GAMES COMPUTERS PLAY

A bit of History and Some Examples

Early History

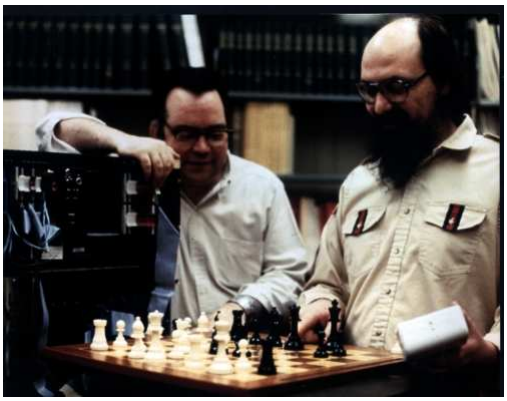
- Checkers is the game for which a computer program was written for the first time. Claude Shannon, the founder of Information Theory, wrote such a program and reported the results in 1950.
- The program tried to imitate human playing strategies.

Chess

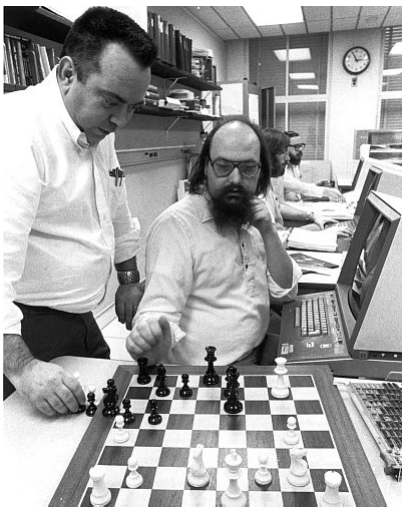
- For many years researchers used a knowledge-based search that tried to make chess programs recognize patterns and formulate chess strategies much like people do.
- Around 1977 Ken Thompson had the idea to use what computers do best – fast calculation – rather than trying to get them to imitate human thinking. The idea was to conduct an exhaustive search.

Belle -A

- With the help of Joe Condon he developed a custom chess-playing computer called ***Belle***.
- *Belle* won the American computer chess championship in 1978 and the World chess championship in 1980. It achieved master level rating against human players.
- *Belle* defeated computer programs running on super-computers that were much faster than the PDP machine that was the basis of *Belle*.



Ken Thompson and Joe Condon at Bell Labs around 1980.



Ken Thompson and Joe Condon at Bell Labs around 1980

Belle - B

- The machine is now at the Smithsonian Museum. Additional information about *Belle* (and other chess playing computer programs) can be found at the web site of the Computer History Museum: http://www.computerhistory.org/chess/
- Typing *Belle* in the search box of the pages returns all the relevant pages, including a link to an interview with Ken Thompson that provides the oral history of *Belle*.

Belle - C

- The most pertinent page is: http://www.computerhistory.org/chess/main.php?sec=thm-42eeabf470432&sel=thm-42f15c52333a3#
- Here we go!

The Consequences of Belle

- The success of Belle was truly a “game changer.”
- Most researchers gave up on trying to write programs that imitate the way humans play and focused instead on the exhaustive search approach.
- The IBM Deep Blue that defeated the human world chess champion was based on the same principles as Belle.

What comes up next

- An illustration of the exhaustive search approach using the simple game of Tic-Tac-Toe.
- Checkers
- More on Chess

TIC-TAC-TOE

Games\TicTacToe\tictactoe.exe

A Computer can play a game without understanding it!

- The computer looks ahead at all possible moves and then backtracks marking the sequence of moves for the best possible result.
- For Tic-Tac-Toe that is an overkill but we will use this game for illustration because:
- It is the strategy used in Checkers and Chess programs!

The Army of Ants Strategy - 1

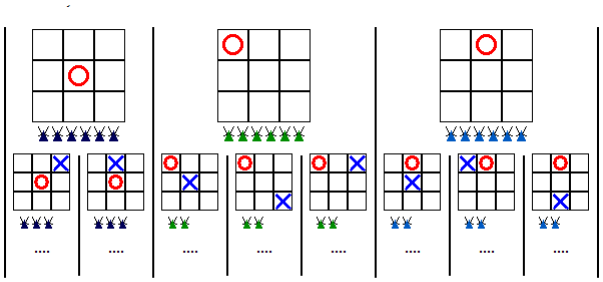


Figure 3.1.1: An army of ants following different plays for the game of Tic-Tac-Toe. We show only a few of the possible plays.

The Army of Ants Strategy – 2

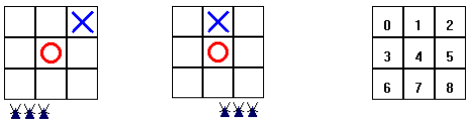


Figure 3.2.2: (a) and (b) The two possible responses to the computer playing the center. The army of ants has divided itself to follow the plays resulting from each of the two positions. (c) Numbering of the squares of the board for reference in the play descriptions.

The Army of Ants Strategy – 3

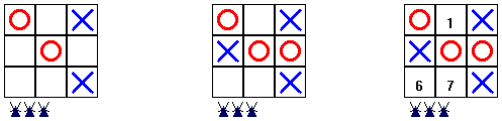


Figure 3.2.3: Possible plays following the position of Fig. 3.2.2a.In this case the computer must play position 5 and that, in turn, forces the opponent to play position 3 with the result shown in Fig. 3.2.3b. That leaves three possible plays marked with their numbers in Figure 3.2.3c. Position 7 is clearly better because it blocks the opponent from lining up in the bottom row and the game must end in a draw.

The Army of Ants Strategy - 4

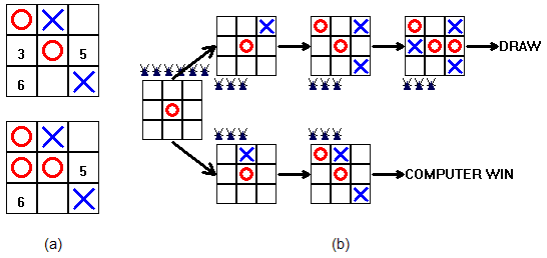


Figure 3.2.4: (a) Follow-up of the arrangement of Figure 3.2.2b. Playing at 3 ensures a win for the computer. (b) Graph of possible plays following a play at the center.

The Army of Ants Strategy - 5

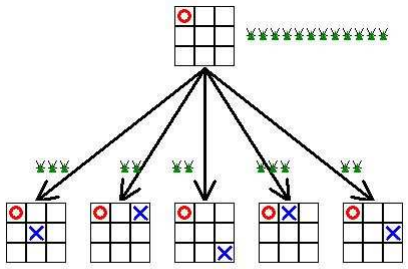


Figure 3.2.5: Alternative moves when the computer plays at a corner.

The Army of Ants Strategy - 6

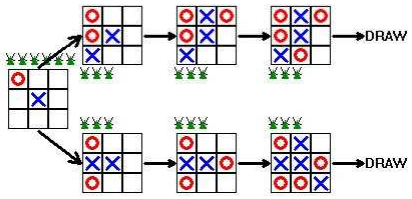


Figure 3.2.6: The graph of sequences of plays when the computer plays first at a corner and its opponent takes the center.

Conclusions from the Search

- **First move**: If the center is open play there. (It may lead to a win and it will do no worse than a draw according to Fig. 3.2.4.) Otherwise play at a corner. (It will do no worse than a draw according to Fig. 3.2.6.)
- **Next move after first move was at the center:** If the opponent plays at a corner follow the moves shown in the upper path of Figure 3.2.4b. Otherwise follow the moves shown in the lower path of Figure 3.2.4b.
- **Next move after first move was at the corner:** Depending upon the opponent's play follow the moves shown in one of the paths of Figure 3.2.5.

The Program - 1

- The squares of the board are numbered from 0 to 8, line this:
  012
  345
  678
- The horizontal rows are (0, 1, 2), (3, 4, 5), and (6, 7, 8). The vertical columns are (0, 3, 6), (1, 4, 7), and (2, 5, 8). The diagonals are (0, 4, 8) and (2, 4, 6).
  For brevity we will refer to a row, column, or diagonal as a **triad**.

The Program - 2

- The program keeps track of location of each square as well as a whether a square is empty, is occupied by the HUMAN player or the COMPUTER player.
- If the computer plays first, it plays on one of the four corners (0, 2, 6, 8) or the center (4). The choice is made by rolling (the equivalent of) a five-sided dice. If it plays second and the first human play is in one of the corners (0, 2, 6, 8), it plays on the opposite corner (8, 6, 2, 0).
- If not it proceeds to play according to the rules below. (This part is suppressed if the flag EASY has been set.)

The Program - 3

- When the user presses the left mouse button the program finds the square pointed by the user, fills it and then it takes the following steps.
- Step 1: It checks if the human has completed a triad in which case it declares the human the winner and ends the game.
- Step 2: It checks if the board is full and ends the game without a winner.

The Program - 4

- Step 3: It checks whether there is a triad with two elements filled by the same player and the third empty. The computer plays the empty element either blocking the human player or winning. Then it goes to step 7. If there is no such move the program continues to step 4.
- Step 4: If the center square is empty the computer occupies it and goes to step 7. Otherwise it continues to step 5.

The Program - 4

- Step 5: The programs looks for an empty corner (0, 2, 6, 8) and if there is an empty space, it takes it and goes to step 7, otherwise it continues to step 6.
- Step 6: The programs looks for an empty middle (1, 3, 5, 7) and if there is an empty space, it takes it.

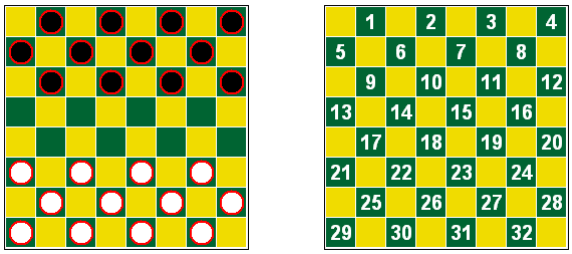
The Program - 5

- Step 7:The program checks if the computer has won. If it has, the program ends the game.
- Step 8: The program checks if the board is full, in which case it ends the game. Otherwise it waits for the human to play.

Let's play TIC-TAC-TOE

Games\TicTacToe\tictactoe.exe

CHECKERS



Modern History

- The modern era of the computer checkers, based on exhaustive search strategies, dates from 1989 when *J. Schaeffer* (University of Alberta in Canada) started the ***Chinook*** project.
- The result was a computer program that by 1996 could defeat any human player. Once this goal was achieved, the next challenge was to find the perfect strategy for checkers.

Finding the Perfect Strategy - 1

- There are about $5*10^{20}$ possible moves for checkers, a huge number but much smaller than that of possible chess moves. The number of moves to be examined can be reduced by eliminating duplicate positions. (For example, the sequence of moves 9-14, 23-19, 11-15, and 22-17 leads to the same board configuration as the moves 11-15, 22-17, 9-14, and 23-19.)

Finding the Perfect Strategy - 2

- If we use a scoring system to estimate how good a position, then we need not examine clearly losing positions and that reduces the possibilities even more. (For the mathematically brave: this is called the *alpha-beta pruning* strategy.)
- The reduced number of positions to be examined is about 10^{14}.

Finding the Perfect Strategy - 3

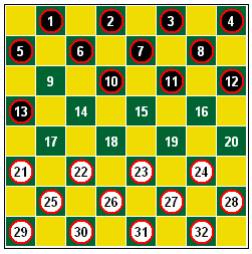
- If a program spends one millisecond (10^{-3}) per move, examining the 10^{14} possible moves would require 10^{11} seconds. There are about $3*10^{7}$ seconds in a year.
- Thus the time to examine all possible moves would be about $3*10^{3}$ or three thousand years. If we spread the work amongst 1000 computers that will take only three years.

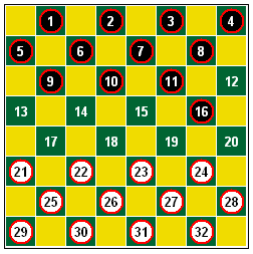
Finding the Perfect Strategy - 4

- This is roughly what happened in the *Chinook* project. They used up to 200 computers at one time and the project took about 15 years to complete.
- An initial effort took seven years (1989-1996) and the project was halted. By 2001 computer speed had improved so much that the task that had taken seven years could then by done in a month. The project was resumed and it was completed by 2004.

Finding the Perfect Strategy - 5

- One of the results was that the opening move 9-13 guarantees a draw if subsequent moves are chosen optimally, no matter how well the opponent plays.
- The worst opening move, one that results in a loss if the opponent plays optimally, is 12-16.





The best (9-13, left) and the worst (12-16, right) opening moves in checkers

Some Checker Playing Programs

Chinook http://webdocs.cs.ualberta.ca/~chinook/

Nemesis http://www.nemesis.info/

Sage http://homepages.tcp.co.uk/~pcsol/sage.htm

Cake http://www.fierz.ch/cake.php

The Challenge for Chess

- At first sight using the exhaustive search strategy for chess seems utterly impossible.
- Before we start counting, a definition:
- The term **ply** is used to denote a player's turn while the word **move** is used to denote a pair of turns. So if white opens with "pawn to King's 4" and black reply is "pawn to Queen's 3" the first move is "white: pawn to King's 4/ black:pawn to Queen's 3".

Counting Moves

- For chess there are 20 possible values for the first ply and another 20 for the second, so that there are 400 possible values for the first move. The number of values in subsequent moves can be greater or smaller than 400 but for our present purpose will suffice to note that there at least 100 possibilities per move. For 30 moves that is 100^{30} or 10^{60}.

Some Huge Numbers

- If it takes only one second to create a move we will need 10^{60} seconds to create all possible combinations for 30 moves. There are about $3*10^7$ seconds in year, so dividing 10^{60} by this number we find that we will need $(1/3)*10^{53}$ years, truly an eternity. (Note that a trillion is 10^{12}.)

Even at the speed of light it will take an eternity

- Suppose that we program a machine to calculate all possible combinations of moves and that it could need one nanosecond per move. That would speed up the process by a factor of the 10^9.
- Putting a million such machines to work in parallel will speed up the process by an additional factor of 10^6 for a total speed up of 10^{15}. This will reduce the total time from $(1/3)*10^{53}$ years to $(1/3)*10^{38}$ years. That is still much longer than a trillion years.

Belle - 1

- Ken Thompson's idea was to limit the number of moves for the exhaustive search and then select the initial move that looked most promising from the results of the exhaustive search after, say, four moves.
- He worked with *Joe Condon* to build special purpose hardware for generating and evaluating chess moves and they called their machine ***Belle***. It could examine 160,000 positions per second.

Belle - 2

- To look ahead by four moves requires examining roughly 400^4 positions. That is approximately $25*10^9$ or 25 billion positions. Examining all such positions it will take about 156,000 seconds or about two days.
- That is too long for the time allotted in chess tournament play. It turns out that one can reduce the number of moves to be examined if a move results in a worse position than another move.
- By using this strategy the number of moves to be examined was reduced by a factor of more than 1000 allowing *Belle* to look ahead four moves in a few minutes.

Belle - 3

- Because Belle looked only four moves ahead, it needed some chess knowledge to evaluate which result was the best.
- Note that the checkers program does not need any such knowledge because it completes the exhaustive search.
- In addition Belle used a different strategy for the end game.

Children of Belle

- In the 1980s, two competing computer chess machines, named *Hitech* and *ChipTest*, emerged from Carnegie Mellon University. They were both based on the same principles as *Belle* but used more advanced hardware.
- The *ChipTest* team developed a second machine, named *Deep Thought.* In 1988 both machines defeated human opponents at the grandmaster level.

Deep Blue

- In 1989, IBM hired three of the *Deep Thought* team members (Feng-Hsiung Hsu, Murray Campbell and Thomas Anantharaman) to develop a computer that would beat reigning World Chess Champion Garry Kasparov.
- The result of the work was the ***Deep Blue*** machine that defeated Kasparov in May 1997.
- http://www.computerhistory.org/chess/main.php?sec=thm-42f15cec6680f&sel=thm-42f15d3399c41

A Stony Brook connection

- In 1997 a Stony Brook team won the regional New York programming contest sponsored by ACM (Association for Computer Machinery).
- That qualified the team to travel to Atlanta in 1998 to take part in the world programming contest.
- IBM sponsored the contest and the Deep Blue members were there and we had a chance to meet them.

Left: The programming team posing in front of the Computer Science building with their diplomas from their first place finish in the New Yorkregional contest. From the left Dario Vlah, Kok-Ho Loh, Pedro Sanders. **Right:** My Olympic badge!