Computer Chess Programming as told by C.E. Shannon

Tsan-sheng Hsu

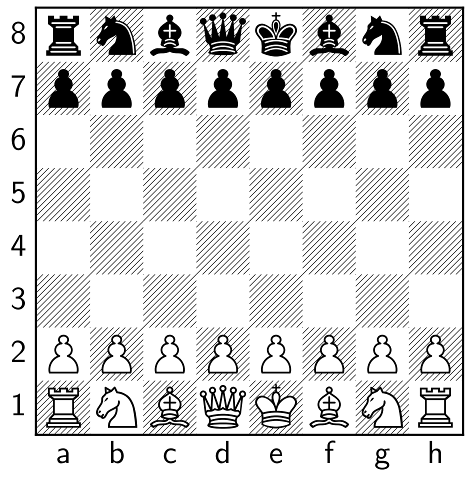
徐讚昇

tshsu@iis.sinica.edu.tw

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

Abstract

- C.E. Shannon.
  - 1916 – 2001.
  - The founding father of Information theory.
  - The founding father of digital circuit design.
- Ground breaking paper for computer game playing: "Programming a Computer for Playing Chess", 1950.
- Presented many novel ideas that are still being used today.



Estimating all possible positions

- **Original paper**
  - **In typical chess positions there will be of in the order of 30 legal moves.**
    - ▷ ***Thus a ply of White and then one for Black gives about 1000 possibilities.***
  - **A typical game lasts about 40 moves.**
    - ▷ ***A move consists of 2 plys, one made for each player in sequence.***
  - **There will be 10^{120} variations to be calculated from the initial position.**
    - ▷ ***Game tree complexity.***
  - **A machine operating at the rate of one variation per micro-second (10^{-6}) would require over 10^{90} years to calculate the first move.**
    - ▷ ***This is not practical.***
- **Comments:**
  - **The current CPU speed is about 10^{-9} second per instruction.**
  - **Can have $\leq 10^5$ cores.**
  - **About 10^8 faster, but still not fast enough.**

Have a dictionary of all possible positions

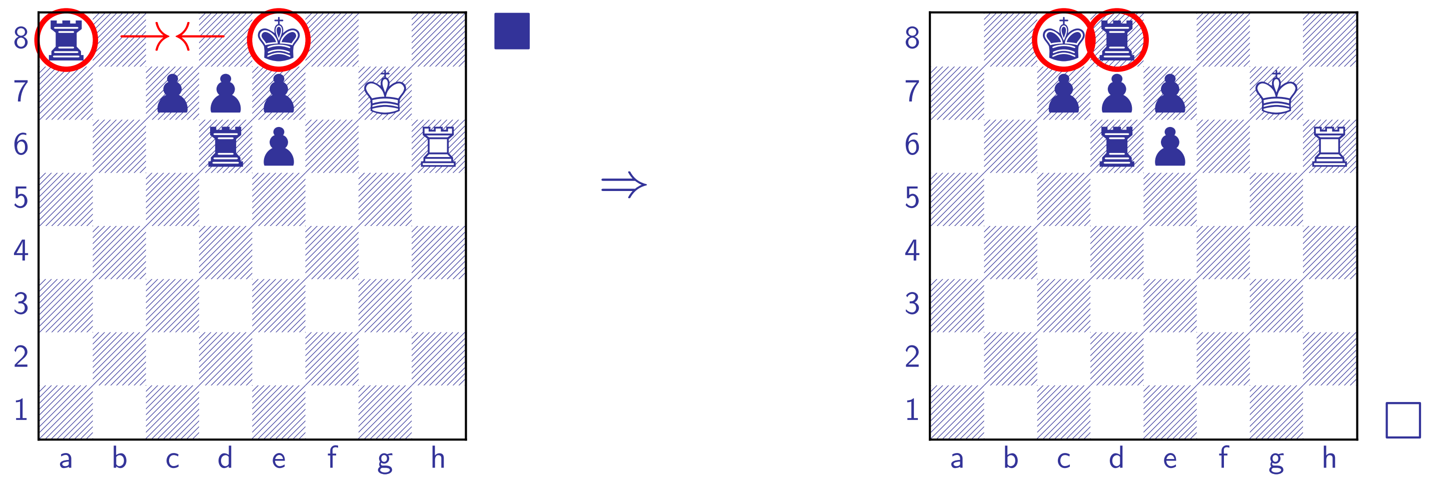
- **Original paper**
  - **The number of possible legal positions is in the order of $64!/(32!(8!)^2(2!)^6)$, or roughly 10^{43}.**
    - ▷ ***State space complexity.***
    - ▷ ***Must get rid of impossible arrangements.***
    - ▷ ***This number does not consider pawns after promotion.***
  - **Equally impractical.**
- **Comments**
  - **It is possible to enumerate small endgames.**
  - **Complete 3- to 5-piece, pawn-less 6-piece endgames have been built for Western chess.**
  - **Selected 6-piece endgames, e.g., KQQKQP for Western chess have also been built.**
    - ▷ ***Roughly $7.75 * 10^9$ positions per endgame.***
    - ▷ ***Perfect information.***
    - ▷ ***1.5 $\sim$ 3 $*10^{12}$ bytes for all 3- to 6-piece endgames.***
  - **The game of Awari was solved by storing all positions in 2002.**
    - ▷ ***A total of 889,063,398,406 ($\sim 10^{12}$) positions.***
  - **Checkers was solved in 2007 with a total endgame size of $3.9 * 10^{13}$.**

Phases of a chess game

- **A game can be divided into 3 phases.**
  - **Opening.**
    - ▷ *Last for about 10 moves.*
    - ▷ *Development of the pieces to good positions.*
  - **The middle game.**
    - ▷ *After the opening and last until a few pieces, e.g., king, pawns and 1 or 2 extra pieces, are left.*
    - ▷ *To obtain relatively good materials combinations and pawn structure.*
  - **The end game.**
    - ▷ *After the middle game until the game is over.*
    - ▷ *Concerning usage of the pawns.*
- **Different principles of play apply in the different phases.**

Evaluating function

- **A position p is the current board status.**
  - **A *legal* arrangement of pieces on the board.**
  - **Which side to move next.**
  - **The history of moves before.**
    - ▷ ***History affects the drawing rule, the right to castling . . .***
    - ▷ ***Other games such as Go and Chinese chess have rules considering history.***



- **An evaluating function f is an assessment of how good or bad the current position p is: $f(p)$.**

Perfect evaluating function

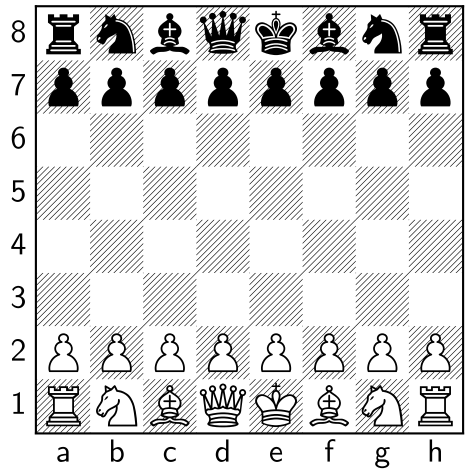
- **Perfect evaluating function f^*:**
  - $f^*(p) = 1$ **for a won position.**
  - $f^*(p) = 0$ **for a drawn position.**
  - $f^*(p) = -1$ **for a lost position.**
- **Perfect evaluating function is impossible for most games, and is not fun or educational.**
  - **A game between two unlimited intellect would proceed as follows.**
    - ▷ *They sit down at the chess-board, draw the colors, and then survey the pieces for a moment. Then either*
    - ▷ *(1) A says "I resign" or*
    - ▷ *(2) B says "I resign" or*
    - ▷ *(3) A says "I offer a draw," and B replies, "I accept."*
  - **This is not fun at all!**
  - **Very little can be used to enable computers being more useful.**

Approximate evaluating function

- **Approximate evaluation has a more or less continuous range of possible values, while an exact (or perfect) evaluation there are only three possible values, namely win, loss or draw.**
- **Factors considered in approximate evaluating functions:**
  - **The relative values of differences in materials.**
  - **Position of pieces.**
    - *Mobility: the freedom to move your pieces.*
    - $\cdots$
  - **Pawn structure: the relative positions of the pawns.**
  - **King safety.**
  - **Threat and attack.**
  - $\cdots$

Material values

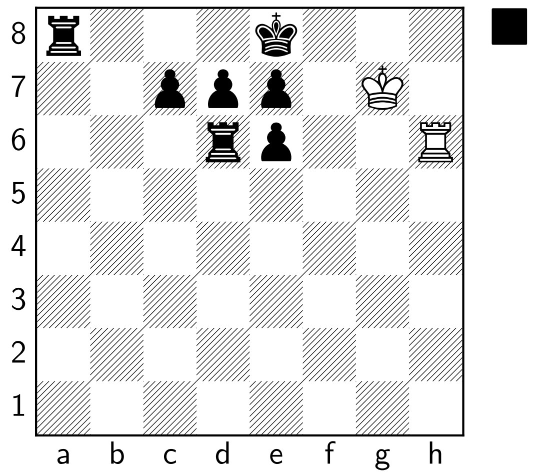
- **The relative values of differences in materials.**
  - **The values of queen, rook, bishop, knight and pawn are about 9, 5, 3, 3, and 1, respectively.**
  - **Note: the value of a pawn increases when it reaches the final rank which it is promoted to any piece, but the king, the player wants.**

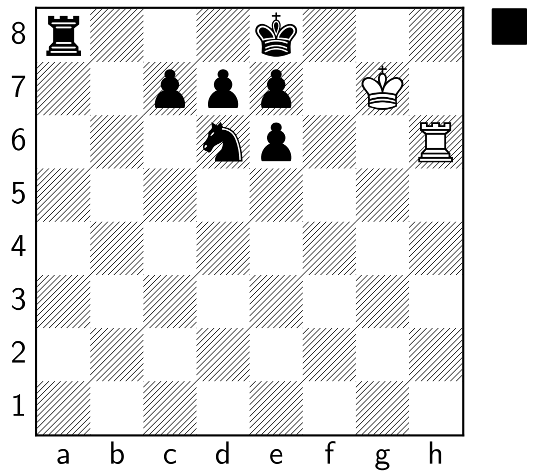


- **Q:**
  - **How to determine good relative values?**
  - **What relative values are logical?**
  - **Static values verse dynamic values?**

Dynamic material values

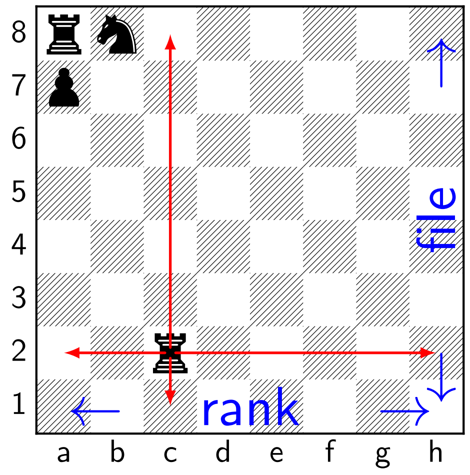
- **In the following example, a knight ($d6$) on the right is more useful than a rook ($d6$) on the left.**
  - **The knight can make a move $d6 - f5$ so that the potential threat from the white is neutralized.**





Positions of pieces (1/2)

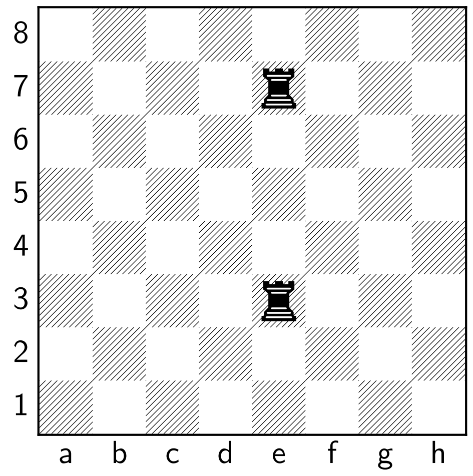
- **Mobility: the amount of freedom to move your pieces.**
  - **This is part of a more general principle that the side with the greater mobility, other things equal, has the better game.**
  - **Example: the rook at $a8$ has poor mobility, while the rook at $f2$ has good mobility and is at the 7th rank.**



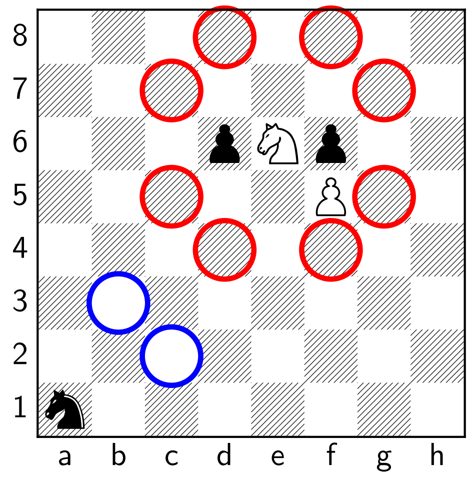
- **Note: file means column, rank means row.**

Positions of pieces (2/2)

- **Absolute positional information:**
  - **Advanced knights (at $e5$, $d5$, $c5$, $f5$, $e6$, $d6$, $c6$, $f6$), especially if protected by pawn and free from pawn attack.**
    - ▷ *Control of the center.*
  - **Rook on the 7th rank.**
- **Relative positional information:**
  - **Rook on open file, or semi-open file.**
  - **Doubled rooks.**



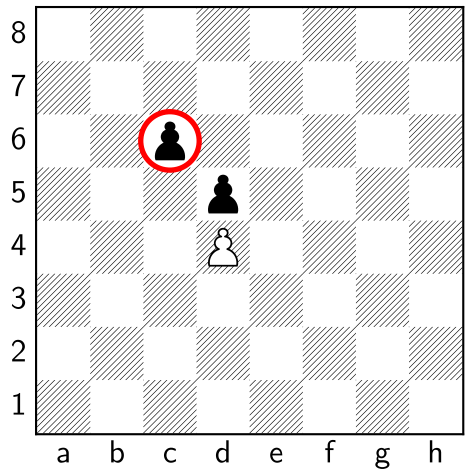
Doubled rooks.



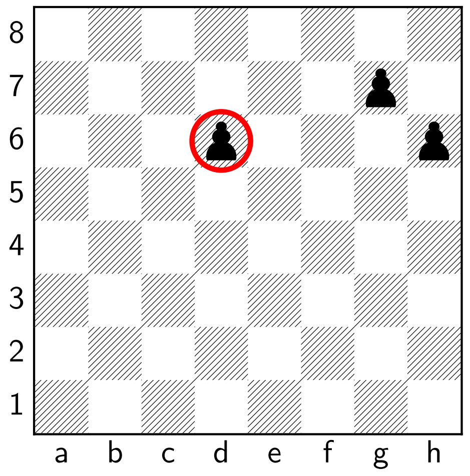
Knight at the center.

Pawn structure (1/2)

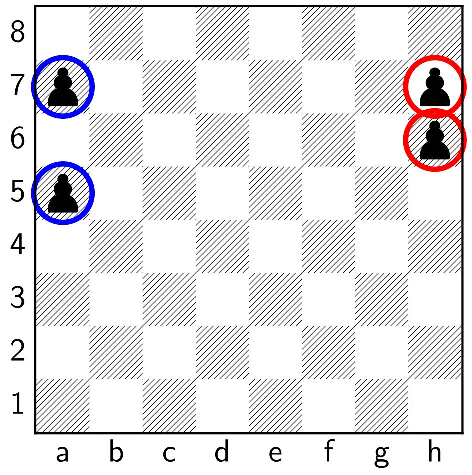
- **Example: Backward, isolated and doubled pawns are weak.**
  - ▷ ***Backward pawn:*** *a pawn that is behind the pawn of the same color on an adjacent file that cannot advance without losing of itself.*
  - ▷ ***Isolated pawn:*** *A pawn that has no friend pawn on the adjacent file.*
  - ▷ ***Doubled pawn:*** *two pawns of the same color on the same file.*



Backward pawn at $c6$.



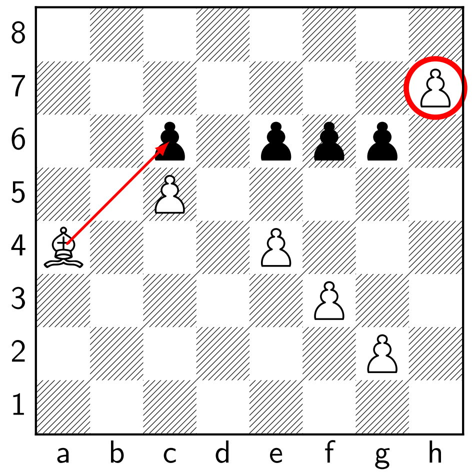
Isolated pawn at $d6$.



Doubled pawns at the a and h columns.

Pawn structure (2/2)

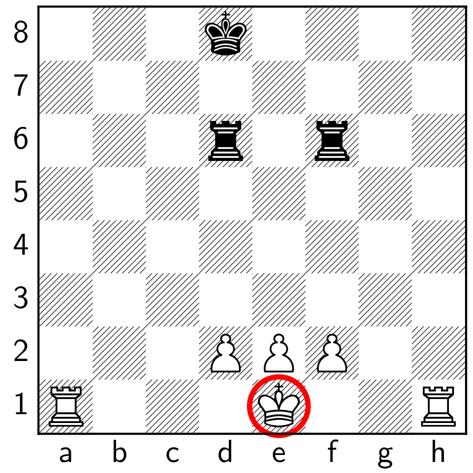
- **Absolute positional information:**
  - **Relative control of centre, for example, pawns at $e4$, $d4$, $c4$.**
- **Relative positional information:**
  - **Backward, isolated and doubled pawns.**
  - **Pawns on opposite colour squares from bishop.**
  - **Passed pawns: pawns that have no opposing pawns to prevent them from reaching the 8th rank.**



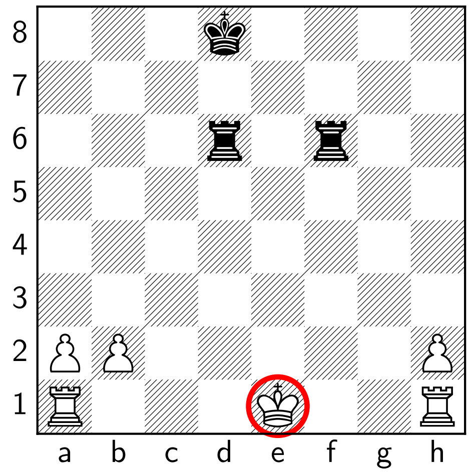
Passed pawn at $h7$ and a black pawn at $c6$ that is attacked by a white bishop.

King safety

- **An exposed king is a weakness (until the end-game).**



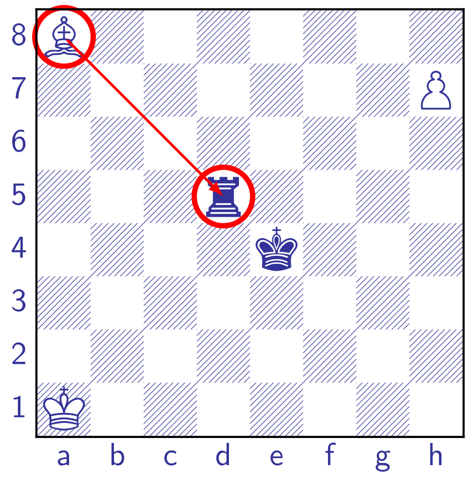
A protected king.



An exposed king.

Threat and attack

- **Commitments.**
  - ▷ ***Pieces which are required for guarding functions and, therefore, committed and with limited mobility; for example, the black king at e4 is committed to protect the rook at at $d5$.***
- **Attacks.**
  - ▷ ***Attacks on pieces which give one player an option of exchanging.***
  - ▷ ***Attacks on squares adjacent to king.***
- **Pins.**
  - ▷ ***Pins which mean here immobilizing pins where the pinned piece is of value not greater than the pinning piece; for example, a black rook at $d5$ is pinned by a bishop at $a8$.***



Making an evaluating function

- **Most chess and chess-like programs use approximate evaluating functions.**
  - **Materials.**
  - **Positions of pieces.**
    - ▷ *Mobility.*
    - ▷ *Absolute information.*
    - ▷ *Relative information.*
  - **Pawn structure.**
  - **King safety.**
  - **Threat and attack.**
  - $\cdots$
- $f(p) = w_1 * MIT(p) + w_2 * POS(p) + w_3 * Pawn(p) + \cdots$**, where**
  - p **is a position,**
  - $MIT(p)$ **is the material strength,**
  - $POS(p)$ **is the score for positions of pieces,**
  - $Pawn(p)$ **is the score of the pawn structure,**
  - **...**

Comments on evaluating functions

$$f(p) = w_1 * MIT(p) + w_2 * POS(p) + w_3 * Pawn(p) + \cdots$$

- **Putting "right" coefficients for different factors calculated above.**
  - **Static setting for simplicity.**
  - **Dynamic setting in practical situations.**
  - **May need to consider different evaluating functions during open-game, middle-game and end-games.**

Quiescent search

- **A simple type of evaluating function can be only applied in relatively quiescent positions.**
    - Positions that are not being checked.
    - Positions that are not in the middle of material exchanging.
    - Positions that are not in the middle of a sequence of moves with little choices.
- **To solve horizontal effect.**
    - Bad cases happened when the searching depth is fixed.
- **What to do when the leaf position is not quiescent.**
    - Apply quiescent search.
    - Generating selected moves and to reach leaves that are quiescent.
    - Be sure to compute Static Exchange Evaluation (SEE) for each piece p_1 you can capture before initiate a chain of piece exchanges (swap).
        - ▷ *Do not initiate a sequence of piece exchange unless you are profitable.*
        - ▷ *Use your least valued piece q_1 to capture p_1.*
        - ▷ *Recursively compute your opponent uses his least valued piece p_2 to capture back q_1 until the swap is over.*
        - ▷ *Compute the net gain.*

Strategy based on an evaluating function

- **A max-min strategy based on an approximate evaluating function $f(p)$.**
  - **In your move, you try to maximize your $f(p)$.**
  - **In the opponent's move, he tries to minimize $f(p)$.**
  - **Example of a one-move strategy**

$$\max_{\forall p'=next(p)} \{ \min_{\forall p''=next(p')} f(p'') \}$$

    where $next(p)$ is the positions that p can reach in one ply.
  - **Can be extended to a strategy with more moves.**

Comments to strategy

- **A strategy in which all variations are considered out to a definite number of moves and the move then determined from a max-min formula is called type A strategy.**
- **Max-min formula is well-known in optimization.**
  - Try to find a path in a graph from a source vertex to a destination vertex with the least number of vertices, but having the largest total edge cost using the max-min formula.
- **This is the basis for a max-min searching algorithm.**
  - Lots of improvements discovered for searching.
  - Alpha-beta pruning.
  - Various forward pruning techniques.

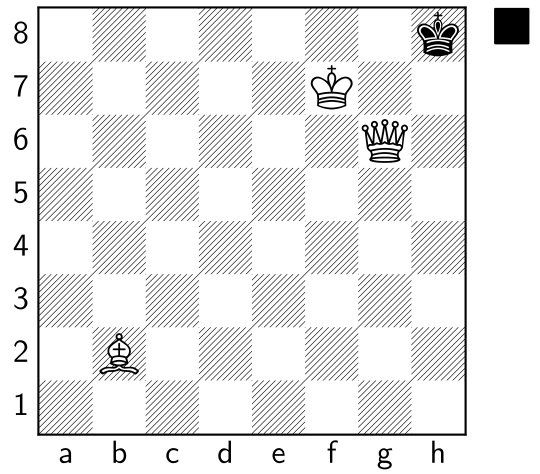
Programming

- **Methods of winning**
  - **Checkmate.**
    - ▷ *The king is in check and it is in check for every possible move.*
  - **Stalemate.**
    - ▷ *Winning by making the opponent having no legal next move.*
    - ▷ *In Western Chess, a suicide move is not legal, and stalemate results in a draw if it is not currently in check.*

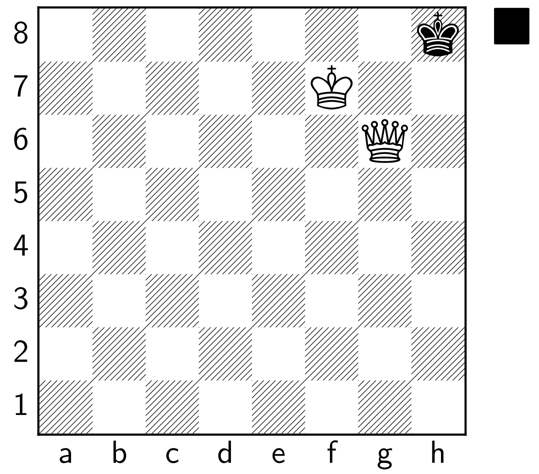
      Note: a suicide move is one that is not in check but will be after the move is made.
  - **Winning by capturing all pieces of the opponent: Chinese dark chess.**
- **Be aware of special configurations:**
  - **Zugzwang:**
    - ▷ *It is usually that you have an advantage if you have the right move.*
    - ▷ *In certain positions, a player is at a disadvantage if he is the next player to move.*
    - ▷ *If he can pass, then it is in a better situation.*

Example: Checkmate and Stalemate

- **Checkmate: it is in check and remains to be in check for every possible move.**
- **Stalemate if black is to move next.**
  - **A stalemate is one that is not in check, but will be in check for every possible move.**



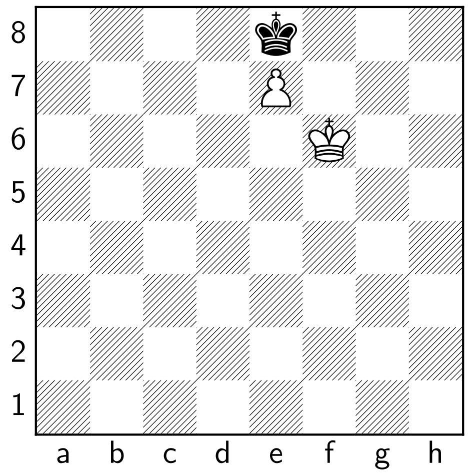
Checkmate if black is to move.



Stalemate if black is to move.

Example: Zugzwang

- **Zugzwang: White to move draws, while it will soon be a black loss if black to move next.**



Zugzwang for the black.

More Programming

- **Basic data structure for positions.**
  - **Using a 2-D array to represent the chess board.**
    - ▷ *For ease of programming, may want to fill the outside boarder with some special symbols.*
    - ▷ *Example in C: board[10][10], where we only use board[1..8][1..8] and board[0,9][*], board[*][0,9] are outside boarders.*
  - **Using numbers to represent different pieces.**
    - ▷ *Whether to use the same number of pieces of the same kind.*
- **Evaluating function.**
  - **Given a position and the next player, return a value that represents how good or bad this position is for the player to move next.**

Move generation

- **Move generation routines.**
  - **For each different piece, write a routine to check for possible legal moves.**
    - ▷ *Moving directions.*
    - ▷ *Considering capturing, blocking and game rules such as the right of castling and promotion.*
    - ▷ *Example: king_mov(), pawn_move(), ...*
- **Move ordering.**
  - **Unchecking.**
  - **Checking.**
  - **Uncapturing.**
  - **Capturing.**
    - ▷ *Capture opponent's most valued pieces (MVP) first.*
    - ▷ *Use my least valued piece (LVP) to capture.*
  - **Moving of piece in the order of importance.**
  - $\cdots$

Programming styles

- **High-level coding and functional decomposition.**
  - **Modules for above functions.**
  - **Combing all modules with a searching engine.**
    - ▷ ***A search engine is a program that finds the best strategy in a game tree with a limited depth.***
- **Comments:**
  - **Very little has changed over the years.**

Forced variations

- It is a pure fantasy that masters foresee everything or nearly everything;
  - The best course to follow is to note the major consequences for two moves, but try to work out forced variations as they go.
- Forced variations are those games that one player has little or no choices in playing.
- Some important variations to be considered:
  - Any piece is attacked by a piece of lower value or by more pieces than defenses.
  - Any check exists on a square controlled by the opponent.
- All important and forced variations need to be explored.
- Need also to explore variations that do not seem to be good for at least two moves, but no more than say 10 moves.

Improvements in the strategy

- **To improve the speed and strength of play, the machine must**
  - **examine forceful variations out as far as possible and evaluate only at reasonable positions, where some quasi-stability has been established;**
    - ▷ *Perform search until quiescent positions are found.*
  - **select the variations to be explored by some process so that the machine does not waste its time in totally pointless variations.**
- **A strategy with these two improvements is called a type B strategy.**

Comments

- Ideas are still being used today.
- Quiescent search is used to check forceful variations.
  - ▷ *Perform search until quiescent positions are found.*
- Move-ordering and other techniques to pick the best selection.
  - Real branching factor for Western chess is about 30.
  - Average useful or effective branching factor is about 2 to 3.
  - Chinese chess has a larger real branching factor, but its average effective branching factor is also about 2 to 3.
- Special rules of games
  - Chinese chess: rules for repetitions.
  - Go: rules for repetitions.
  - Shogi: rules for owing captured pieces.
  - Chinese dark chess: the rule to flip a previously covered piece.

Variations in play, style and strategy (1/2)

- It is interesting that the “style” of play by the machine can be changed very easily by altering some of the coefficients and numerical factors involved in the evaluating function and the other modules.
- A chess master, on the other hand, has available knowledge of hundreds or perhaps thousands of standard situations, stock combinations, and common manoeuvres based on pins, forks, discoveries, promotions, etc.
    - In a given position he recognizes some similarity to a familiar situation and this directs his mental calculations along the lines with greater probability of success.

Variations in play, style and strategy (2/2)

- ... books are written for human consumption, not for computing machines.
- It is not being suggested that we should design the strategy in our own image.
  - Rather it should be matched to the capacities and weakness of the computer.

Comments

- **Need to re-think the goal of writing a computer program that plays games.**
  - **To discover intelligence:**
    - ▷ ***What is considered intelligence for computers may not be considered so for human.***
  - **To have fun:**
    - ▷ ***A very strong program may not be a program that gives you the most pleasure.***
  - **To find ways to make computers more helpful to human.**
    - ▷ ***Techniques or (machine) intelligence discovered may be useful to computers performing other tasks.***

References and further readings

* C. E. Shannon. Programming a computer for playing chess. *Philosophical Magazine*, 41(314):256–275, 1950.
- A. Samuel. Programming computers to play games. *Advances in Computers*, 1:165–192, 1960.
- A. Samuel. Some studies in machine learning using the game of checkers. *IBM J. Res. Develop.*, 11:601–617, 1967.