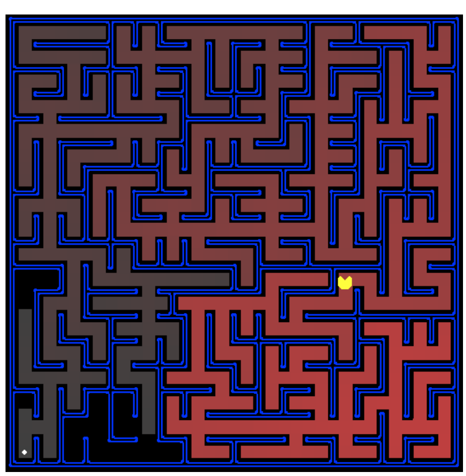
CMSC 373 Artificial Intelligence
Fall 2023
05-Game Playing

Deepak Kumar
Bryn Mawr College

1



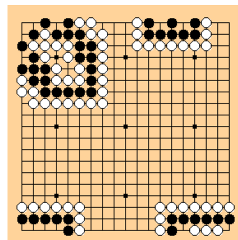
2

2

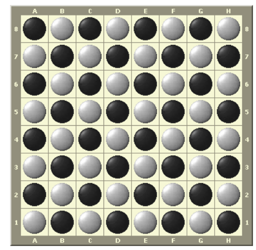
2-Person Games

Checkers

Chess



Go



Konane

Tic Tac Toe

3

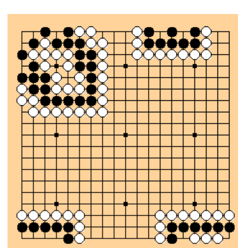
3

2-Person Games

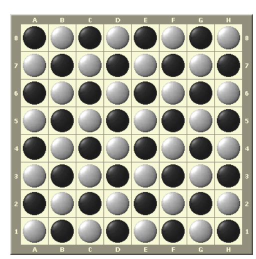
- Perfect Information Game
- Zero Sum Game

Checkers

Chess



Go



Konane

Tic Tac Toe

4

4

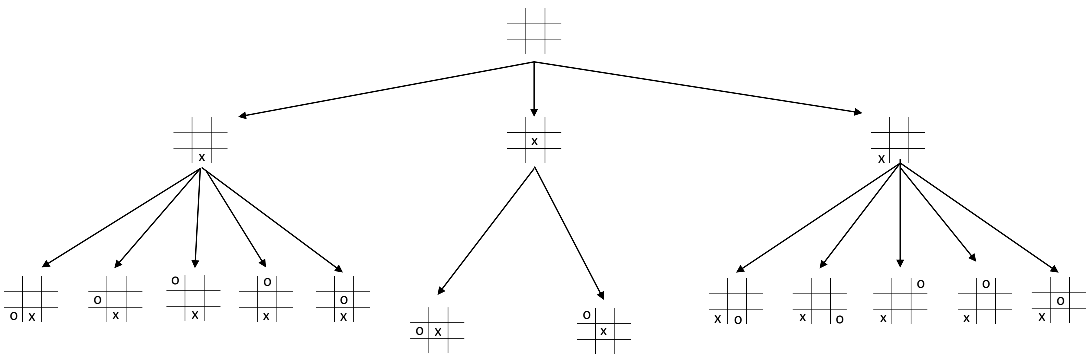
Samuel's Checkers Program, 1950s

Image: https://medium.com/ibm-data-ai/the-first-of-its-kind-ai-model-samuels-checkers-playing-program-1b712fa4ab96

Writing Game Playing Programs

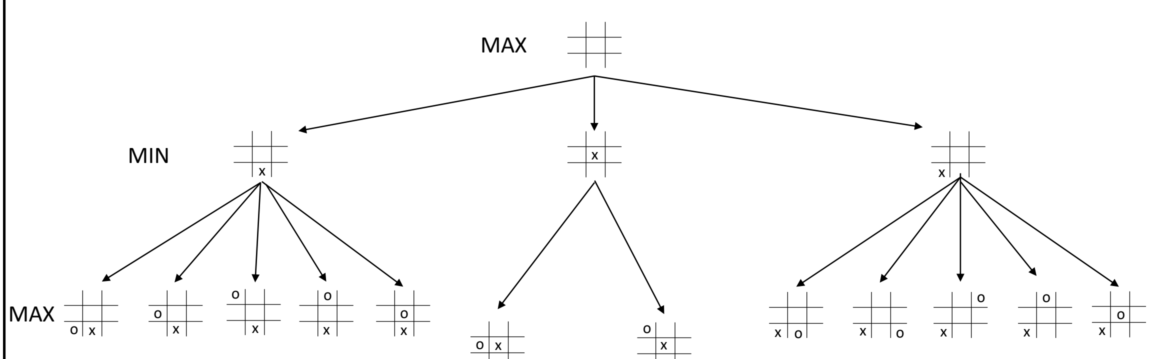
- The base algorithm for most 2-person, zero-sum, perfect information games is called the **Minimax Algorithm**.
- The algorithm explores possible moves the computer can play. And, in response examines possible moves the user can play. This is done for multiple levels. Information gathered from this exploration is then used to decide a move the computer should play.
- This strategy can be used for all 2-person, zero-sum, perfect information games. And, for imperfect information games as well.

Tic Tac Toe – Exploring Moves (depth = 2)



7

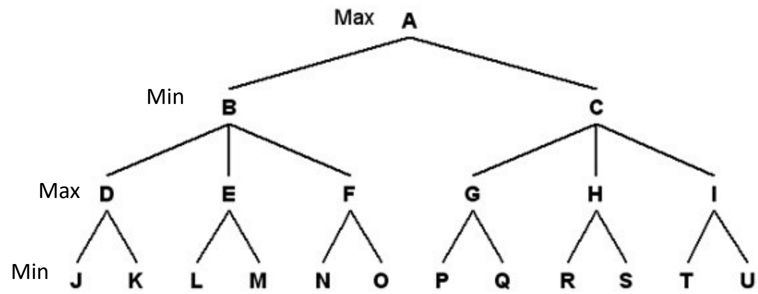
Minimax Search (MAX & MIN levels)



8

Minimax Search Algorithm

- **A..U** represent board positions
- **A** is the current board and it is the computer's turn to move
- The computer has two possible moves, **B** & **C**
- Minimax answers the question: Which is a better move? **B**? **C**?
- If the computer makes move **B**, the user will have the moves **D**, **E**, **F**. Similarly for **C** the user will have **G**, **H**, **I**. etc.
- Minimax explores this as a search tree in a **depth-first** fashion.



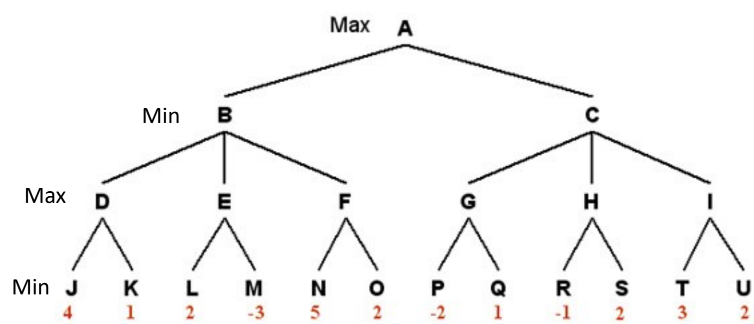
Minimax Search Algorithm

- Upon reaching a limit, the computer does an evaluation of the state of the board (see numbers in red).

  A 4 in board state **J** represents the state of the board. Compared to -3 in board position **M** implies that **J** is a more desirable state than **M**.
- Each level is labelled as **Max**, or **Min**.

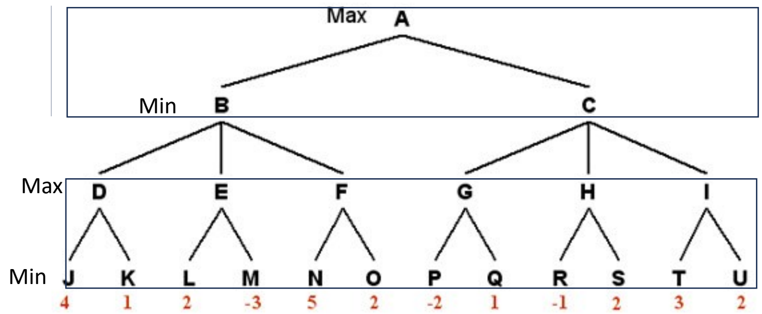
  Max level means the maximum value will be selected (to favor the computer).

  Min level means the minimum value will be selected as it is assumed that the user will always make the worst move possible for the computer.
- Through the search process, these values are **sifted up** the tree to enable the computer to decide about its move.
- **Question:** Where do these values come from???



Game Ply

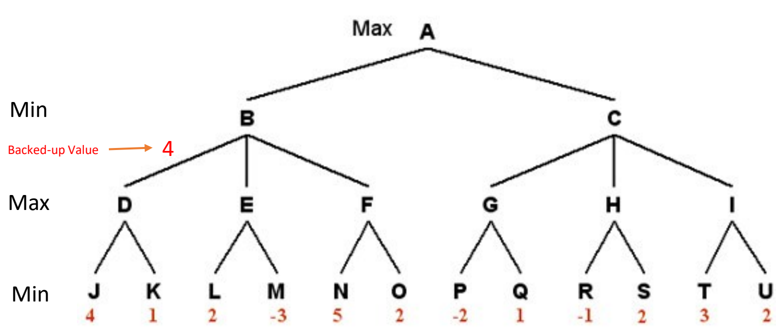
- Each pair of Max and Min levels is called a ply
- Typically, one can specify how many plies to look ahead.
- Typically, more plies searched leads to better moves selected. The longer it takes.



11

11

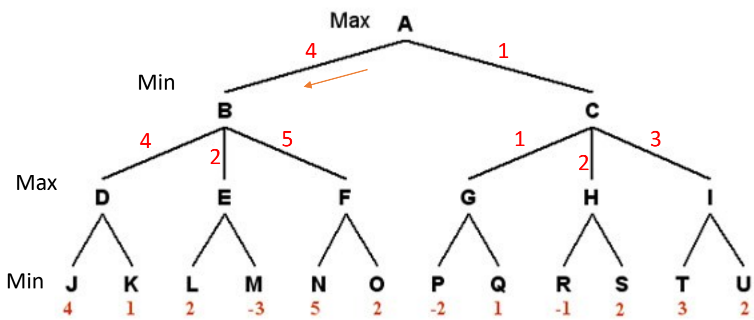
How Minimax Works



12

12

How Minimax Works



13

13

Where do those values come from???

- **Heuristics!!**
- Given a board state, ***b***

  e(b) = an evaluation of state, ***b*** to indicate its goodness for computer

  e(b) is called the **static evaluation function**

  positive values represent a favorable state for the computer
  negative values represent a favorable state for the user

- Heuristics will vary from game to game. From programmer to programmer.

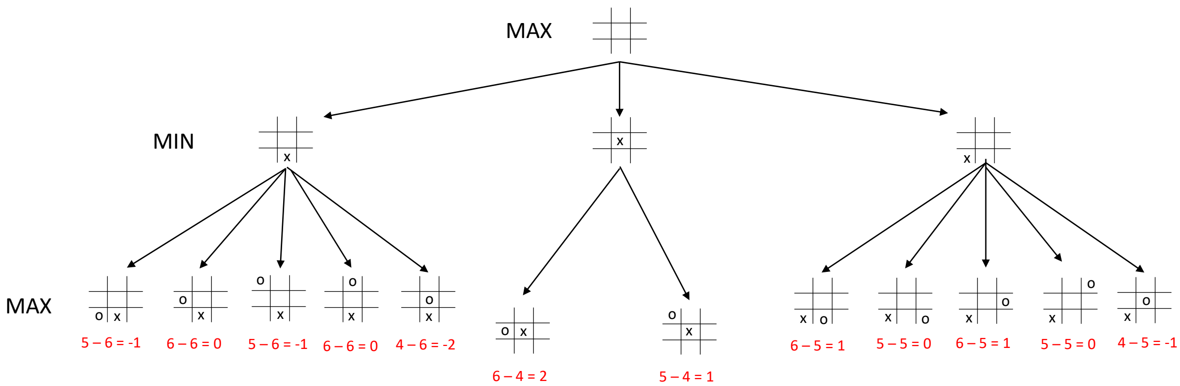
  It is more of an art.
  Quality of the heuristic function determines the quality of Minimax/game

14

14

A Heuristic for Tic Tac Toe

e(b) = [#of open lines for computer (x)] – [#open lines for user (o)]

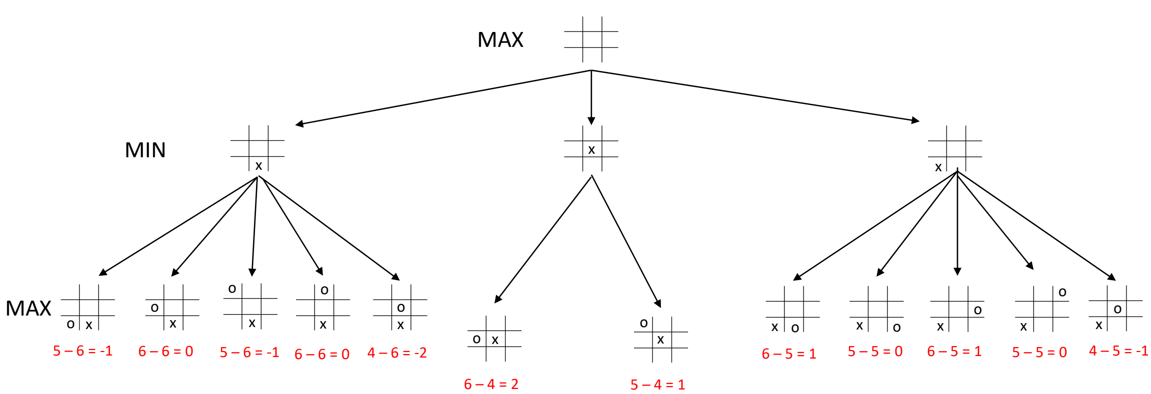


15

15

A Heuristic for Tic Tac Toe

e(b) = [#of open lines for computer (x)] – [#open lines for user (o)]



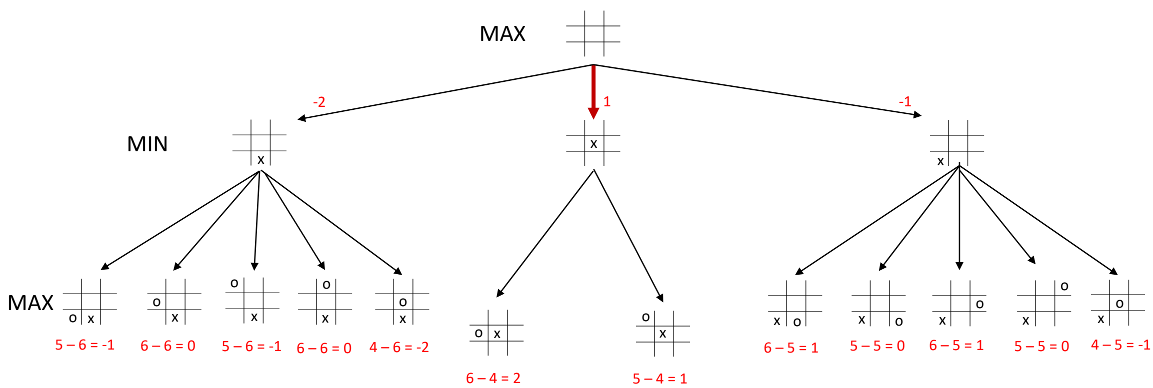
Which move should the computer make???

16

16

A Heuristic for Tic Tac Toe

e(b) = [#of open lines for computer (x)] – [#open lines for user (o)]

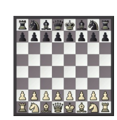


Example Static Evaluation Functions

- **A simple heuristic for chess**

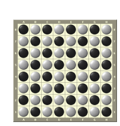
  pawn = 1
  knight = 3
  bishop = 3.5
  rook = 5
  queen = 9

  e(b) = [add up all the values of each piece for computer]
  - [add up all the values of each piece for user]



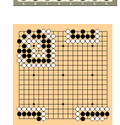
- **Konane**

  e(b) = [#of moves available for computer] – [#of moves available for user]



- **Go**

  Incorporates the difference between # of liberties, # of pieces, # of eyes



The Minimax Algorithm

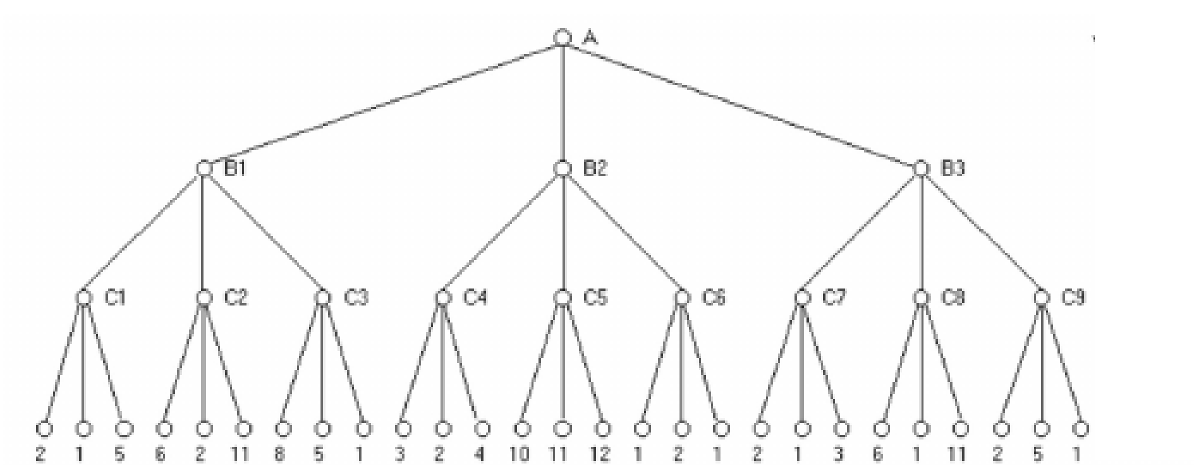
cbv = current backed up value
pbv = progressive backed up value

Function minimax(n) **returns** (pbv, move)
  if n at depth bound
    return (e(n), move(n))
  expand n to $n_1, n_2, \ldots, n_b$ successors

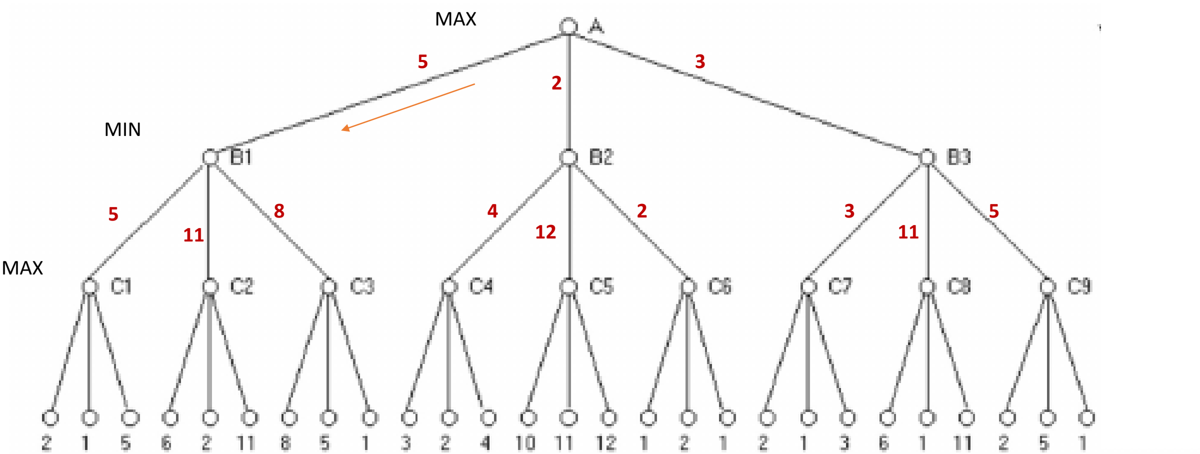
if n is a MAX node:
  cbv = -∞, bestMove = Ø
  for each n_i **in** $n_1, n_2, \ldots, n_b$
    bv, move = minimax(n_i)
    if bv > cbv
      cbv = bv, bestMove=move
  return (cbv, bestMove)

if n is a MIN node:
  cbv = ∞, bestMove = Ø
  for each n_i **in** $n_1, n_2, \ldots, n_b$
    bv, move = minimax(n_i)
    if bv < cbv
      cbv = bv, bestMove=move
  return (cbv, bestMove)

Which Move? What Value?



Which Move? What Value?



21

21

Complexity of Game Playing

- Minimax searches the entire tree up to level-***d***
- With a branching factor, ***b*** the complexity is **O(b^d)**

Tic Tac Toe
Average branching factor is 4, max depth is 9, i.e. 4^9=262,144 states
The actual number is far less since many games end well before 9 moves.

Konane
Average branching factor is 10. A typical game lasts ~20 moves per player. Therefore, 10^{40} states!

Chess
Average branching factor is 31..35. A typical game lasts ~20 moves per player.
Therefore, 31^{40} states!!

Go
Average branching factor is 250. A typical game lasts ~100 moves per player.
Therefore, 200^{250} states!!!

22

22

How to manage the Combinatorial Explosion?

- Only search to a limited ply (typically no more than 3-6)

Tic Tac Toe
Average branching factor is 4. If limited ply is 3 (i.e. ***d***=6), i.e. 4^6=4096 states

Konane
Average branching factor is 10. If limited ply is 3 (i.e. ***d***=6), 10^6 states

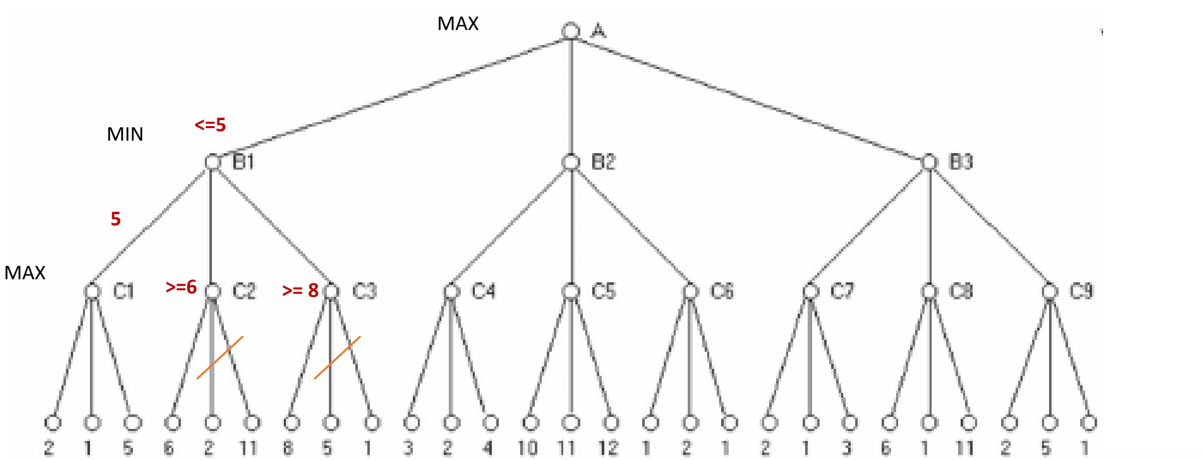
Chess
Average branching factor is 31..35. If limited ply is 3 (i.e. ***d***=6), 31^6 = 887 billion states. Still too large!

Go
Fuhgeddaboudit!!!

Improving Minimax with α-β Pruning

- Instead of searching the entire tree to level d, we can reduce the number of states searched by **pruning the tree** as the search progresses.

Minimax with α-β Pruning



25

25

Improving Minimax with α-β Pruning

- Instead of searching the entire tree to level ***d***, we can reduce the number of states searched by **pruning the tree** as the search progresses.
- Other improvements can also be made: **move ordering** is very common.
- The branching factor of the search can be effectively reduced to $\sqrt{\boldsymbol{b}}$ allowing the search to go deeper in the same amount of time for ***b***.
- In the end, all this and bigger faster computers have been very successful!

26

26

Minimax with α-β Pruning

Function minimax-α-β(n, α, β) **returns** (pbv, move)
  if n at depth bound
    return (e(n), move(n))
  expand n to $n_1, n_2, \ldots, n_b$ successors

if n is a MAX node:
  bestMove = Ø
  for each n_i **in** $n_1, n_2, \ldots, n_b$
    bv, move = minimax-α-β (n_i, α, β)
    if bv > α
      α = bv, bestMove=move
    if α >= β
      return (β, bestMove)
  return (α, bestMove)

if n is a MIN node:
  bestMove = Ø
  for each n_i **in** $n_1, n_2, \ldots, n_b$
    bv, move = minimax-α-β (n_i, α, β)
    if bv < β
      β = bv, bestMove=move
    if β <= α
      return (α, bestMove)
  return (β, bestMove)

27

27

Game Playing Successes

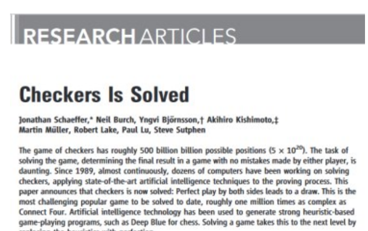
- **Checkers**

  Chinook (U. of Alberta)
  Beat the best player in the world in 1995 (though storied history!)
  In 2007, Chinook's team declared that "Checkers is solved!"

- **Chess**

  In 1996, IBM's Deep Blue beat Garry Kasparov in Philadelphia!

- **Go**
  We'll see later in the course!


RESEARCH ARTICLES

Checkers Is Solved

Jonathan Schaeffer,* Neil Burch, Yngvi Björnsson,† Akihiro Kishimoto,‡
Martin Müller, Robert Lake, Paul Lu, Steve Sutphen

The game of checkers has roughly 500 billion billion possible positions (5×10^{20}). The task of solving the game, determining the final result in a game with no mistakes made by either player, is daunting. Since 1989, almost continuously, dozens of computers have been working on solving checkers, applying state-of-the-art artificial intelligence techniques to the proving process. This paper announces that checkers is now solved: Perfect play by both sides leads to a draw. This is the most challenging popular game to be solved to date, roughly one million times as complex as Connect Four. Artificial intelligence technology has been used to generate strong heuristic-based game-playing programs, such as Deep Blue for chess. Solving a game takes this to the next level by replacing the heuristics with perfection.

Science, September 2007

28

28

References

- M. Wooldridge: *A Brief History of Artificial Intelligence*. Flatiron Books, 2020.
- Nils Nilsson, *Artificial Intelligence: A New Synthesis*, Morgan Kauffman, 1998.

29

29