility network*, which represents the mobility of the pieces. Every node in the mobility network is a square on the board that is either occupied by a piece or directly reachable by a piece. Each node that represents an occupied square is assigned a color that matches the color of the piece that occupies the square represented by the node, the *source piece*. Network nodes that represent unoccupied squares have no color. Edges in the mobility network represent the ability of a piece to either capture an opposing piece or move into an unoccupied square. Every edge is directed and originates at a node that represents an occupied square. The destination for each edge is a node that represents either a square containing a piece that could be captured by the edge's source piece (as in the support network), or an unoccupied square that can be reached by the edge's source piece. There are no edges between nodes that represent squares occupied by pieces of the same color. Nodes that represent unoccupied squares all have out degree 0. Every edge is assigned a color that matches the color of the edge's source piece. For consistency, every edge is also assigned a type that is always 'attack'.

The third network type constructed is the *position network*, which represents the relationships among pieces and the squares they can reach. In the position network, every node represents a square of the board. Network nodes that represent occupied squares are assigned a color that is the same as that of the piece occupying the square represented by the node, the *souce piece*. Network nodes that represent unoccupied squares have no color. The position network is a strict union of the support network, where the nodes of the support network equate to the colored nodes of the position network, and the mobility network. The edges in the position network are exactly those described in the support and mobility networks above.

The forth network type constructed is the *tracking network*, which represents the history of movements of the pieces through the squares of the board. In the tracking network, every square of the board is represented as a node in the network. Unlike the previous three networks, the tracking network cannot be constructed from a single static position. In the course of the game, when a piece is moved from a source square to a destination square, an edge is created in the network between the nodes representing those squares. Once an edge is created in the tracking graph, that edge remains in the network for the rest of the game. Each move in the game adds one or more edges to the network, and the network representation of final game position contains edges for every move that occurred during the game. Note that the network does not encode any form of time series information about the moves. The edges are completely unordered, and one could, in theory, arrive at identical networks from playing the same series of moves in a permuted order.

In the tracking graph, pieces which are able to move distances longer than a single square, such as queens, rooks and bishops, create additional network edges when moving more than one square. For each square traversed during a single move, an edge is created between the previous square, starting with the source square, and the traversed square, ending with the destination square. For example, if the move that resulted in the position shown in Figure 1 was *Qg6*, where the white queen had previously been on emphe4, then that move would have created edges from the *e4* node to the *f5* node and from the *f5* node to the *g6* node. Each edge is assigned a color that matches the color of the piece whose movement created it. For convenience, all edges are assigned a type that is always 'attack'.

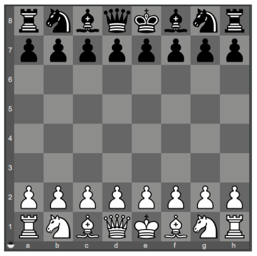

Fig. 2. Initial board position

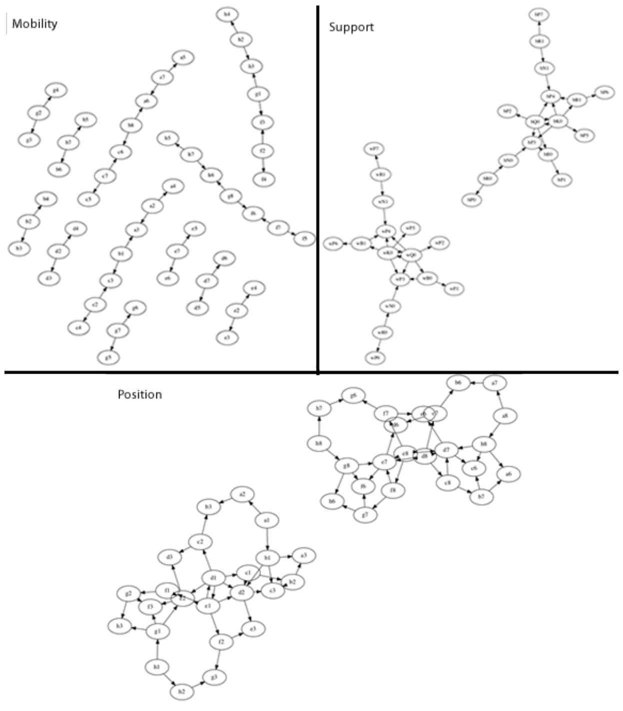

Fig. 3. Graph for the initial position

These four graph types are created (or updated in the case of the tracking graph) for every move in a game of chess, resulting in an evolving series of networks. With the exception of the tracking graph, each move in a game selects a network node (that corresponds to the moving piece or the square it occupies), removes all edges leaving that node, and creates new edges from that node to the set of nodes that are connected by the piece's new position. The new edges may overlap the original set of edges completely, partially, or not at all. In the tracking graph, the networks evolve by adding additional edges. Between any two nodes there may be multiple edges of the same or different colors.

4.2 Example positions and networks

Figure 2 shows the initial board position, and Figure 3 shows examples of three of the types of networks for the initial position. The tracking network is excluded as the network has no edges in the initial position.

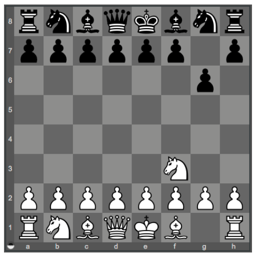

Fig. 4. Opening board position

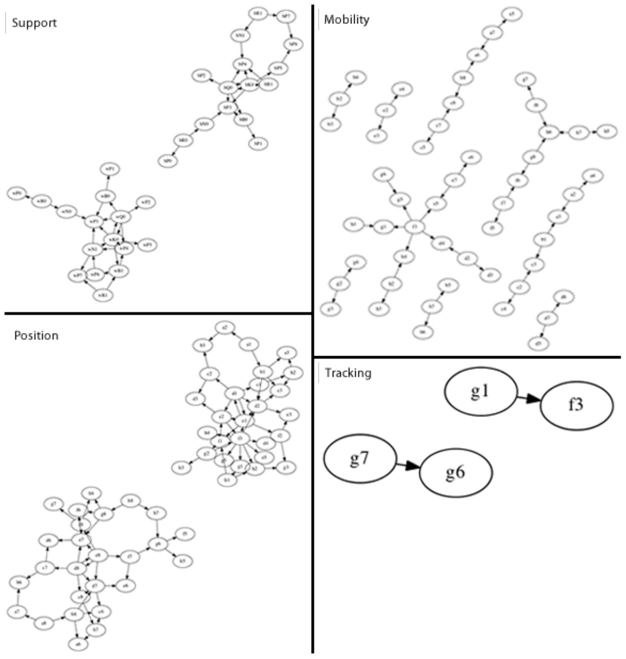

Fig. 5. Graphs for an opening position

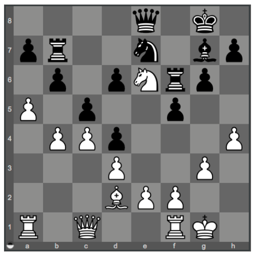

Fig. 6. Middle game board position

Figure 4 shows the board after the first moves, and Figure 5 shows examples of the four types of networks for that position.

Figure 6 shows the board during the middle game, and Figure 7 shows examples of the four types of networks for that position.

Figure 8 shows the board just before the final move, and Figure 0**??** shows examples of the four types of networks for that position.

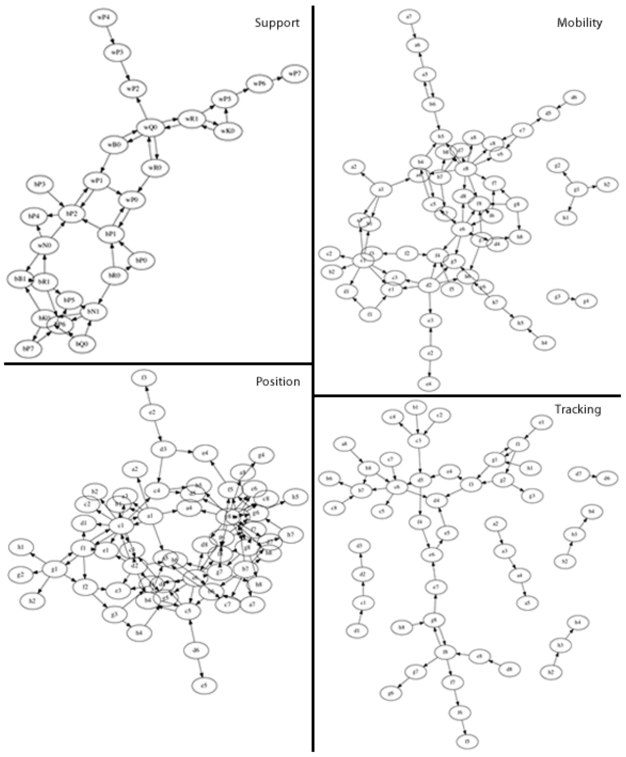

Fig. 7. Graphs for a middle game position

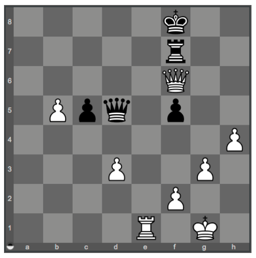

Fig. 8. End game board position

As can be seen from this set of graphs, there is a distinct pattern to the evolution of the networks in the course of the game. In the opening, the networks are very ordered with significant symmetry, not just between black and white but within the components as well. As the game progresses into the middle game, the network components become much more complex, with large numbers of nodes and edges and no clear order or structure. When the game passes into the end game, the chaos of the middle game subsides into a more regular structure with a distinct ”hub and spoke” character. In the tracking graph, the complexity of the middle game continues to grow into the end game, when the game reaches its maximum complexity.

4.3 Network properties

To evaluate the networks, we experimented with a large number of network properties as features. To gather meaningful data for predicting game results, we divide each network into a white component and a black component and compute the network

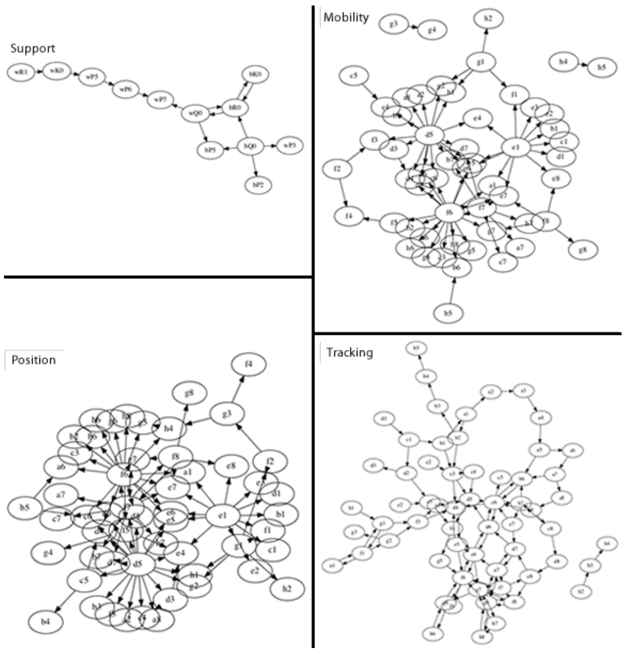

Fig. 9. Graphs for an end game position

properties for each component independently.

For each of the graphs types, we evaluate the following network features for the white and black components. In the following feature descriptions, the *friendly* component is the network component being examined, either black or white. The *opposing* component is the network component of the opposite color. Similarly, a *friendly* edge is an edge of the same color as the network component being examined, and an *opposing* edge is an edge of the opposite color.

(1) The smallest degree of any node that is greater than 0
(2) The largest degree of any node
(3) The average node degree
(4) The degree distribution — one feature for every degree size
(5) The graph density
(6) The number of nodes
(7) The number of edges
(8) The number of WCCs larger than 1 node
(9) The number of SCCs larger than 1 node
(10) The number of tree structures larger than 1 node
(11) The percentage of nodes that are included in WCCs that are larger than 1 node
(12) The percentage of nodes that are included in SCCs that are larger than 1 node
(13) The percentage of nodes that are included in tree structures that are larger than 1 node
(14) The percentage of nodes in WCCs of size 1
(15) The percentage of nodes contained in the smallest WCC that is larger than 1 node
(16) The percentage of nodes in the largest WCC
(17) The average percentage of nodes contained in WCCs larger than 1 node
(18) The WCC size distribution — one feature for the percentage of nodes contained in each WCC larger than 1 node
(19) The diameter of the smallest WCC that is larger than 1 node
(20) The diameter of the widest WCC
(21) The average diameter of WCCs with size greater than 1 node
(22) The WCC diameter distribution — one feature for the diameter of each WCC larger than 1 node
(23) The percentage of nodes contained in the smallest SCC that is larger than 1 node
(24) The percentage of nodes in the largest SCC
(25) The average percentage of nodes contained in SCCs larger than 1 node
(26) The SCC size distribution — one feature for the percentage of nodes contained in each SCC larger than 1 node
(27) The diameter of the smallest SCC that is larger than 1 node
(28) The diameter of the widest SCC
(29) The average diameter of SCCs with size greater than 1 node
(30) The SCC diameter distribution — one feature for the diameter of each SCC larger than 1 node
(31) The percentage of nodes in tree structures of size 1
(32) The percentage of nodes contained in the smallest tree structure that is larger than 1 node
(33) The percentage of nodes in the largest tree structure
(34) The average percentage of nodes contained in tree structures larger than 1 node
(35) The tree structure size distribution — one feature for the percentage of nodes contained in each tree structure larger than 1 node
(36) The depth of the smallest tree structure that is larger than 1 node
(37) The depth of the widest tree structure
(38) The average depth of tree structures with size greater than 1 node
(39) The tree structure diameter distribution — one feature for the diameter of each tree structure larger than 1 node
(40) The clustering coefficient
(41) The size of the largest clique
(42) The size of the minimum cardinality edge dominating set
(43) The maximum independent set size
(44) The size of the smallest maximal matching
(45) The degree assortativity coefficient [Newman]
(46) Transitivity — the fraction of all possible triangles that are present
(47) The number of attracting components
(48) The number of nodes in the 3-core
(49) The number of nodes in the 2-core
(50) The Rich club coefficient
(51) The ratio of friendly edges that enter the opposing component to the number of friendly edges that remain within the friendly component
(52) The ratio of the number opposing edges entering the friendly component from the opposing component to the number of friendly edges entering the opposing component from the friendly component
(53) The average number of friendly edges that enter the opposing component per node

(54) The average number of opposing edges that enter the friendly component per node

(55) The percentage of nodes with friendly edges that enter the opposing component

(56) The percentage of nodes with an opposing in degree greater than 0

In total, including the distribution features, each graph produces roughly 100 features per move, with roughly 30-50 moves per game. Each feature is the difference between the value of the property for white and the value of the property for black.

5. PREDICTING GAME OUTCOME

Each of the features listed above relates either directly or indirectly to the positional strategy in the game. Given enough data, it's almost always possible to discern patterns, not all of which are actually present in the distribution. In order to test the value of the network features defined above, we chose to use the features in a model to predict game outcomes.

As any chess player can attest, predicting the outcome of a game prior to the end game is challenging, and because of the possibility of human error, even predicting the outcome during the end game can be difficult. Given that none of the above network features take into account the rules or strategy of the game, predicting the outcome with any reliability would seem a significant challenge.

In order to benchmark the predictive power of the above network features, we chose to compare against Shannon's evaluation function. While simple and built to work with hardware constraints no longer present, Shannon's evaluation function successfully and succinctly captures the major factors in determining the strength of a position, and as is explained below, it has reasonably good predictive power.

To reduce the complexity of the experiment, the data set was filtered down to games in which there was a clear winner, i.e. no draws, and in which the win was by checkmate. The later criteria was chosen because it reduces the variability around the stage in which the game ended.

5.1 Approach

While the network properties defined above each model some aspect of game position and strategy, viewed as a snapshot of a single move, the information content is actually quite small. In order to have any true predictive power, the network features must be viewed as a time series, showing the evolution of the networks. A simple way to achieve this is to include the features from the previous m moves as independent model features. We experimentally determined that it is sufficient to include only the features from the previous 10 moves in the model. In order to reduce the variance caused by different opening styles, prediction is done counting backwards from the final move towards the initial move.

If we let f_{pi} be the feature vector composed of features from move pi, then the feature vector we found to be optimal is defined as:

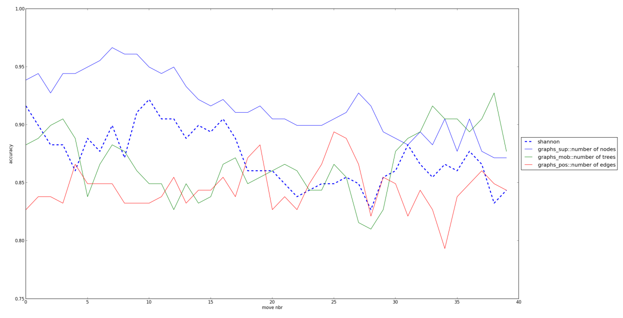

Fig. 10. Top 3 network features by accuracy

$$S_{pfi} = \{f_{pi}, f_{pi-1}, f_{pi-2}, ..., f_{pi-9}\}$$

Where:

—S_{pfi} is the player p feature vector based on feature value f at move i.

—f_{pi} is the player p feature f value at move i.

We decided to validate the predictive power of the networks with two different experiments. In the first experiment, we used features from the support, mobility, and position networks to classify games via logistic regression. Those three networks are closely related, as the position is a union of the mobility and support graphs, and can be logically treated as a unit. In the second experiment, we used features only from the tracking network to train an SVM classifier.

We also experimented with adding a decay factor, similar to the Bellman equation, but we determined empirically that the larger the value of γ used, the more accurate the results

5.2 Experiment 1

Using feature vector S_{pfi}, where f_{pi} consists only of a single feature f, we trained a logistic regression classifier on 2,000 games or a total of about 70,000 positions. Additionally, we trained a second logistic regression classifier using the results of Shannon's evaluation function as the only feature. The results are shown in Figure 10 for the top 3 most predictive features.

The first thing to note in these results is that the number of nodes in the support network is a much stronger predictor of game outcome than Shannon's evaluation function. The second thing to note is that the top features all represent very basic network features. One reason for that may be an insufficient number of network edges to enable the deeper network features to have meaning. On last thing to note is that the various features are most predictive at different parts of the game. For example, the number of nodes in the support network is most predictive in the end game, whereas the number of tree structures in the mobility network is most predictive in the middle game.

In order to take advantage of the different strengths of various features, we decided to try an ensemble approach by combining select features into a single model. Through experimentation, it was determined that the strongest ensemble classifier included

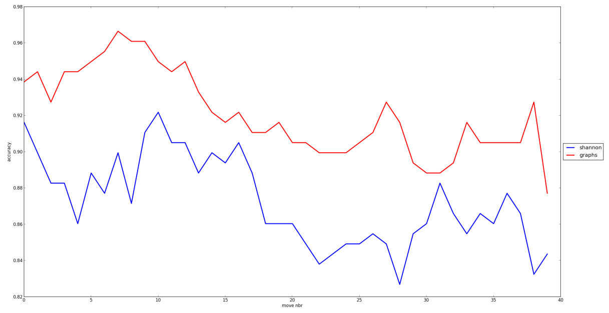

Fig. 11. Accuracy using ensemble features

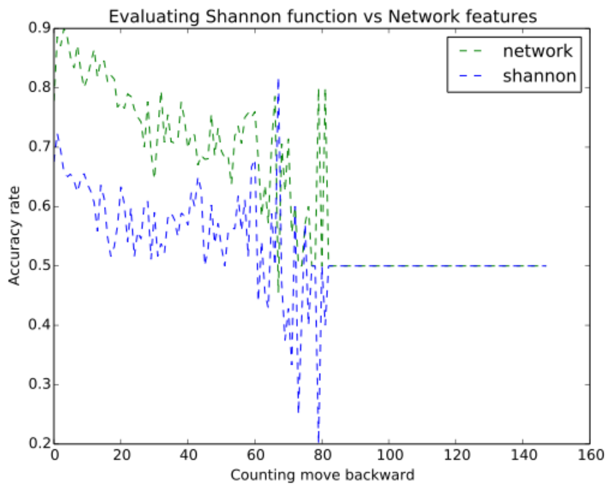

Fig. 12. Accuracy using tracking graph

only the top two features. One reason may be the interrelatedness of the three modeled network graphs. Figure 11 shows the result of training that classifier on 70,000 moves using a feature vector S'_{pfi}:

$$S'_{pfi} = \{S_{pf_1 i}, S_{pf_2 i}, S_{pf_3 i}\}$$

Where:

—S'_{pfi} is the player p combined feature vector based on feature f at move i.

—$S_{pf_j i}$ is the player p feature vector based on the j^{th} best performing feature f at move i.

5.3 Experiment 2

Using the S_{pfi} feature vector, we trained an SVM classifier on 1,000 games or a total of roughly 35,000 positions. We also trained a second SVM classifier on the same set of games using Shannon's evaluation function as the only feature. The results are shown in Figure 12.

The tracking network performs very well, predicting game outcomes with above 90% accuracy in the end game, significantly better than Shannon's evaluation function, and on par with the combination of the other three networks from the first experiment. One reason for this network's predictive power may be the abundance of edges, which allows some of the deeper network features to become fully realized.

5.4 Results

In both experiments, we were able to not only surpass the predictive power of Shannon's evaluation function, we were able to achieve accuracy levels above 90% in the end game. In addition, we successfully identified a set of features that are strongly predictive of game outcome. Perhaps unsurprisingly, those features line up well with common chess wisdom, but given that the trained classifiers were built using features completely oblivious to the strategy and even the rules of chess beyond basic piece movement, those results are a strong endorsement of the model of chess games as networks.

6. DATA COLLECTION PROCESS

The data for the proposed project is obtained from Ingo Schwartz' ICOfy Base (http://icofy-blog.de/icofy-base/), which includes roughly 5 million world campaign master games. We used specifically the IB1320 and IB1321 game collections which includes almost the complete collection of 5 million games. After filtering for games that end with checkmate, the final data set size available for this project was roughly 40,000 games. All game data is in PGN format and contains the full game transcriptions along with results and player information, such as names and Elo scores.

6.1 Translating PGN into a network

The PGN format uses a concise human-readable format that requires game context to interpret. As each move is notated using only the piece type and a destination square, without knowing the locations of all pieces, it is impossible to interpret a move out of context or out of sequence. For example:

```
1.d4 Nf6 2.c4 e6 3.Nf3 b6 4.Nc3 Bb7 5.a3 d5
    6.cxd5 Nxd5 7.e3 Be7 8.Bb5+ c6
......
43.Bxf6 gxf6 44.Kg3 Ra3+ 45.Kh4 Ra1 46.g3 a3
     47.Ra8 Ra2 1/2-1/2
```

To make PGN files consumable by our model, we had to build a simplistic chess engine to replay the above PGN moves. After each PGN move to performed by the engine, a corresponding function to convert the current chessboard position information into the network is called to generate networks for analysis. The Network package for Python was chosen as the tool to create and analyze the networks. It was chosen because of its simplicity, completeness, and its documentation, even though other packages, like SNAP or Snap.py, offer better performance. In the end, performance was an issue, but we feel that the tradeoff for ease of use was nonetheless positive.

7. DIFFICULTIES

Our source dataset is in PGN format. PGN is a data format made to be read by humans, not machines. Parsing that data and making sense of it was challenging, mainly because the game of chess has many details, and the format was not designed to be machine readable. One example: a play notated by Ne4 (move knight to e4) may potentially refer to either of the two knights (for example if one knight is in d2 and the other is in f2). A human, however, would realize that one knight is defending an attacked king (”pinned” in common chess parlance) and can’t be moved. Discovering and including these subtleties in the parsing algorithm was an arduous process.

8. FUTURE WORK

Through this research, we have demonstrated the viability of modeling chess games as networks. While the most predictive features are those that follow common chess wisdom, the fact that those features were uncovered by a model that is blind to common chess wisdom is significant and opens the door for further study of network models of chess games. While it is unlikely that such network models will improve upon the already highly tuned evaluation functions in modern chess engines, the use of network models in the evaluation of games to assist players in learning the nuances of chess strategy is a potential application of this research. Further research into the the relationship of the evolution of chess network models and chess strategy may yield practical results.

Along a different line, another opportunity for further research is the prediction of moves based on which move produces the most favorable network characteristics. The higher level insights provided by the network models may be useful in suggesting moves, which could mean such models could indeed be usefully included into a modern evaluation function.

An area where the current research could be expanded is in loosening the filtering requirements for games to allow for draw games to also be predicted. A three-way classification problem would require a different a classifier, such as a random forest.

REFERENCES

R. H. Atkin AND I. H. Witten. A Multi-dimensional Approach to Positional Chess *Int. J. Man-Machine Studies (1975) 7, 727-750*

Blasius and Tönjes. Zipf Law in the Popularity Distribution of Chess Openings. In *Phys. Rev. Lett. 103, 218701 (2009)*

Claude E. Shannon. Programming a Computer for Playing Chess. *Philosophical Magazine, Ser.7, Vol. 41, No. 314 - March 1950*

Herbert A. Simon. and Jonathan Schaeffer. The Game of Chess. *Handbook of Game theory, Volume I OF Handbooks in Economics, V*

Standage, T. *The Turk: The Life and Times of the Famous Eighteenth-Century Chess-Playing Machine* Walker Company, New York, 2002.

Wikipedia: Elo rating system, 2013. Retrieved December 9, 2013, from Wikipedia: http://en.wikipedia.org/wiki/Elo$_rating_system$

Newman, M. E. J. Mixing patterns in networks. *Phys. Rev. E, 67, 026126 (2003)*

Ariana Anderson and Mark S.Cohen. Decreased small-world functional network connectivity and clustering across resting state networks in schizophrenia: an fMRI classification tutorial. *Frontiers in Human Neuroscience. Vol. 7 Article 520(2013)*

Sailu Yellaboina. Kshama Goyal. and Shekhar C. Mande. Inferring genome-wide functional linkages in E. coliby combining improved genome context methods: Comparison with high-throughput experimental data. *Gnome Research 17:527535 (2007)*

Yi Wang. Marios Iliofotou. Michalis Faloutsos. and Bin Wu. Analyzing Interaction Communication Networks in Enterprises and Identifying Hierarchies. *International Workshop on Network Science (2011)*

Shide Liang. Dandan Zheng. Daron M Standley. Huarong Guo. and Chi Zhang. A novel function prediction approach using protein overlap networks. *BMC Systems Biology 7:61 (2013)*

Fernandez, A. and Salmeron, A. BayesChess: A computer chess program based on Bayesian networks. *Pattern Recognition Letters archive, Volume 29 Issue 8, June, 2008 Pages 1154-1159*