Computer Chess Programming as told by C.E. Shannon in the year 1950

Tsan-sheng Hsu

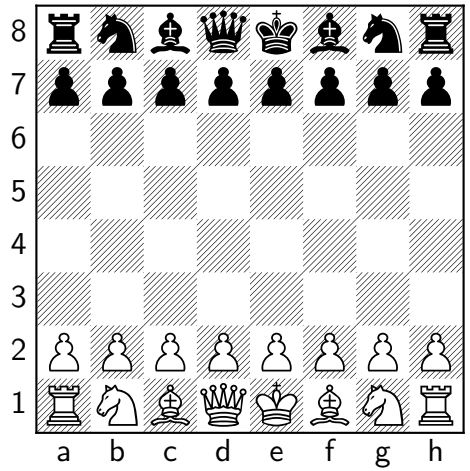
徐讚昇

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**
  - **A typical chess position has in the order of 30 legal next moves.**
    - ▷ *A legal next move is a ply that one can play.*
    - ▷ *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 to a goal 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} or maybe $2 \cdot 10^{-10}$ second (5 G Hz) 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 or even unreasonable arrangements.***
    - ▷ ***This number does not consider pawns after promotion[1].***
  - **Equally impractical.**
  - **The above figure is for the case that all, namely 32, pieces are not captured. That is, we first pick 32 cells out of 64 possibles ones, and then we count the combinations to place the pieces.**

$$\binom{64}{32} \cdot \frac{32!}{(8!)^2(2!)^6}$$

    - ▷ ***Contain unreasonable and maybe illegal positions such as ones caused by pawns' placement.***
  - **It gets very complicated when the number of pieces left is < 32.**

[1] A pawn is promoted to any piece you want after advancing to the final column.

Comments

- **It is possible to enumerate small endgames.**
- **Complete 3- to 5-piece, pawn-less 6-piece endgames have been built for Western chess.**
  - ▷ *1.5 $\sim$ 3 $*10^{12}$ bytes for all 3- to 6-piece endgames.*
- **7-piece endgames, e.g. 4 vs 3 and 5 vs 2, for Western chess have also been built.**
  - ▷ *Roughly 150T bytes using a supercomputer.*
  - ▷ *DTM, or (minimum) Distance To Mate.*
  - ▷ *Year 2012.*
- **Comparing other games**
  - **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 $3.9 * 10^{13}$.**
- **Probably can store upto $\sim 10^{15}$ positions since 2021.**

Phases of a chess game

- **A game can be divided into 3 phases.**
  - **Opening.**
    - ▷ *Last for about 10 moves.*
    - ▷ *Deploy 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* ***material combinations*** *and pawn structures.*
  - **The end game.**
    - ▷ *After the middle game until the game is over.*
    - ▷ *Concerning usage of the pawns.*
- **Apply different strategies in different phases.**

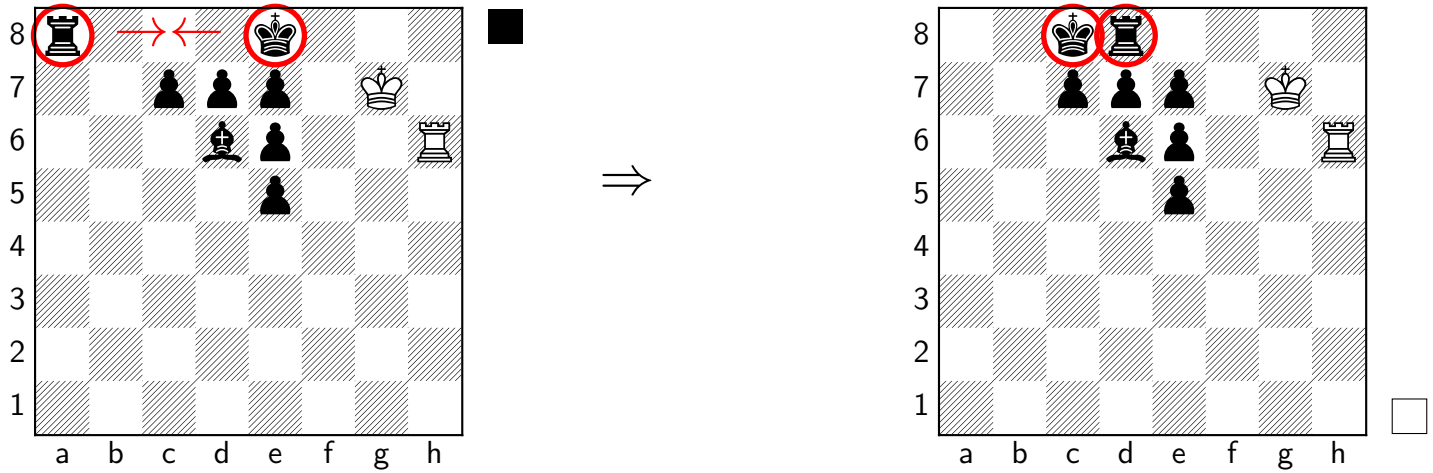
Evaluation function

- **It is usually the case you cannot search to the end of a game, so we need a way to evaluate how good a position is when searching stops.**
- **A position p is the current board status.**
  - **A *legal* arrangement of pieces on the board.**
  - **Which side to move next: turn, player to play next.**
  - **The history of moves before.**
    - ▷ ***History affects the cycling rule[2], the right to castling . . .***
    - ▷ ***Other games such as Go and Chinese chess have rules considering history.***
- **An evaluation function f is an assessment of how good or bad the current position p is: $f(p)$.**

[2] In games, to enforce meaningful playing, it is often the case that when a cyclic loop of positions is created, the game ends with either a draw or a loss for the cycle creator.

How History affects the Evaluation

- **In chess, under certain conditions, a special moving rule called the right to castling can be exercised by a player.**
  - **If both his king and castle are not moved ever, and exercising the right to castling does not result in checking the opponent, then the king and castle can move at the same time in one ply as illustrated below.**



- **In the above example,**
  - **if the black does not have the right to castling, then he will be checkmated in a few plys;**
  - **if the black does, then he can use that to result in a great advantage.**

Perfect evaluation function

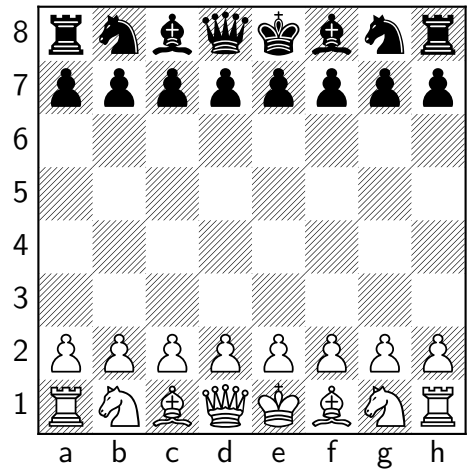
- **Perfect evaluation function f^*:**
  - $f^*(p) = 1$ **for a won position.**
  - $f^*(p) = 0$ **for a drawn position.**
  - $f^*(p) = -1$ **for a lost position.**
- **Perfect evaluation function is very difficult to obtain for most games.**
  - **A game between two unlimited intellects would proceed 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!**
  - **If f can be found, it can be used to understand the internal structure of the game.**
- **Remarks: A deterministic game always has a perfect evaluation function, but a stochastic game may or may not has one since each position can have a non-0/1 winning probability.**

Approximate evaluation 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.**
  - **Remark: this refers to deterministic games in the original paper.**
- **Factors considered in approximate evaluation functions:**
  - **The relative values of differences in materials.**
  - **Position of pieces.**
    - ▷ *Mobility: the freedom to move your pieces.*
    - ▷ ...
  - **Pawn structure: the relative positions of the pawns.**
  - **King safety.**
  - **Threat and attack.**
  - ...

Material values

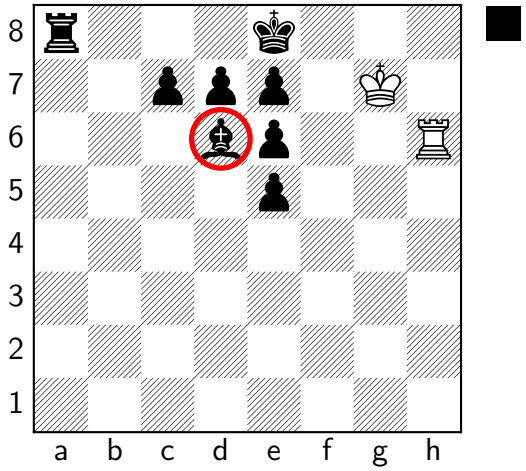
- **The relative values of differences in materials.**
  - **The values of queen, rook, bishop, knight and pawn are about 9, 5, 3, 3, and 1, respectively.**
  - **Note: the value of a pawn increases when it reaches the final rank which it is promoted to any piece, but the king, the player wants.**

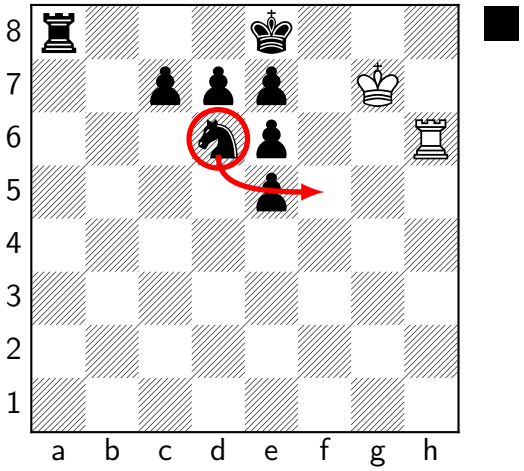


- **Q:**
  - **How to determine good relative values?**
  - **What relative values are logical?**
  - **Static values verse dynamic values.**

Dynamic material values (1/2)

- **In the following example, a black knight ($d6$) on the right is more useful than a black bishop ($d6$) on the left.**
  - **The knight can make a move $d6 - f5$ so that the potential threat from the white is neutralized.**
  - **Note: Black can also alternatively use the right of castling if available.**



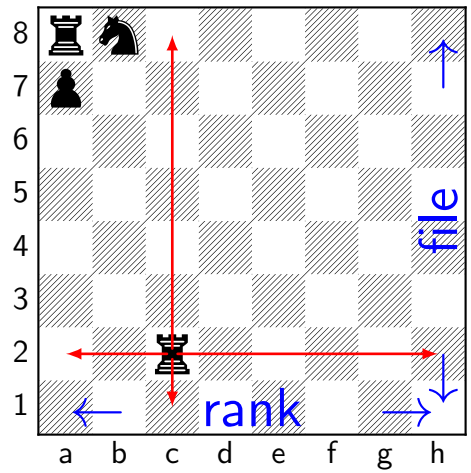


Dynamic material values (2/2)

- **In Chinese chess, the sums of static material values of (rook, knight, cannon), (rook, knight, knight) and (rook, cannon, cannon) are about equal.**
- **However, the dynamic values are different as (rook, knight, cannon) $>$ (rook, knight, knight) $>$ (rook, cannon, cannon).**

Positions of pieces (1/2)

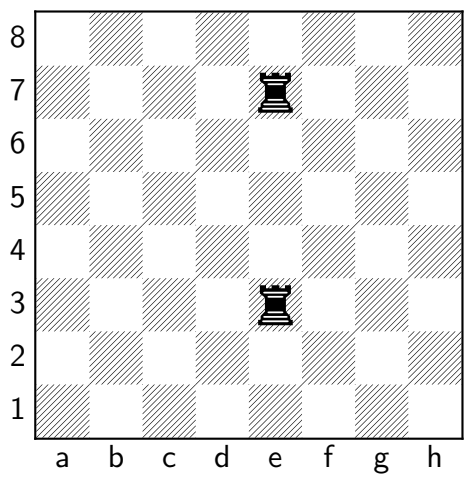
- **Mobility: the amount of freedom to move your pieces.**
  - **This is part of a more general principle that the side with the greater mobility, other things equal, has a better game.**
  - **Example: the rook at $a8$ has a poor mobility, while the rook at $c2$ has a good mobility and is at the 7th rank.**



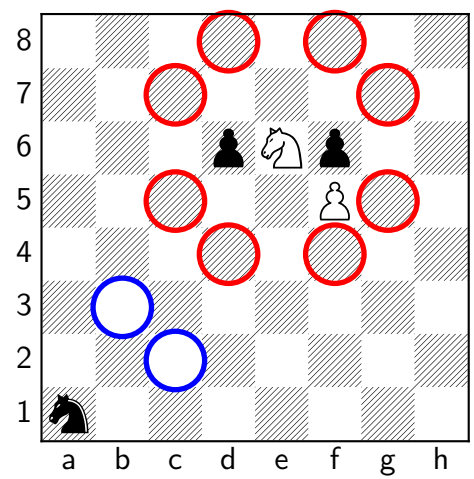
- **Note: file means column ($a \sim h$), rank means row ($1 \sim 8$).**
  - **Count the rank number from your side.**
    - ▷ ***Black rook on $c2$ is on the 7th rank.***
    - ▷ ***White rook on $c2$ is on the 2nd rank.***

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.**



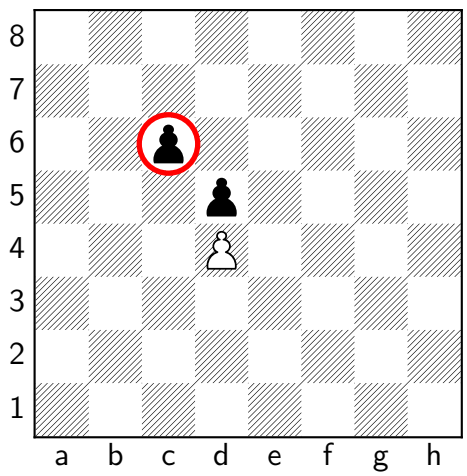
Doubled rooks.



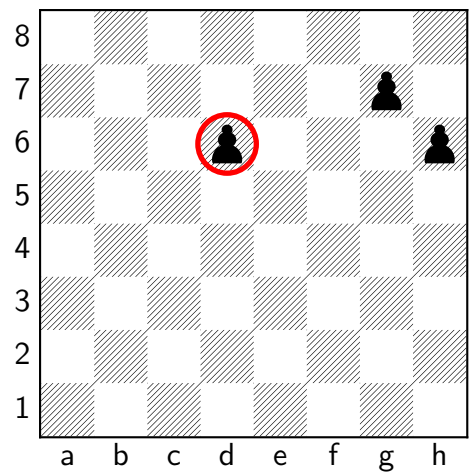
Knight at the center.

Pawn structure (1/2)

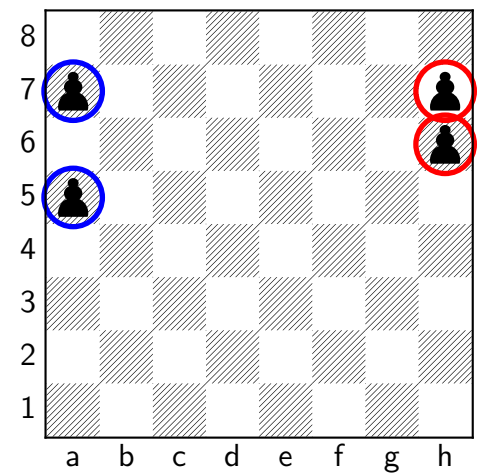
- **Example: Backward, isolated and doubled pawns are weak.**
  - ▷ ***Backward pawn:*** *a pawn that is behind the pawn of the same color on an adjacent file that cannot advance without losing of itself.*
  - ▷ ***Isolated pawn:*** *A pawn that has no friend pawn on the adjacent file.*
  - ▷ ***Doubled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



Backward pawn at $c6$.



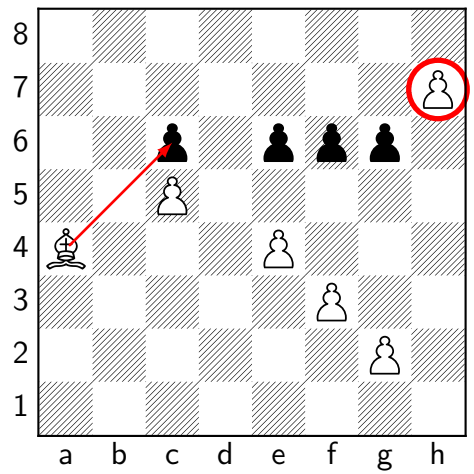
Isolated pawn at $d6$.



Doubled pawns at the a and h columns.

Pawn structure (2/2)

- **Absolute positional information:**
  - **Relative control of centre, for example, white pawns at $d4$ and $e4$.**
- **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.**



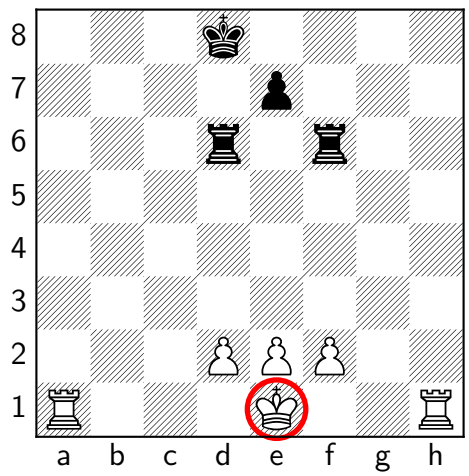
Passed pawn at $h7$ and a black pawn at $c6$ that is attacked by a white bishop.

Pawn structure: comments

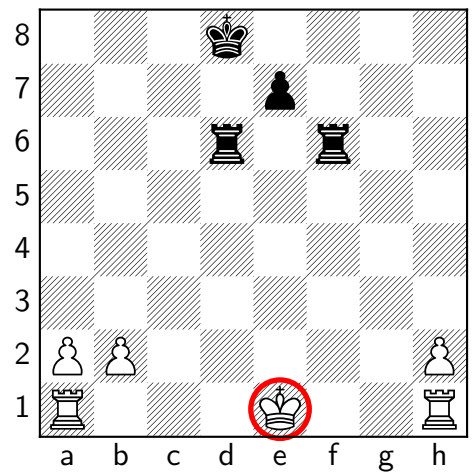
- **For lower-valued pieces,**
  - **forming defensive pattern such that**
    - ▷ ***guarding each other, and***
    - ▷ ***blocking opponent's movements.***
- **Assume q_1 is guarded by q_2 and is under attack by p.**
  - **If q_1 is captured by p, then q_2 can capture p.**
  - **The value of p is often more than that of q_1.**
  - **The value of q_2 can be small, such as it is a pawn.**

King safety

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



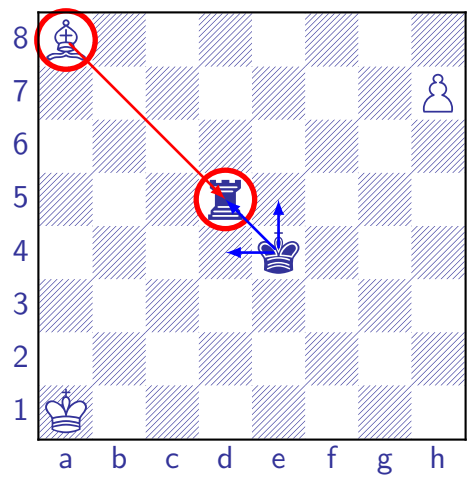
A protected king.



An exposed king.

Threat and attack

- **Attacks.**
  - ▷ *Attacks on pieces which give one player an option of exchanging; for example, the white bishop at $a8$ is attacking the rook at $d5$.*
  - ▷ *Attacks on squares adjacent to king.*
- **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 $d5$.*
- **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 evaluation function

- **Most chess and chess-like programs use approximate evaluation 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 evaluation 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.**
    - ▷ *Can be non-linear combinations such as the values of pawns depend on whether opponent's king is still alive in Chinese Dark Chess (CDC).*
  - **May need to consider different evaluation functions during open-game, middle-game and end-game.**
  - **Can use hyper-parameter optimization techniques from machine/deep learning to better fine tune the coefficients.**
    - ▷ *Example: Gradient descent, grid search ...*
- **The right to move next, namely turn, has a place in f.**
  - **Given the same arrangement of pieces on the board, the value of f may depend a lot on who the next player is.**

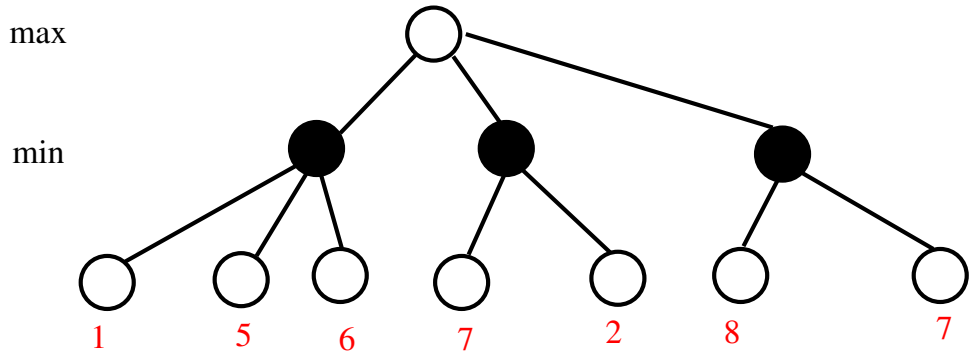
Strategy based on an evaluation function

- **A max-min strategy based on an approximate evaluation 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**

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

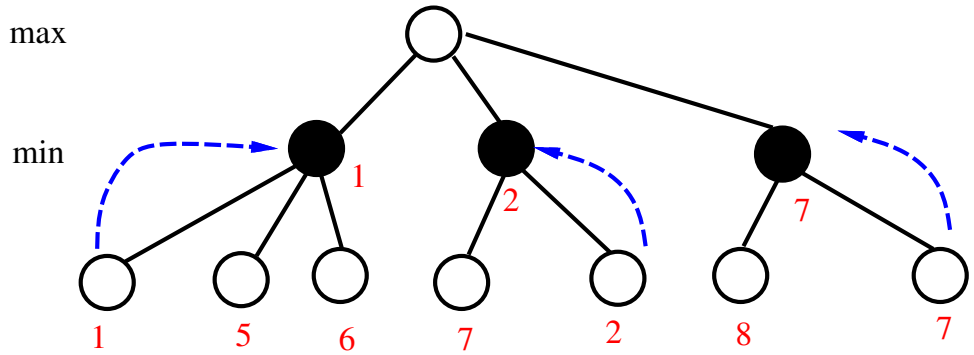
    where $next(p)$ is the positions that p can reach in one ply.
  - **Can be extended to a strategy with more moves.**

Mini-max formulation



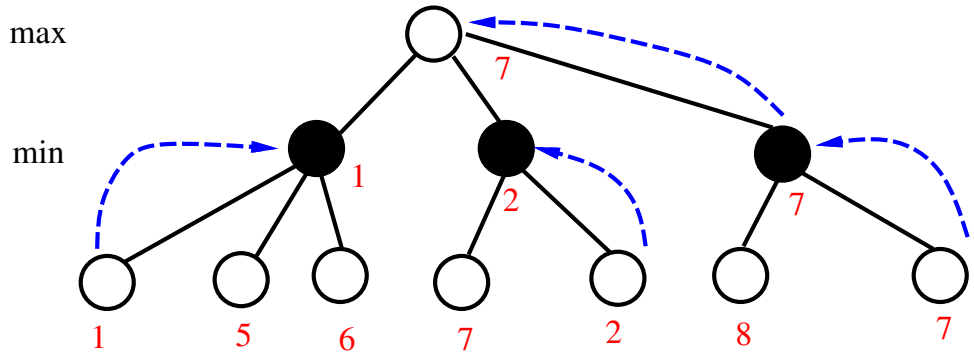
- **Mini-max formulation:**
  - $$F'(p) = \begin{cases} f(p) & \textbf{if } b = 0 \\ max\{G'(p_1), \ldots, G'(p_b)\} & \textbf{if } b > 0 \end{cases}$$
  - $$G'(p) = \begin{cases} f(p) & \textbf{if } b = 0 \\ min\{F'(p_1), \ldots, F'(p_b)\} & \textbf{if } b > 0 \end{cases}$$
  - **An indirect recursive formula with a bottom-up evaluation!**
  - **Equivalent to AND-OR logic.**

Mini-max formulation



- **Mini-max formulation:**
  - $$F'(p) = \begin{cases} f(p) & \textbf{if } b = 0 \\ max\{G'(p_1), \ldots, G'(p_b)\} & \textbf{if } b > 0 \end{cases}$$
  - $$G'(p) = \begin{cases} f(p) & \textbf{if } b = 0 \\ min\{F'(p_1), \ldots, F'(p_b)\} & \textbf{if } b > 0 \end{cases}$$
  - **An indirect recursive formula with a bottom-up evaluation!**
  - **Equivalent to AND-OR logic.**

Mini-max formulation



- **Mini-max formulation:**
  - $$F'(p) = \begin{cases} f(p) & \textbf{if } b = 0 \\ max\{G'(p_1), \ldots, G'(p_b)\} & \textbf{if } b > 0 \end{cases}$$
  - $$G'(p) = \begin{cases} f(p) & \textbf{if } b = 0 \\ min\{F'(p_1), \ldots, F'(p_b)\} & \textbf{if } b > 0 \end{cases}$$
  - **An indirect recursive formula with a bottom-up evaluation!**
  - **Equivalent to AND-OR logic.**

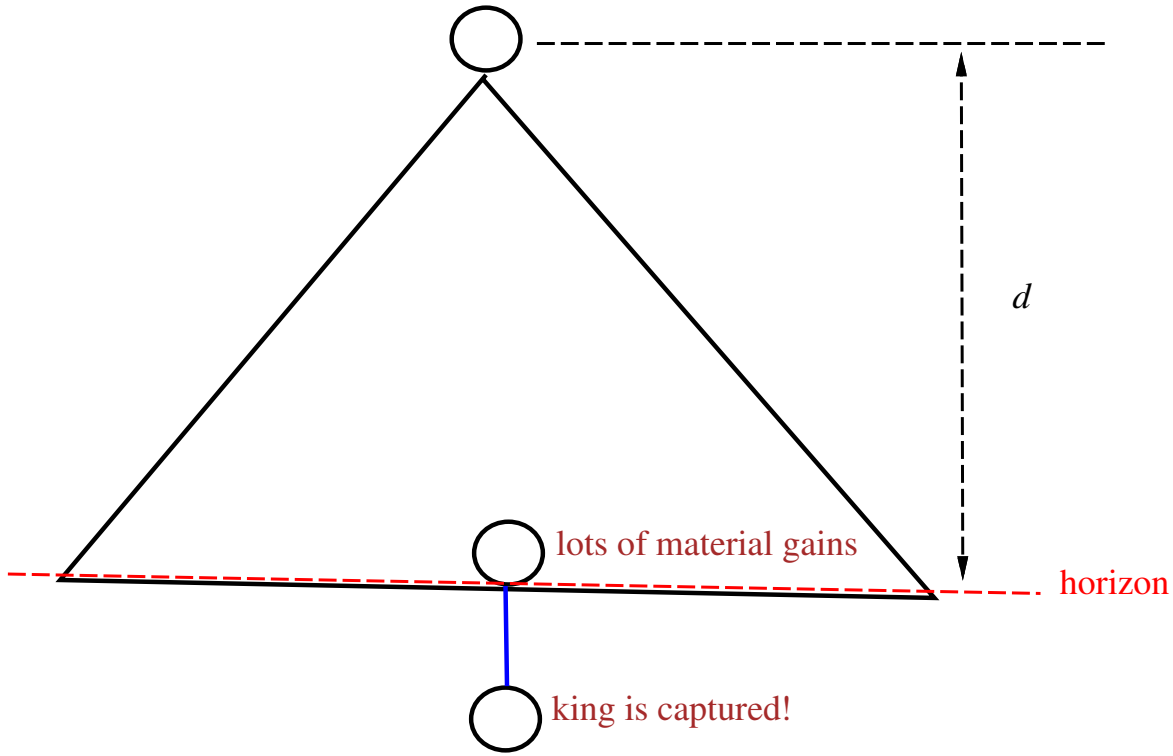
Comments to strategy

- **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.**

Quiescent search

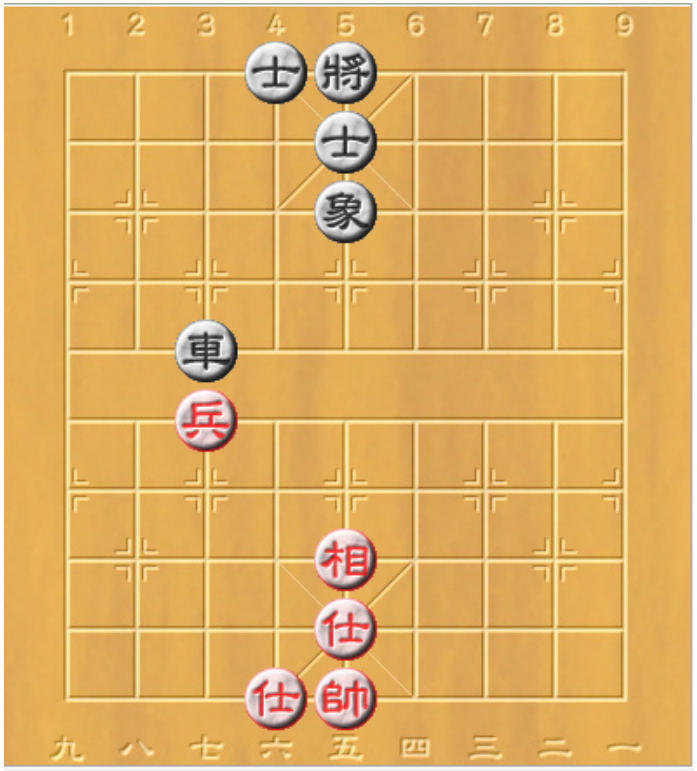
- **A simple type of evaluation function can be only applied in relatively quiescent positions.**
  - **Positions that are not in the middle of material exchanging (swap).**
  - **Positions that are not being checked.**
  - **Positions that are not in the middle of a sequence of moves with little choices.**
- **To solve the horizon effect.**
  - **Bad cases happen when the searching depth is fixed.**
- **When a position is not quiescent: apply quiescent search.**
  - **Generating selected moves and to reach positions that are quiescent.**
  - **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 can be 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.*
- **More details will be given in later lectures.**

Horizon effect



Quiescent search: example

- **Example: red pawn will capture black rook in red's turn, but black rook will not capture red pawn in black's turn.**
  - **If the red to move next, then it is draw.**
  - **If the black to move next, then black has an advantage.**

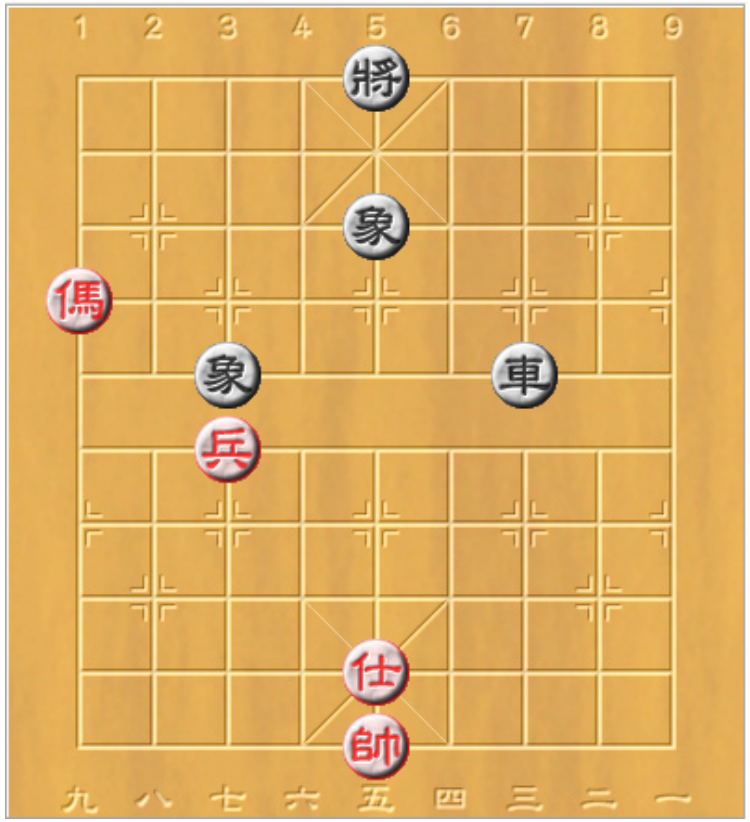


Algorithm SEE($location$)

- **Assume w.l.o.g. it is red's turn and there is a black piece bp at $location$.**
- **Algorithm SEE($location$)**
  - $R :=$ **the list of red pieces that can capture a black piece at** $location$**.**
  - **if** $R = \emptyset$**, then return 0;**
  - **Sort** R **according to their material values in non-decreasing order.**
  - $B :=$ **the list of black pieces that can capture a red piece at** $location$**.**
  - **Sort** B **according to their material values in non-decreasing order.**
  - $gain := 0$; $piece := bp$;
  - **While** $R \neq \emptyset$ **do**
    - ▷ ***capture*** $piece$ ***at*** $location$ ***using the first element*** w ***in*** R***;***
    - ▷ ***remove*** w ***from*** R***;***
    - ▷ $gain := gain + value(piece)$***;***
    - ▷ $piece := w$***;***
    - ▷ ***if*** $B \neq \emptyset$
      then **{** ***capture*** $piece$ ***at*** $location$ ***using the first element*** h ***in*** B***;***
      remove h ***from*** B***;*** $gain := gain - value(piece)$***;*** $piece := h$***;*** **}**
      else break;
  - **return** $gain$ **//the net gain of material values during the exchange**

SEE: Example

- **Net gain in red's turn.**
  - **Captured: two black elephants**
  - **Be captured: a pawn and a knight.**
  - **Not a good deal for the red to do a piece exchanging.**

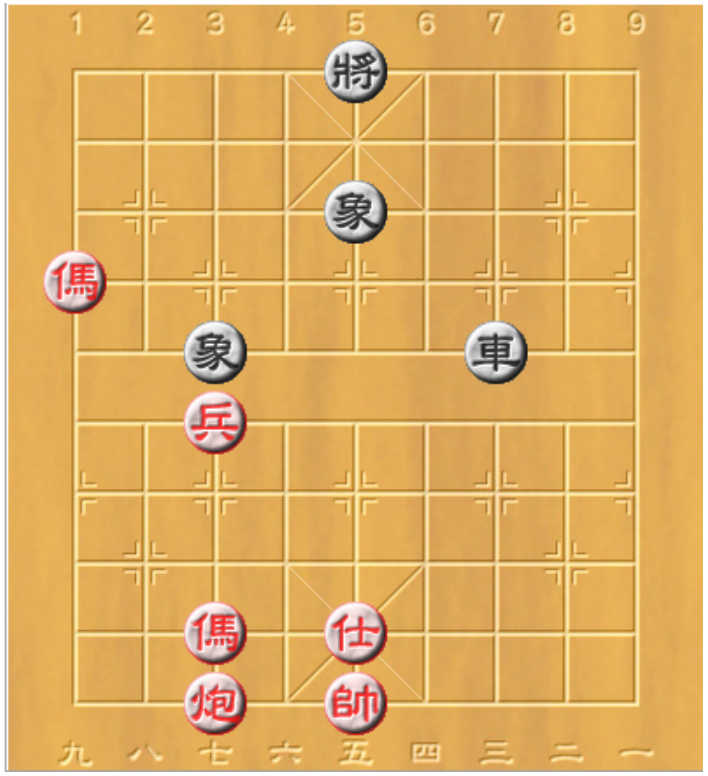


SEE: Comments

- **We carry out a capturing move in Quiescent search only if the net gain is positive.**
  - **Currently, we only consider material gains.**
  - **May lose the right to move next, which obviously has a place in the evaluation function, after the material exchange.**
- **Always use a lower valued piece to capture if there are two choices to get the best gain.**
  - **This only works for capturing equivalent games.**
  - **For games of other types, such as Chinese Dark Chess, you always use the highest ranked piece to capture if possible.**
- **SEE is designed to be static and imprecise for performance issues.**
  - **Some pieces may capture or not be able to capture a piece at a location because of the exchanges carried out before.**
  - **If SEE considers more dynamic situations, then it costs more time.**
    - ▷ ***It becomes searching one step deeper.***

Counter example of SEE

- **Red cannon can attack the location where the black elephant was at the river after red pawn captures black elephant, and then the black elephant captures the red pawn.**

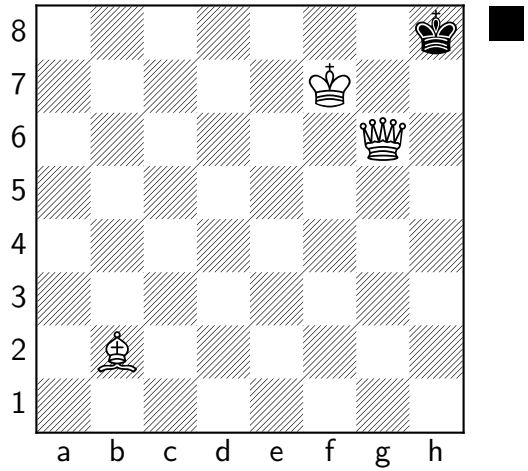


Methods of winning

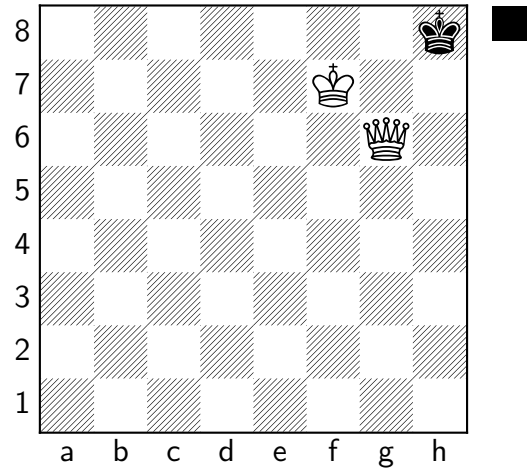
- **Methods of winning**
  - **Checkmate.**
    - ▷ *The king is in check and it is in check for every possible move.*
  - **Stalemate is draw in western chess.**
    - ▷ *Winning by making the opponent having no legal next move.*
    - ▷ *In Western Chess, a suicide move is illegal, and stalemate results 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.
  - **Stalemate is win in Chinese dark chess and many other games.**
    - ▷ *Capturing all pieces of the opponent.*
    - ▷ *Blocking all pieces of the opponent.*

Example: Checkmate and Stalemate

- **Checkmate for the bottom left position: it is in check and remains to be in check for every possible move.**
- **Stalemate for the bottom right position.**
  - ***A stalemate is one that is not in check, but will be in check for every possible move.***
  - $h8 - h7$, $h8 - g7$ ***and*** $h8 - g8$ ***are all suicide moves.***



Checkmate if black is to move.



Stalemate if black is to move.

More on Stalemate

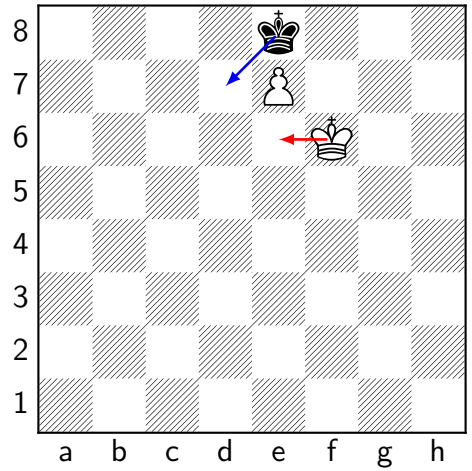
- **Many games such as Chinese chess and Chinese dark chess consider stalemate as win.**
- **The reason of making "stalemate" as draw in Chess has a long history and was first written into London Chess Club laws of 1807.**
  - **Stalemate should be penalized for having the game "come to its end before the arrival of the regular result." [A. Saul, 1614, The Famous game of Chessplay] — the first book in English on Chess.**
- **In game playing, you want to design rules so that**
  - **it is more difficult to win than to draw;**
  - **you wish to encourage "meaningful" plays of a game.**
    - ▷ *Discourage easily falling into a loop.*
    - ▷ *Discourage playing without progressing for a long time such as making safe moves in Chinese dark chess.*
- **Now is a debatable issue.**
  - **Some variations of Chess make stalemate as half-win [E. Lasker, 1926, Mein Wettkampf mit Capablanca].**

Special configurations

- **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.*
- **Using the right of pass, i.e., to make no move in one turn, sometimes is a special way to protect yourself in other games such as Go.**

Example: Zugzwang

- **Example: White to move is draw, while it will soon be a black loss if black is to move next.**



Zugzwang for the black.

- **Discussion**
  - **White to move**
    - ▷ $f6 - f7$: *illegal move.*
    - ▷ $f6 - e6$: *stallmate.*
    - ▷ $f6 - (e5, f5, g6, g7)$: *black then captures the pawn and ends in a draw.*
  - **Black to move**
    - ▷ $e8 - d7$: *the only legal move and then white to advance pawn for a* ***promotion****.*

More on 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.*
- **Advanced data structure: bit-boards [Browne 2014] which will be covered in later lectures.**
  - **Using a bit stream to represent a position.**
  - **Bit streams for pieces of different colors or kinds.**
  - **Bit-wise operations, such as PEXT, are used to maintain these bit streams.**
- **Evaluation 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.**
  - **Un-capturing.**
  - **Capturing.**
    - ▷ *Capture opponent's most valued pieces (MVP) first.*
    - ▷ *Use my least valued piece (LVP) to capture.*
  - **Consider piece moving in some order of importance.**
  - ...

Move ordering: CDC

- **Chinese dark chess (CDC) has a special capturing rule: capturing asymmetric.**
- **A better move ordering may need to consider:**
  - **Capture opponent's most valued pieces (MVP) first.**
    - ▷ *If you have the option of capturing using same type of pieces, then capture the most valued such one.*
  - **When an opponent piece can be captured by several different my pieces, yse the following strategy:**
    - ▷ *Use the same ranked piece first.*
    - ▷ *If it can be captured by cannon, then check whether the cannon can be captured by the opponent later.*
    - ▷ *In computing SEE, only the largest ones plus cannons in the R and B lists matter.*

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.*
- **SEE is used before initializing a sequence of piece exchanging or swaps.**
- **Move-ordering and other techniques are used 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 owning 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 evaluation function and the other modules.**
  - **A starting point of using an "adapted" strategy which later becomes "machine learning".**
  - **Arthur Samuel in his 1967 paper first used the phrase "machine learning" which is believed to follow Shannon.**
    - ▷ *A. Samuel. Some studies in machine learning using the game of checkers.* IBM J. Res. Develop., *3:210–229, 1959.*
- **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.*, 3:210–229, 1959.
- Browne, C. Bitboard Methods For Games. *ICGA* Journal, 37(2), 67-84, 2014. https://doi.org/10.3233/ICG-2014-37202
- London Chess Club laws, 1807.
- A. Saul. The Famous game of Chessplay 1614.
- E. Lasker. Mein Wettkampf mit Capablanca 1926.
- Müller, Karsten and Haworth, Guy. Stalemate and "DTS" Depth to Stalemate Endgame Tables'. ICGA journal, 1 Jan. 2019 : 191 – 199.