Storia delle macchine per giocare a Scacchi

Paolo Ciancarini
Università di Bologna

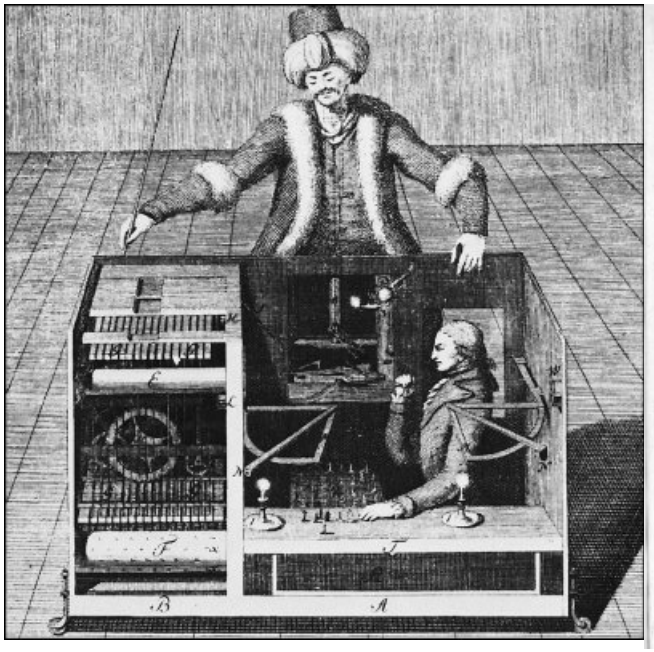
Sommario

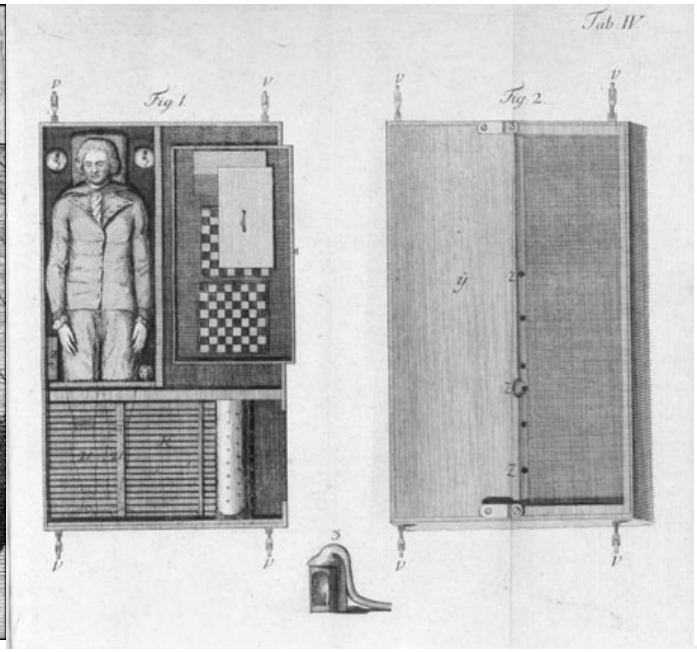
- Le macchine per giocare a Scacchi
- Elementi di Teoria dei Giochi
- La valutazione dell'albero di gioco
- I principali ricercatori
- Il panorama attuale

Il Turco

- Costruito nel 1769 dall'ungherese von Kempelen (1734-1804) per la regina Maria Teresa d'Austria
- Mostrato in tutte le corti d'Europa ed esibito al grande pubblico
- Distrutto verso il 1870, ricostruito di recente

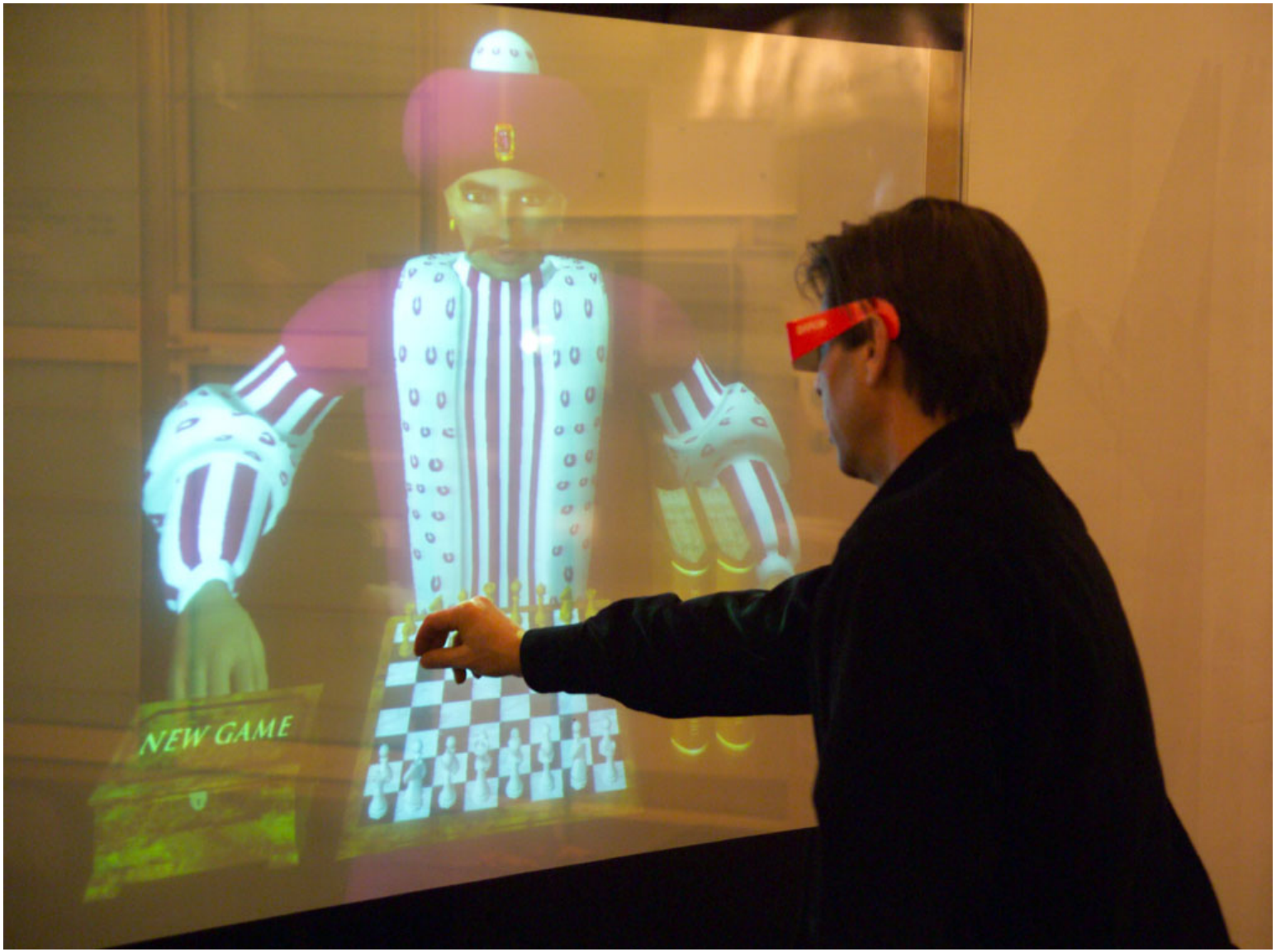
Il Turco



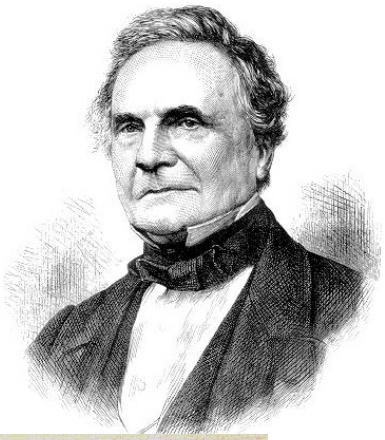


Il Turco giocava a Scacchi molto bene perché la sua intelligenza era… umana: la macchina conteneva un giocatore ben nascosto
Tuttavia esibiva alcuni accorgimenti meccanici d'avanguardia

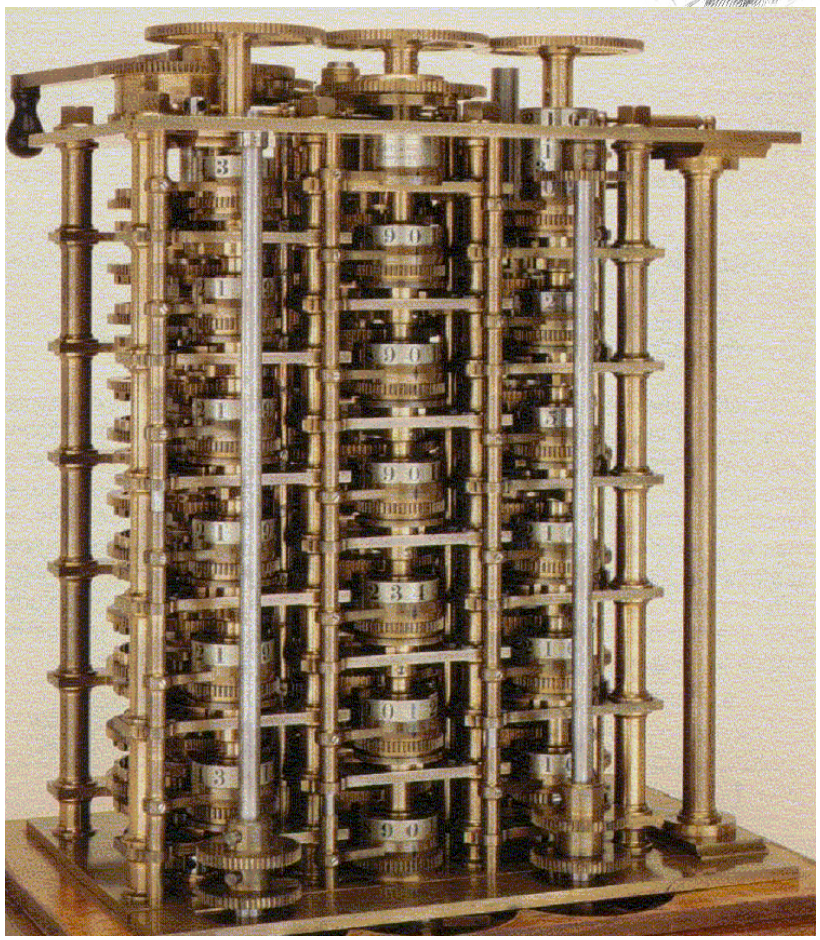
Il Turco virtuale



La macchina di Babbage

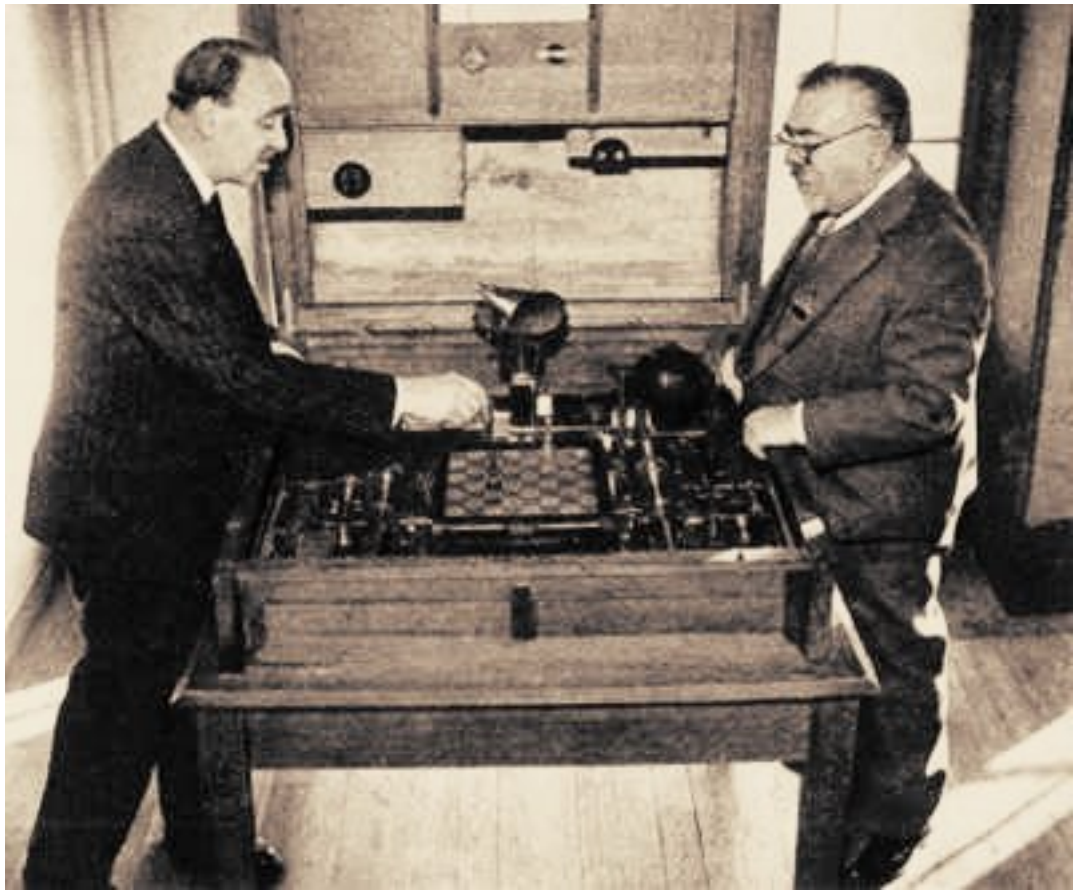


- Charles Babbage (1791-1871) progettò la prima macchina meccanica programmabile
- Descrisse come programmarla per giocare a Scacchi
- Non costruì mai la macchina (ricostruzioni vennero fatte nel '900)

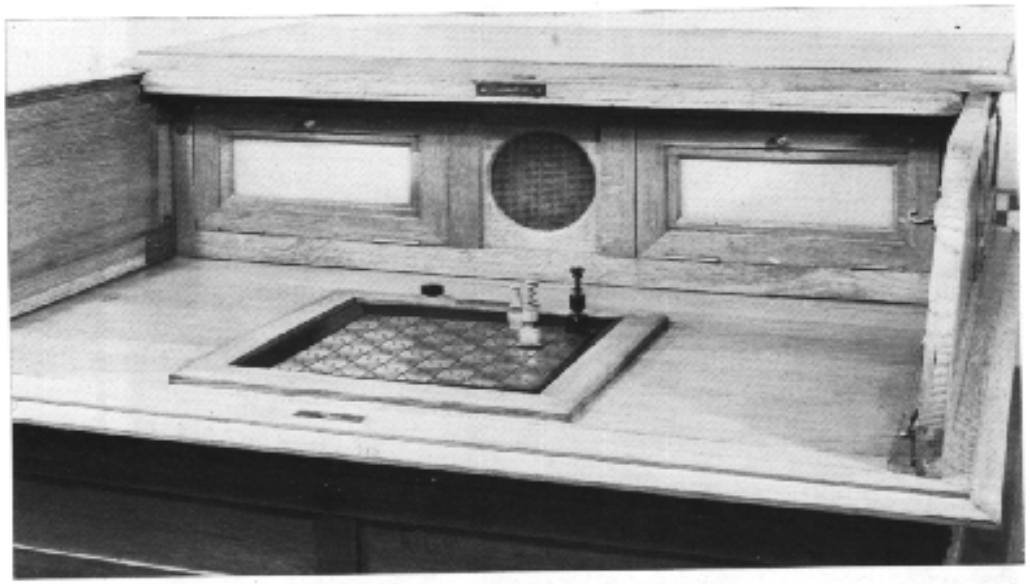


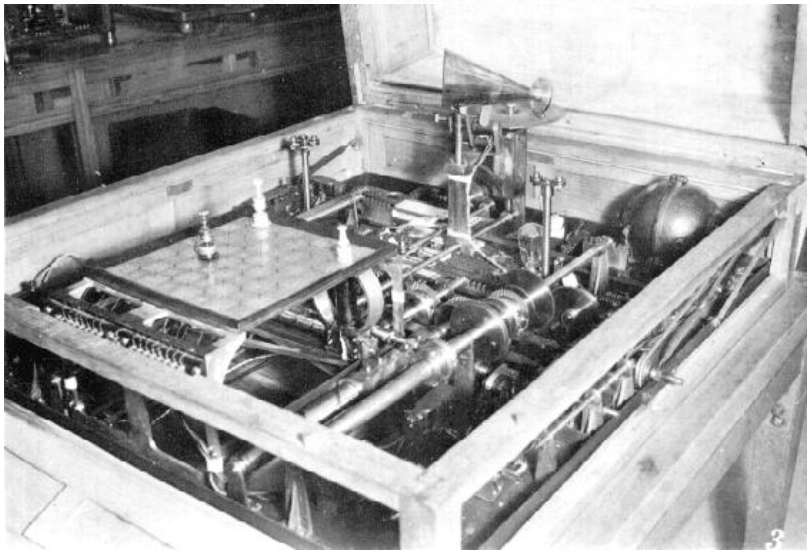
La macchina di Torre y Quevedo

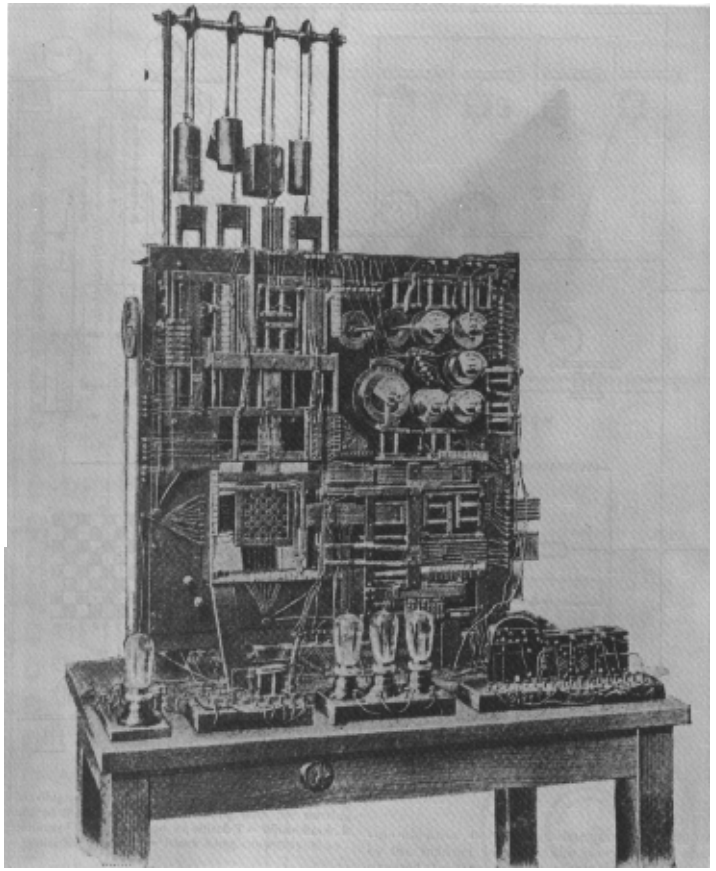
- Lo spagnolo Torres y Quevedo (1852-1936) costruì nel 1911 la prima macchina capace di giocare il finale di RT vs R
- La macchina si basava su sette regole realizzate meccanicamente
- Le sue macchine sono visibili in museo a Madrid
- Nella foto: il figlio di Torres y Quevedo mostra la macchina a Norbert Wiener, padre della cibernetica



La macchina di Torre Y Quevedo







Algoritmo cablato nella macchina di Torre y Quevedo

Torres y Quevedo's Mating Algorithm

Torres' scheme for effecting mate in the KRK endgame assumes an initial position with the automaton's White King on **a8**, Rook on **b8**, and the opponent's King on any unchecked square in the first six ranks. His algorithm for moving can be described in programming notation:

```
if        both BK and R are on left side {files a,b,c}
  then    move R to file h {keep R out of reach of K}
elseif    both BK and R are on right side {files f,g,h}
  then    move rook to file a {keep R away from K}
elseif    rank of R exceeds rank of BK by more than one
  then    move R down one rank {limit scope of BK}
elseif    rank of WK exceeds rank of BK by more than two
  then    move WK down one {WK approaches to support R}
elseif    horizontal distance between kings is odd
  then    {make tempo move with R}
          if      R is on a file then move R to b file
          elseif  R is on b file then move R to a file
          elseif  R is on g file then move R to h file
          else    {R is on h file} move R to g file
          endif
elseif    horizontal distance between kings is not zero
  then    move WK horizontally toward BK {keep opposition}
else      give check by moving rook down
          {and if on first rank, it's mate}
endif
```

If the opponent's King is placed on **a6**, with best delaying tactics mate can be staved off for 61 moves.

Teorema di Zermelo, 1913

Ernst Zermelo

- Nei giochi simili agli Scacchi, con un **numero finito di posizioni**, considerati anche senza regole di terminazione certa, **dove non esiste un elemento aleatorio**, e dove entrambi gli avversari hanno **informazione perfetta**:
  Ogni posizione,compresa quella iniziale, è:

- **Vinta per il Bianco**. Il Bianco può forzare la vittoria qualsiasi sia la strategia del Nero; **oppure**
- **Patta**. Sia il Bianco che il Nero possono forzare la patta, comunque giochi l'avversario. Se entrambi i giocatori fanno le mosse migliori la partita finisce pari, **oppure**
- **Vinta per il Nero**. Il Nero può forzare la vittoria qualsiasi sia la strategia del Bianco.

Von Neumann

Teorema del minimax

(Von Neumann and Morgenstern, 1944)

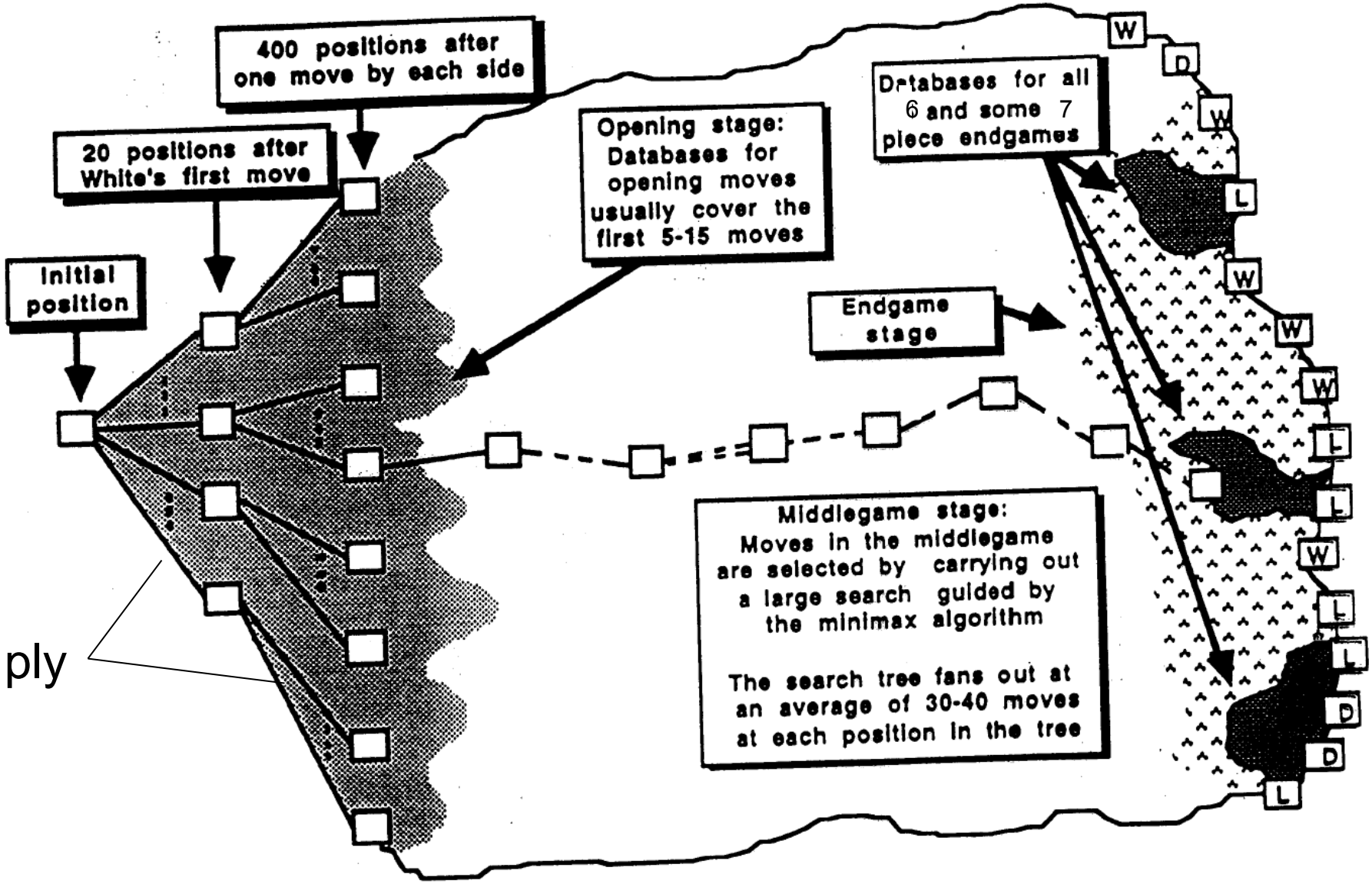
Morgenstern

- **Teorema del minimax**: costruzione della strategia vincente nei giochi simili agli Scacchi
- Impossibile da applicare agli Scacchi perché l'albero di gioco completo è troppo grande (**esplosione combinatoria**)
- NB: meno male, perché i giochi "risolti" sono poco interessanti dal punto di vista umano

Giochi finiti a due giocatori a somma zero

- Somma zero: io vinco tu perdi
- Albero di gioco
  - Radice: posizione iniziale, muove Max
  - Livelli pari: muove Max
  - Livelli dispari: muove Min
- I nodi foglia sono chiamati “posizioni terminali”; le regole di gioco definiscono il valore delle foglie

Albero di gioco (*game tree*) degli Scacchi



Prospettiva della teoria dei giochi

- Scacchi: gioco a informazione perfetta
- Gioco finito
  - Mosse enumerabili finite
  - Lunghezza finita della partita (vedi regole di patta)
- Gli scacchi hanno soluzione: vince/pari/perde
  - Equilibrio di J.Nash, razionalità perfetta
- Ma: la complessità computazionale limita le scelte razionali
  - Ricerche di H.Simon sulla razionalità limitata

John Nash

Herbert Simon

Tipi di giochi

- I giochi possono essere deterministici o no
  (se non lo sono si dicono “probabilistici” o “stocastici”)
- I giochi possono essere basati su informazione completa o no
  “informazione completa” : lo stato del gioco è completamente accessibile al giocatore

	Deterministici	Non-deterministici
Informazione perfetta	Scacchi, Dama, Go, Reversi	Backgammon, Gioco dell'Oca
Informazione imperfetta	Battaglia navale, Kriegspiel, Stratego	Bridge, Poker, Scarabeo, Risiko, Monopoli

Gli anni ’50

- Turing
- Shannon
- Simon
- Bernstein
- L’algoritmo alfabeta

Il programma di Turing

- Alan Turing (1912-1954) scrisse un programma per giocare a Scacchi che non venne mai implementato
- Conteneva istruzioni dettagliate: Turing lo eseguì agendo come CPU umana, ma gli occorrevano circa 30' per mossa
- Giocò una sola partita, persa, contro un umano

Turochamp-Glennie

- **Turing's paper machine – Alick Glennie**, Manchester 1952
- **1.e4 e5 2.Nc3 Nf6 3.d4 Bb4 4.Nf3 d6 5.Bd2 Nc6 6.d5 Nd4 7.h4 Bg4 8.a4 Nxf3+ 9.gxf3 Bh5 10.Bb5+ c6 11.dxc6 0-0 12.cxb7 Rb8 13.Ba6 Qa5 14.Qe2 Nd7 15.Rg1 Nc5 16.Rg5 Bg6 17.Bb5 Nxb7 18.0-0-0 Nc5 19.Bc6 Rfc8 20.Bd5 Bxc3 21.Bxc3 Qxa4 22.Kd2?** [22.h5 would have trapped the bishop] **22...Ne6 23.Rg4 Nd4?** [23...Rxb2! 24.Bxb2 Rxc2+] **24.Qd3 Nb5 25.Bb3 Qa6 26.Bc4 Bh5 27.Rg3 Qa4 28.Bxb5 Qxb5 29.Qxd6 Rd8 0-1.**

Maniac

- A Los Alamos un primo programma per Minichess venne realizzato su MANIAC

General Problem Solver

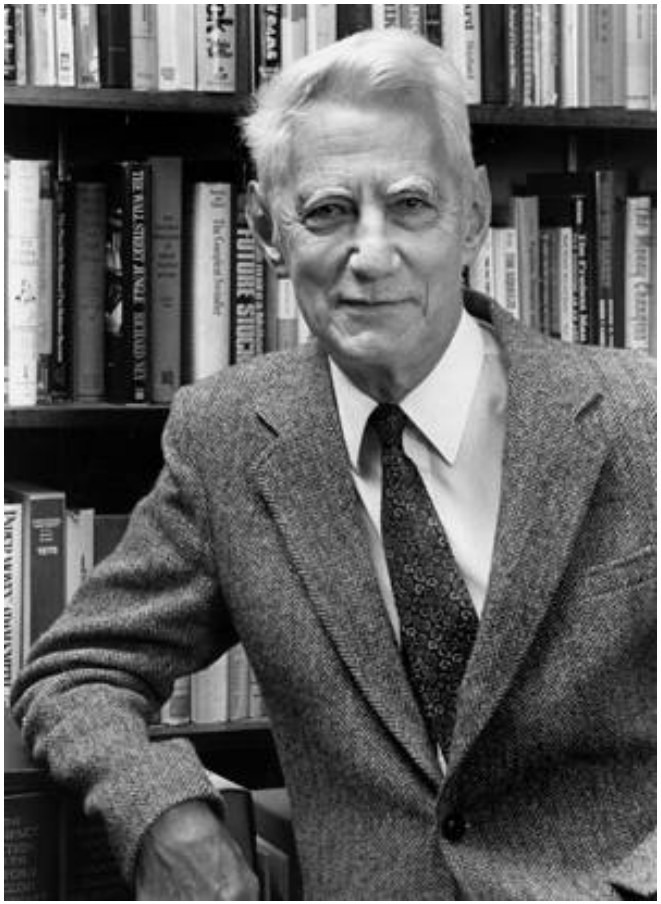
- Newell, Simon (CMU)
- Uno dei primi programmi AI
- “adattato” agli scacchi

Perché gli Scacchi?

- Se un computer potesse giocare a Scacchi, in un certo senso potrebbe “pensare”
- Problema ben definito formalmente sia nella parte operativa (le mosse) che nella parte obiettivo (lo scaccomatto), adatto alla programmazione su computer
- Soluzione non banale (spazio degli stati 10^{120}) ma non impossibile
- Abbondanza di avversari umani
- Letteratura sul gioco plurisecolare

La visione di Shannon

- Claude Shannon (1916-2001): padre della Teoria dell'Informazione
- Scrive il primo articolo dell'era moderna su come programmare una macchina scacchistica
- Pubblicato su Philosophical Magazine, 1950
- Disponibile in rete



L’architettura di Shannon

- Rappresentazione della scacchiera
- La funzione di valutazione
- La strategia di ricerca di tipo A (forza bruta: esplora tutte le varianti)
- La strategia di ricerca di tipo B (uso di euristiche di taglio della ricerca)

Contenuto informativo di una posizione scacchistica

- Rappresentazione di tutti i pezzi sulla scacchiera
- Chi ha la mossa?
- Re e torri hanno mosso? Serve per stabilire se è possibile l'arrocco
- Ultima mossa? Serve per stabilire se è possibile una presa *en passant*
- Numero di mosse fatte dall'ultima cattura o mossa di pedone? (serve per la condizione di patta)

Notazione FEN

Posizione pezzi (maiuscole Bianco, minuscole Nero)

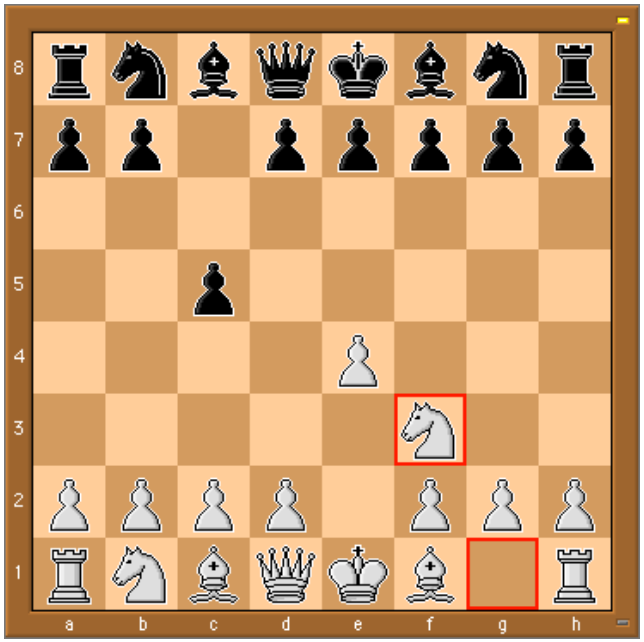
Giocatore che ha la mossa

Possibilità di arroccare

Possibilità di presa en passant

Numero semimosse da ultima presa o spinta di pedone

Numero progressivo prossima mossa



Il diagramma si rappresenta così:

[rnbqkbnr/pp1ppppp/8/2p5/4P3/5N2/PPPP1PPP/RNBQKB1R b KQkq - 1 2]

La patta per le 50 mosse

Regolamento F.I.D.E. degli Scacchi

- **9.3**

  The game is drawn, upon a correct claim by the player having the move, if

  a. he writes on his scoresheet, and declares to the arbiter his intention to make a move which shall result in the last 50 moves having been made by each player without the movement of any pawn and without the capture of any piece, or

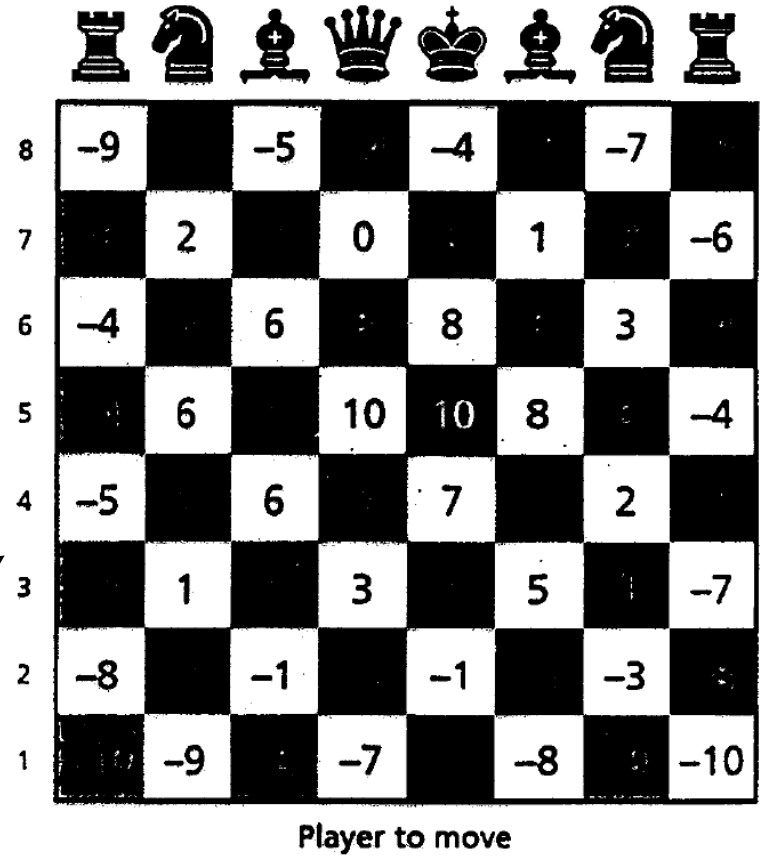
  b. the last 50 consecutive moves have been made by each player without the movement of any pawn and without the capture of any piece.

- Esistono posizioni in cui sarebbe possibile vincere se non valesse questa regola.
- Per esempio, certe posizioni KRBvsKNN sono vinte in non meno di 223 mosse senza catture

La funzione di valutazione

- Σ val(Bianco)- Σ val(Nero)

- Differenza di
  - Materiale
  - Mobilità
  - Sicurezza del Re
  - Coppia alfieri
  - Coppia torri
  - Colonne aperte
  - Controllo del centro
  - Altro



Valore della posizione del Cavallo
(dalla funzione di val di Deep Blue)

Una funzione di valutazione

- **Materiale**: I valori relativi di Regina (Q), Torre(R), Alfiere (B), Cavallo (N), Pedone sono nell'ordine 9, 5, 3, 3,1
- **Mobilità**: (M) premia l'aumento delle mosse disponibili (es. una torre dovrebbe occupare una colonna aperta)
- **Struttura pedonale**: i pedoni isolati (I), doppiati (D) o arretrati (S) sono deboli e valgono la metà
- **Sicurezza del Re**: il Re "coperto" dà un bonus

$$f(P) = 200(K-K') + 9(Q-Q') + 5(R-R') + 3(B-B'+N-N') + (P-P') - 0.5(D-D'+S-S'+I-I') + 0.1(M-M') + \ldots$$

La funzione di valutazione di Deep Blue

- Deep Blue usava ~6,000 diversi parametri calcolati in hardware
- Il peso dei parametri poteva variare mossa per mossa, in questo modo migliorava la capacità di valutazione posizionale
- Come vennero inizializzati i pesi:
  - Apprendimento da un db di circa 1000 partite di grandi maestri (~120 parametri)
  - Altri meccanismi di apprendimento (come per es. in backgammon)
  - Manualmente: un gran maestro giocò molte partite e dopo ogni errore la valutazione veniva visualizzata e i pesi modificati

Strategia di tipo A

- Considera (minimax) **tutte** le posizioni dell'albero di gioco fino ad una profondità prefissata d
- Per ciascuna posizione a profondità d, usa una funzione di valutazione per valutare (= prevedere il risultato finale) la posizione
- Shannon sostiene che si può implementare in 3 Kbit !
- Semplice ma inefficiente, a causa dell’esplosione combinatoria

Algoritmo Minimax

- Generare l'albero di gioco completo
- Alternativa: generarlo fino a una profondità prefissata
- Valutare gli stati terminali (foglie)
- Propagare i valori Min-Max dalle foglie alla radice
  - verso livello pari: si offre valore max
  - verso livello dispari: si offre valore min

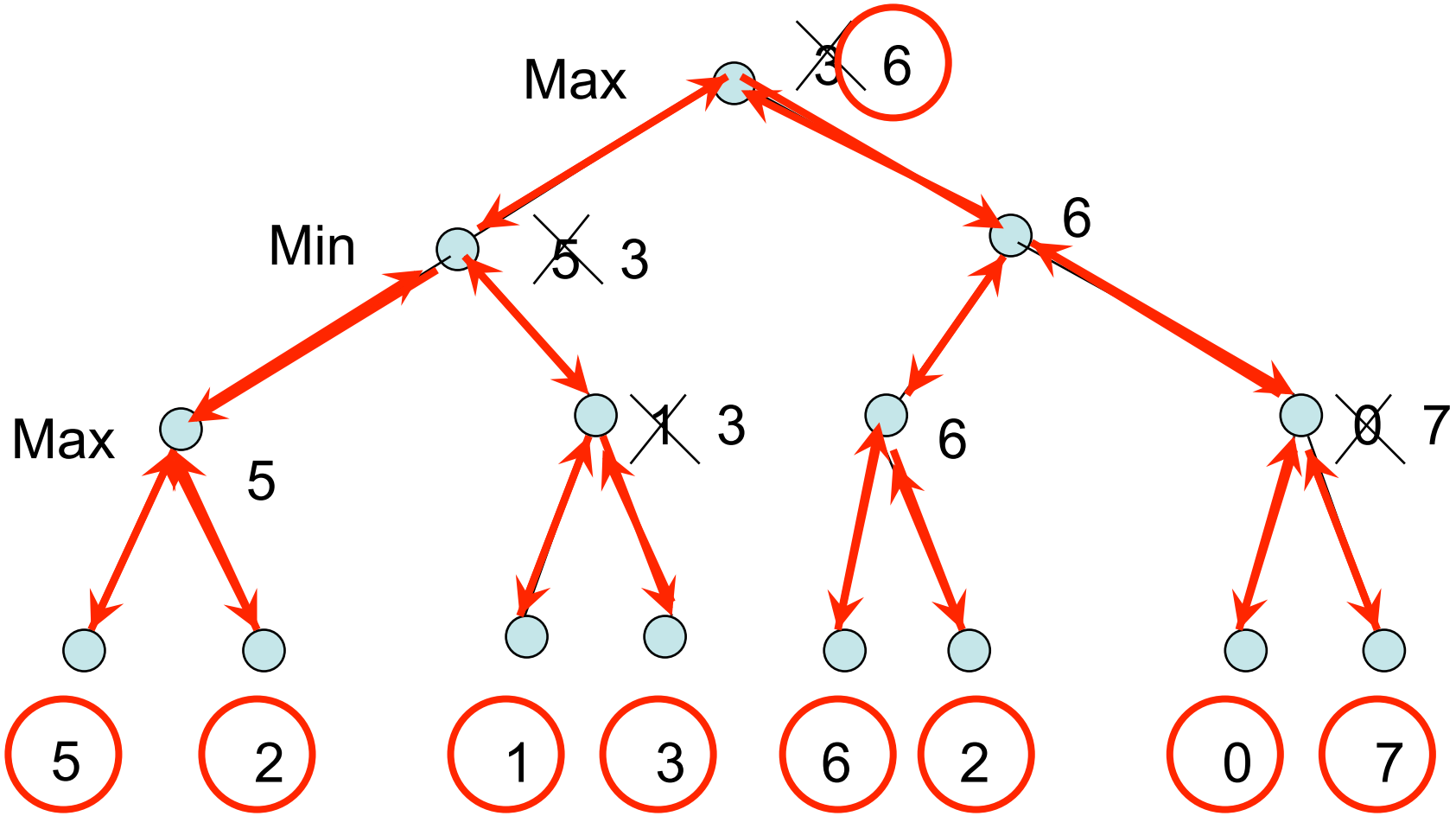
Pseudocodice di Minimax

```
fun minimax(n: node): int =
  if leaf(n) then return evaluate(n)
  if n is a max node
    v := L
    for each child of n
       v' := minimax (child)
       if v' > v, v:= v'
    return v
  if n is a min node
    v := W
    for each child of n
       v' := minimax (child)
       if v' < v, v:= v'
    return v
```

Pseudocodice di Minimax con terminazione a profondità d

```
fun minimax(n: node, depth: int): int =
  if leaf(n) or depth = 0 then return evaluate(n)
  if n is a max node
    v := L
    for each child of n
       v' := minimax(child,depth-1)
       if v' > v, v:= v'
    return v
  if n is a min node
    v := W
    for each child of n
       v' := minimax(child,depth-1)
       if v' < v, v:= v'
    return v
```

Minimax – Animazione



Varianti (migliorate) di Minimax

Negamax: come minimax, codice più compatto

```
int negaMax( int depth ) {
    if ( depth == 0 ) return evaluate();
    int max = -oo;
    for ( all moves)  {
       score = -negaMax( depth - 1 );
       if( score > max )
            max = score;
    }
    return max;
}
```

Complessità del Minimax

- La complessità è proporzionale alla dimensione dell'albero di gioco
- Se l'albero di gioco ha fattore di diramazione (*branching*) b e profondità d la complessità è $O(b^d)$
- Branching: numero medio di mosse in una posizione arbitraria
  Esempio:
- il branching degli Scacchi vale circa 33
- Profondità media di una partita a Scacchi: 40 mosse (80 semimosse)
- Stima della dimensione albero di gioco degli Scacchi: $33^{80} \sim 10^{120}$

Risultati controintuitivi (1)

- Beal (1980) e Nau (1982,83) si chiesero se i valori restituiti dal minimax fossero più credibili dei valori stressi. La ricerca mostrò che i valori portati alla radice dal minimax sono **meno** credibili
- L'anomalia sparisce se i valori dei nodi fratelli sono altamente correlati [Beal 1982, Bratko & Gams 1982, Nau 1982]
- Pearl (1983) contestò questa conclusione, e mostrò che anche se la dipendenza tra nodi fratelli può eliminare l'anomalia, in pratica giochi come gli Scacchi non possiedono dipendenze sufficientemente forti
  - Poche posizioni sono talmente vinte da non poter essere "rovinate" se uno lo fa apposta
  - Il successo del minimax "is based on the fact that common games do not possess a uniform structure but are riddled with early terminal positions, colloquially named blunders, pitfalls or traps. Close ancestors of such traps carry more reliable evaluations than the rest of the nodes, and when more of these ancestors are exposed by the search, the decisions become more valid."
- Pearl, On the Nature of Pathology in Game Searching. *Artificial Intelligenc*e 20(4), 1984
- Fenomeno non sufficientemente spiegato.
  Vedi: Lustrek, Gams, Bratko, Why Minimax works (IJCAI 2005)

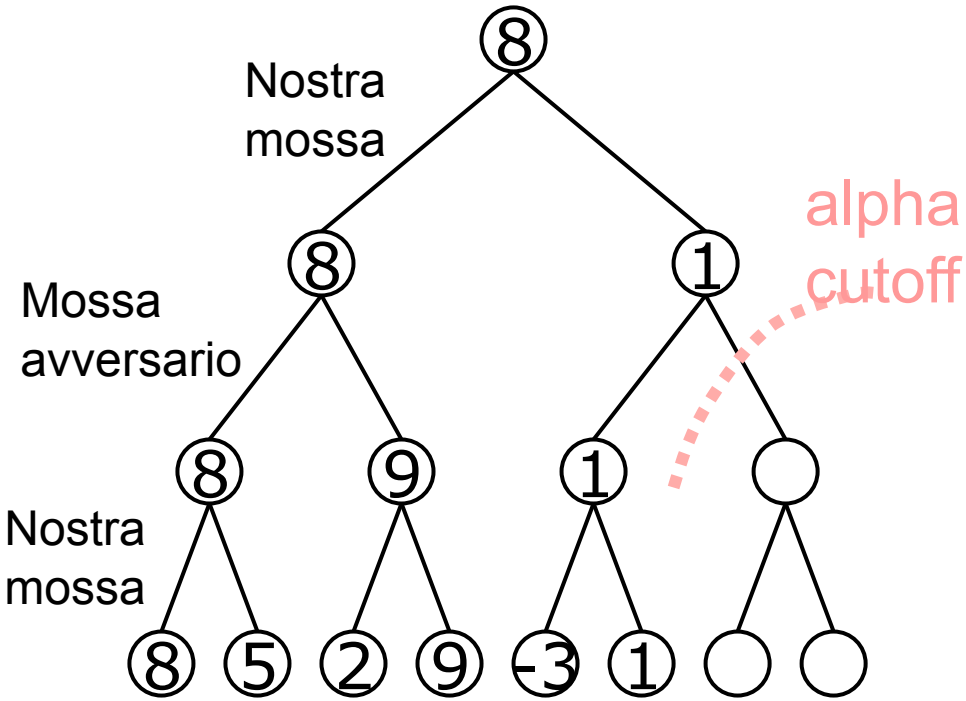
Risultati controintuitivi (2)

- Beal e Smith dimostrarono che ricerche minimax più profonde con valori foglia casuali portano a gioco migliore
- D.Beal e M.Smith, Random Evaluations in Chess. *ICCA Journal* 1994

Tagliare la ricerca

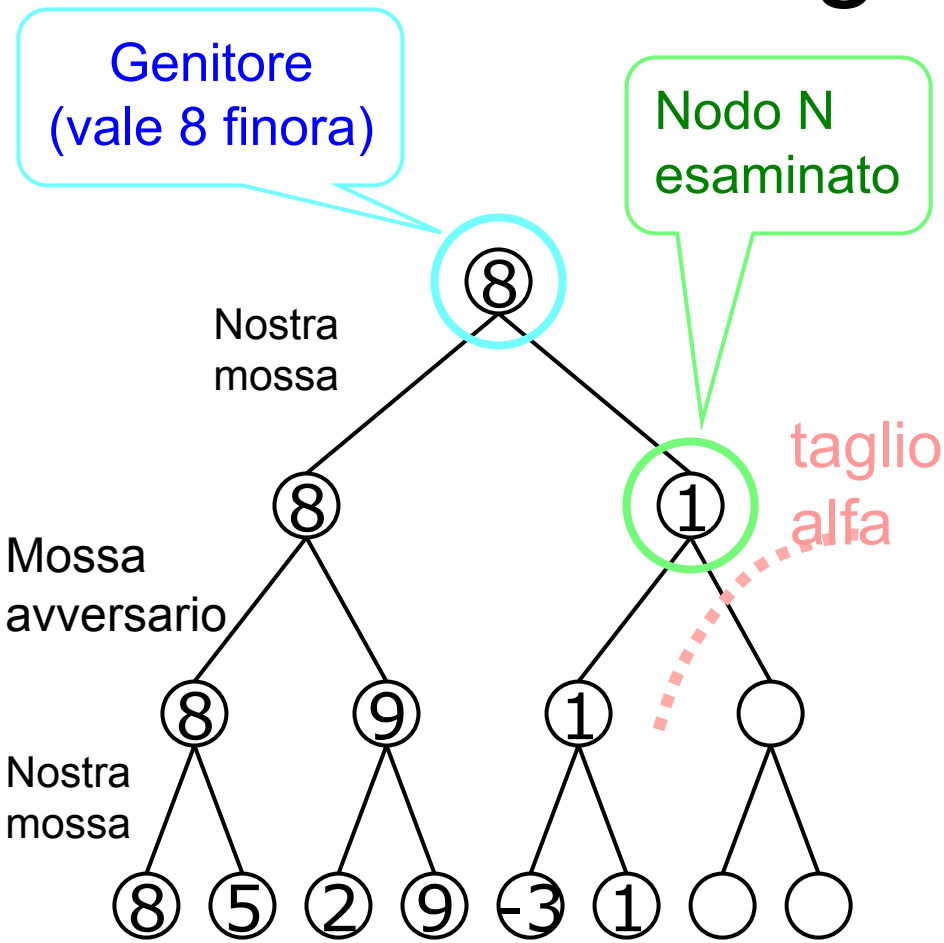
- Il Minimax effettua una visita completa dell'albero di gioco
- Siccome lo scopo è valutare la posizione iniziale, ci sono certi rami di ricerca che possono essere **tagliati**, senza influenzare la valutazione finale

Esempio algoritmo Alfabeta



- Il valore di root è 8 (finora)
- La mossa a destra vale 1, (finora)
- L'avversario non potrà mai aumentare questo valore: sarà sempre minore
- Possiamo ignorare gli altri nodi

Tagli alfa



- Si ha un alpha cutoff quando:
  - E' in esame un nodo N con mossa all'avversario, *e*
  - Il genitore di N ha un valore provvisorio, *e*
  - Il valore che sta salendo da N è minore di quello del genitore, *e*
  - Il nodo N ha altri rami figli non ancora considerati (che possiamo "potare")

Tagli beta

- Capita un taglio beta quando
  - E’ il nostro turno al nodo N
  - Il genitore di N ha un valore provvisorio minore di quello che sta salendo da N
  - N ha altri rami figli non ancora considerati e che quindi possiamo “potare”

Come funziona Alfabeta

- Si basa su due valori:
  - *alpha* valore della miglior scelta nel cammino MAX
  - *beta* valore della miglior scelta nel cammino MIN
- Regola: non visitare il nodo *n* quando

  beta <= *alpha*
  - Per un nodo MAX restituire *beta*
  - Per un nodo MIN restituire *alpha*

Minimax con tagli alfabeta

```
(* the minimax value of n, searched to depth d. If the value is less
 * than min, returns min. If greater than max, returns max. )
fun minimax(n: node, d: int, min: int, max: int): int =
 if leaf(n) or depth=0 return evaluate(n)
 if n is a max node
    v := min
    for each child of n
      v' := minimax (child,d-1,v,max)
      if v' > v, v:= v'
      if v > max return max
  return v
 if n is a min node
    v := max
    for each child of n
       v' := minimax (child,d-1,min,v)
       if v' < v, v:= v'
       if v < min return min
    return v
```

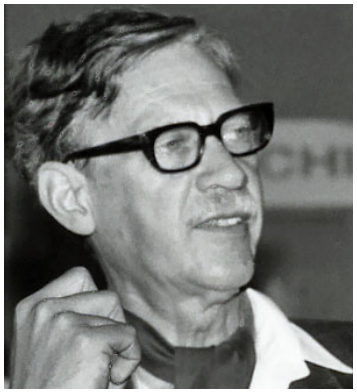
Complessità di Alfabeta

- Nel caso peggiore non ci sono tagli e la complessità è pari a quella di Minimax, cioè $O(b^d)$
- In media, ed in pratica, la complessità scende a $O(b^{d/2})$
- In sostanza se Minimax visita K nodi, Alfabeta ne visita $\sqrt{K}$

Strategia tipo A - svantaggi

- La strategia di tipo A soffre di
  - Esplosione combinatoria: ~10^9 valutazioni dopo 3 mosse (6 plies)
  - Ha bisogno di test di quiescenza, altrimenti valuta posizioni instabili
  - Non sviluppa strategie
  - Gli umani "funzionano" diversamente

Come giocano gli umani



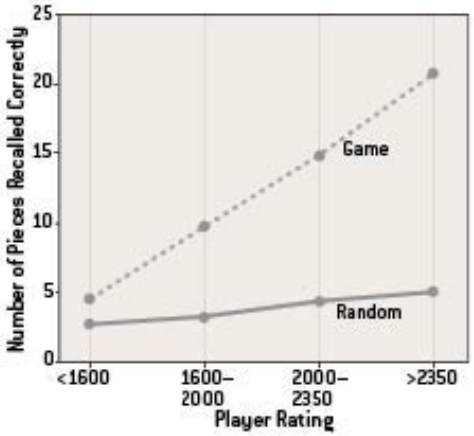
- Studio sperimentale dello psicologo De Groot, 1946
- Mostrò a diversi maestri varie posizioni
- I maestri esaminarono sino a 16 varianti, ciascuna profonda da uno a nove ply
- In tutto il giocatore più "riflessivo" considerò 44 posizioni dell'albero di gioco

A GRANDMASTER'S MEMORY

Experiments indicate that the memory of chess masters is tuned to typical game positions. In 13 studies conducted between 1973 and 1996 (the results were compiled in a review article published in 1996), players at various skill levels were shown positions from actual games (*a*) and positions obtained by randomly shuffling the pieces (*b*). After observing the positions for 10 seconds or less, the players were asked to reconstruct them from memory. The results (*graph at bottom*) showed that chess masters (with ratings of 2200 or higher) and grandmasters (generally 2500 or higher) were significantly better than weaker players at recalling the game positions but only marginally better at remembering the random positions. This finely tuned long-term memory appears to be crucial to chess expertise.

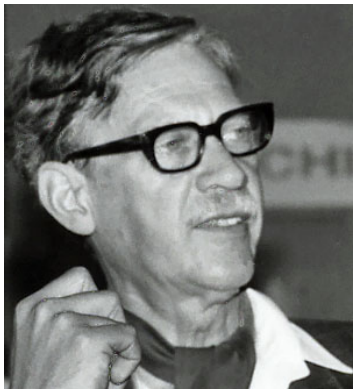
a

b



A structured knowledge of chess positions enables a grandmaster to spot the correct move quickly. The position at the right comes from a famous 1889 game between Emanuel Lasker (*white*) and Johann Bauer (*black*). Although a novice player would have to analyze the position extensively to see the winning move for white, any grandmaster would immediately recognize it. The correct move is shown on page 71.

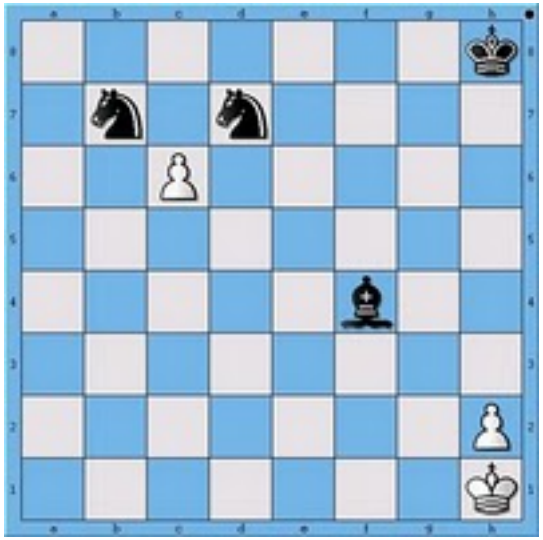
LUCY READING-IKKANDA; SOURCE: "CHUNKING MECHANISMS IN HUMAN LEARNING, BY FERNAND GOBET, PETER C.R. LANE, STEVE CROKER, PETER C.-H. CHENG, GARY JONES, IAIN OLIVER AND JULIAN M. PINE, IN *TRENDS IN COGNITIVE SCIENCES*, VOL. 5, NO. 6; JUNE 2001



A. de Groot

Strategia tipo B

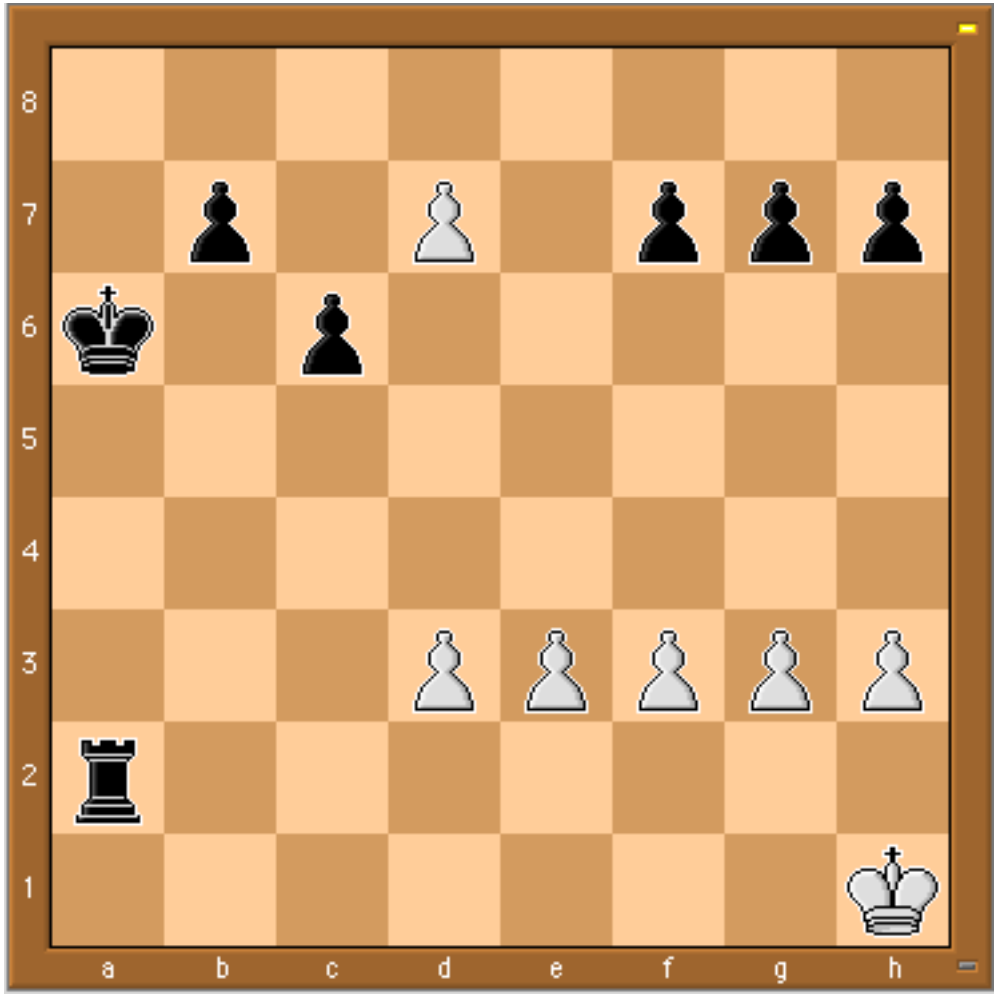
- Sceglie il sottoinsieme dell'albero da visitare in modo da non perdere tempo su varianti insensate
- Esamina le varianti "forzate" più a fondo; usa la f di valutazione solo in posizioni stabili (**quiescenti**)
- Nota: in apertura le posizioni quiescenti sono scarse: si usa un libro di aperture



Il Nero pensa di perdere un pezzo comunque, quindi A:h2

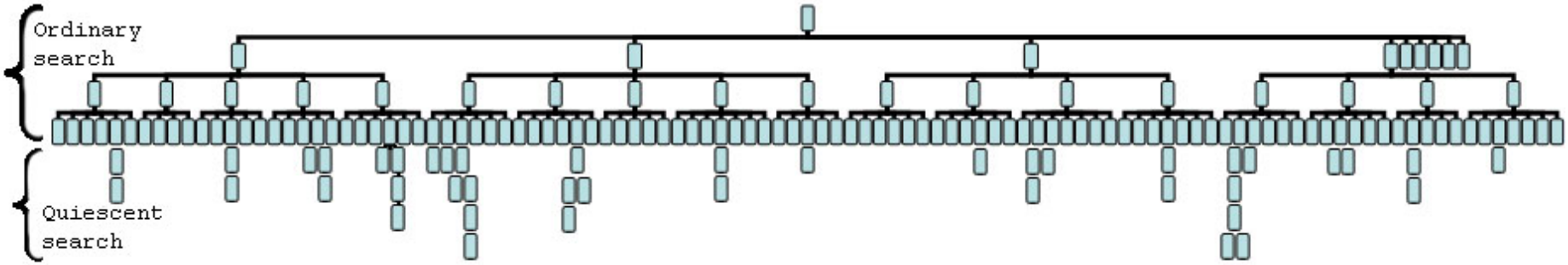
Effetto orizzonte

- Muove il Nero
- Il programma (col Nero) crede di essere in vantaggio
- Siccome sono disponibili parecchi scacchi, la promozione del pedone bianco è “oltre l’orizzonte”



Come combattere l'effetto orizzonte

- Ricerca quiescente
  - La funzione di valutazione (domain specific) calcola la stabilità di una posizione
  - Approfondisce la ricerca (oltre l'orizzonte normale) se la posizione è instabile
  - Necessita di tempo di ricerca variabile
- Estensioni singolari
  - Euristica indipendente dal dominio
  - Un nodo foglia viene approfondito se il suo valore è molto migliore di quello dei suoi fratelli
  - Anche 30-40 ply
  - Usato da Deep Blue



Posizioni quiescenti

La funzione g(P) di una posizione determina se una posizione è approsimativamente stabile (quiescente) :

- g(P) = 1 ,
  - un pezzo è attaccato da pezzo di minor valore, o se il numero dei suoi difensori è superato dai suoi attaccanti
  - esiste possibilità di Scacco al Re

- g(P) = 0 , altrimenti.

- La visita di un ramo dell'albero di gioco si ferma quando g(P)=0
- Si valutano tra 2 e 10 possibilità per ogni posizione

Limitare la ricerca

- Per ogni mossa M possibile nella posizione P occorre decidere se valutare il seguito di M:
- Sì, sempre:
  - Scacchi al re
  - Catture
  - Attacchi a pezzi maggiori
- Forse (dipende dalla posizione):
  - Mosse di sviluppo
  - Mosse difensive
- No: altre mosse
- Non bisogna trascurare mosse solo perché sembrano deboli a prima vista (es. pezzo lasciato in presa)

Altri sviluppi

- Libro di apertura
- Variazioni statistiche: creare una distribuzione di aperture e sceglierle a caso
- Approfondimento iterativo (iterative deepening)
- Gioco posizionale
- Gioco perfetto nei finali

Gli anni ’60

- Il programma di Kotok al MIT
- Il programma sovietico dell’ITEP
- MacHack (primo a usare tabelle di trasposizione)

Il programma di Kotok

- MIT, 1961
- Allievo di J.McCarthy
- assembler

I programmi sovietici

- Il programma ITEP
- PIONEER di Botvinnik
- Kaissa di AdelsonVelsky e Donskoy

AdelsonVelski

Botvinnik con Donskoy

Transposizioni

- Questa posizione si può ottenere con diverse sequenze di mosse
- 1.d4 d5 2.c4 e6 3.Cc3 Cf6
- 1.c4 Cf6 2.Cc3 e6 3.d4 d5
- eccetera

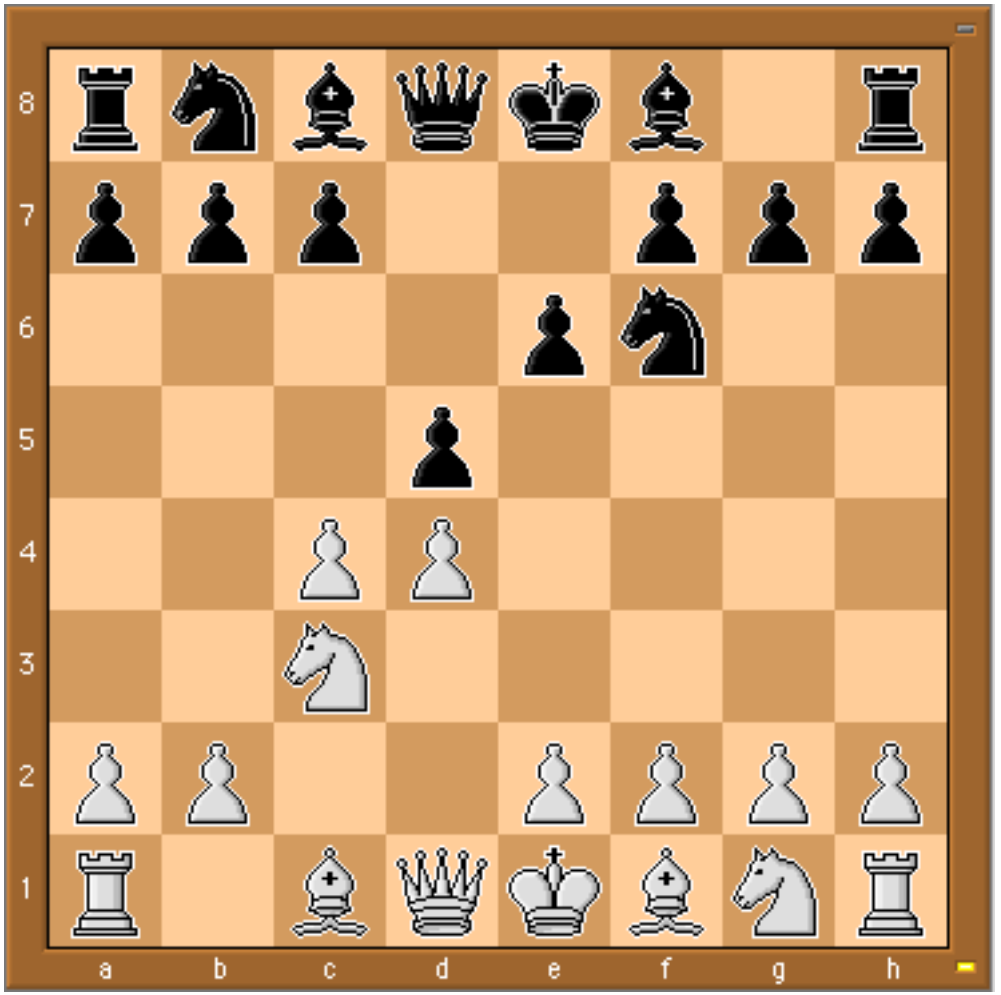
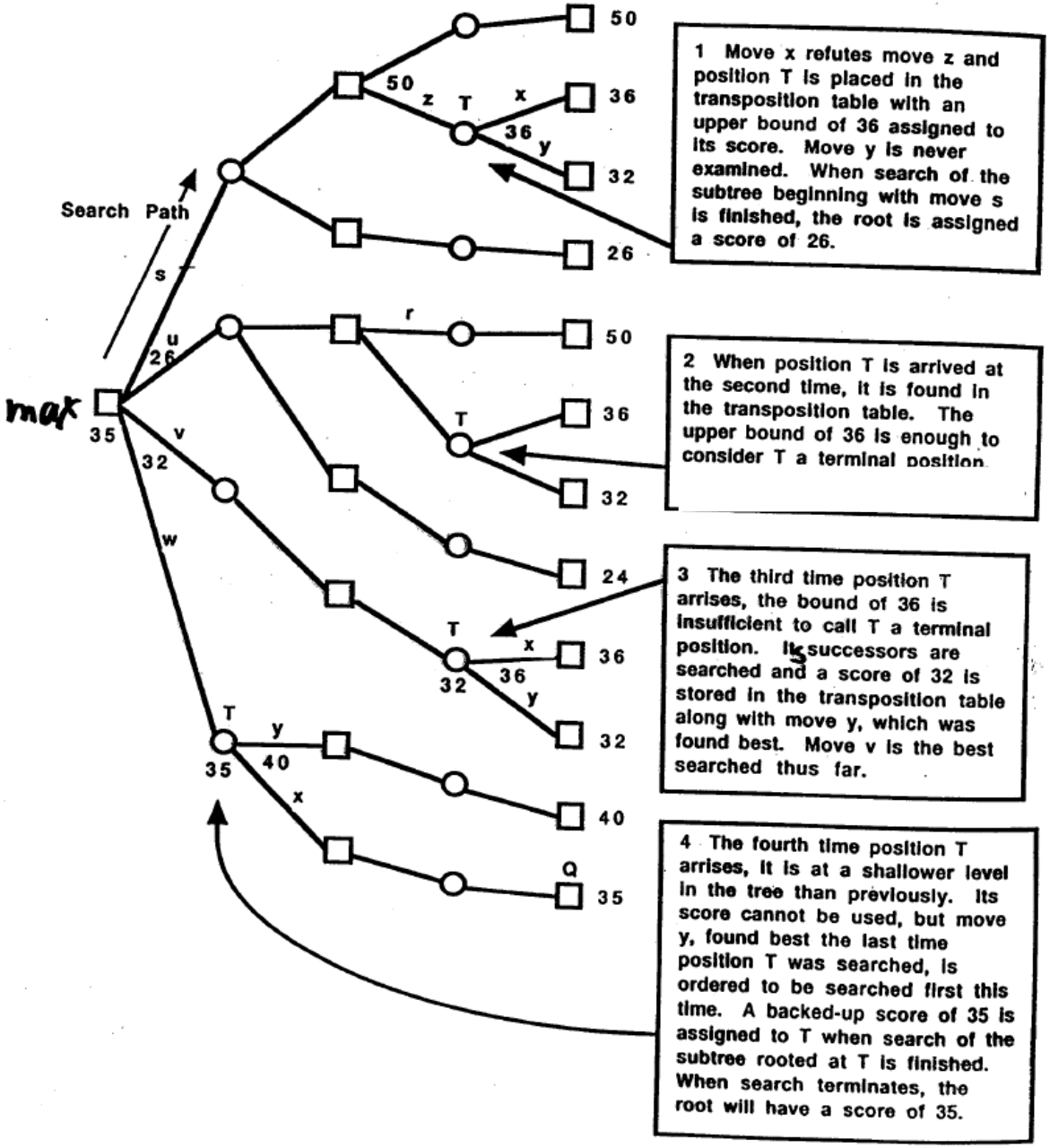


Tabella delle trasposizioni

- Una tabella hash memorizza milioni di posizioni “valutate”, per non doverle rivalutare
  - Posizione P
  - Codice hash di P
  - Valore di P
  - Mossa da fare (la migliore in P)
  - Profondità del sottoalbero di P valutato, con limiti inferiore e superiore
- Algoritmo
  - Per ogni posizione P da valutare si controlla la tabella
  - Se P è in tabella e
    - Nuova_profondità(P) ≤ profondità_memorizzata(P), e
    - I limiti del valore di P sono inferiori a qualche altra scelta
  - Allora P prende i valori in tabella
  - Altrimenti continua la valutazione (direttamente fval(P) o dopo ricerca a partire dalla vecchia mossa migliore: i nuovi valori per P vengono memorizzati in tabella
- Tabella piena: strategie di rimpiazzamento
  - Conserva posizioni con albero più profondo valutato sotto di esse
  - Conserva posizioni con più nodi cercati sotto di esse

Uso della tabella delle trasposizioni



Gli anni ’70

Il trofeo Shannon

- Kaissa, di Donskoy
- Chess, di Slate e Atkin
- Il primo campionato del Mondo
- Le prime scacchiere elettroniche basate su microprocessore

Slate e Atkin (1975)

- Programma Chess
- Nel 1973 abbandonano la strategia di Shannon di tipo B e passano al tipo A
- Usano tra l'altro iterative deepening

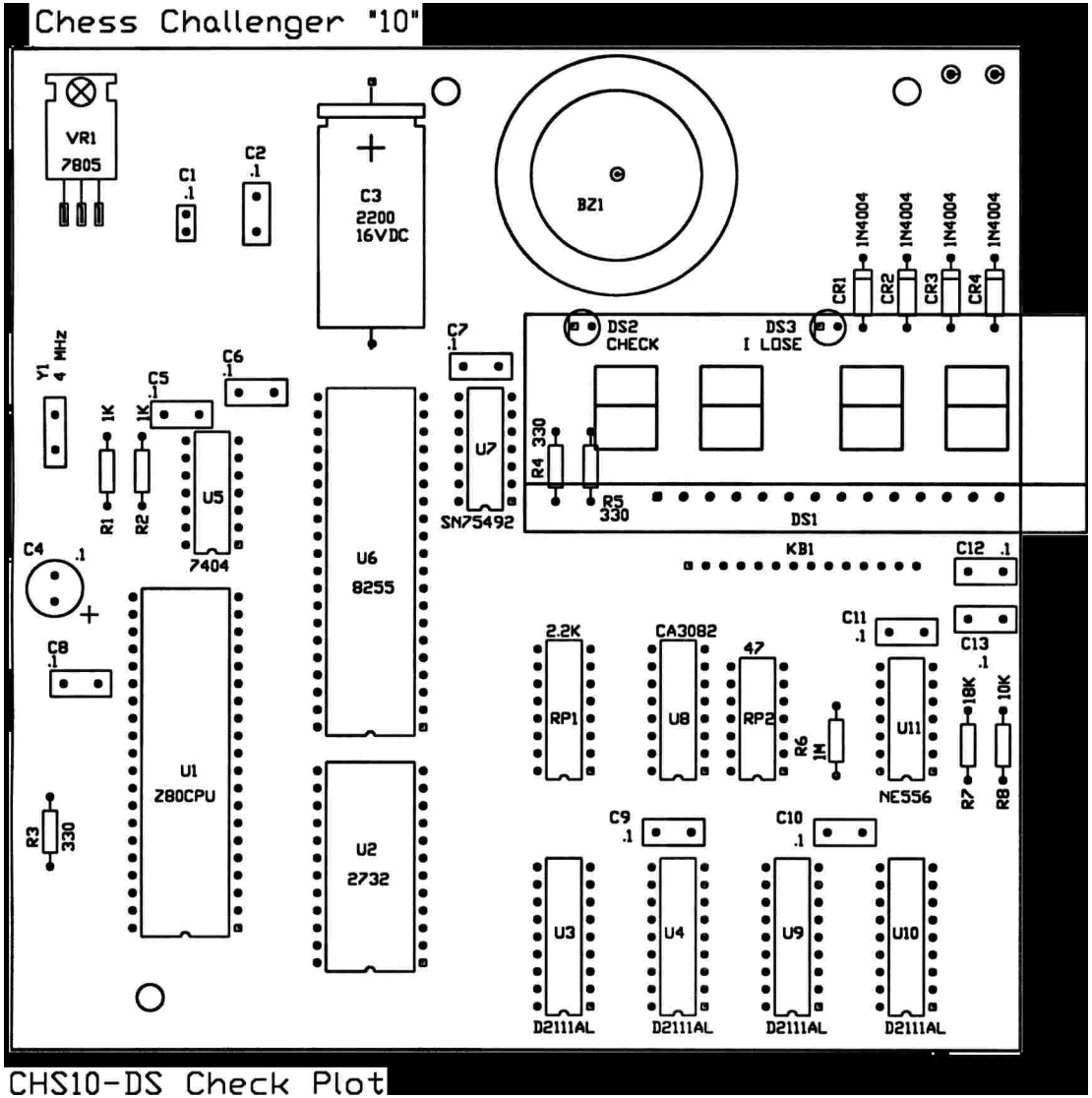
Iterative deepening

- Alfabeta è più efficace se i nodi vengono esaminati *best-move-first*
- Invece di espandere l'albero in "verticale" sino a profondità fissa (e applicando solo alle foglie la funzione di valutazione), meglio analizzarlo esaustivamente in "orizzontale" applicando la funzione sin dal primo livello per ordinare le mosse candidate
- Sembra costoso, ma il tempo investito a valutare i nodi interni è ripagato nell'ottenere tagli alfabeta spesso quasi ottimali
- Un altro vantaggio è che in ogni stato dell'algoritmo è sempre nota la mossa migliore trovata sin qui, il che è utile in situazioni agonistiche ove si gioca a tempo prefissato per ciascuna mossa

Chess Challenger

Caratteristiche CC10

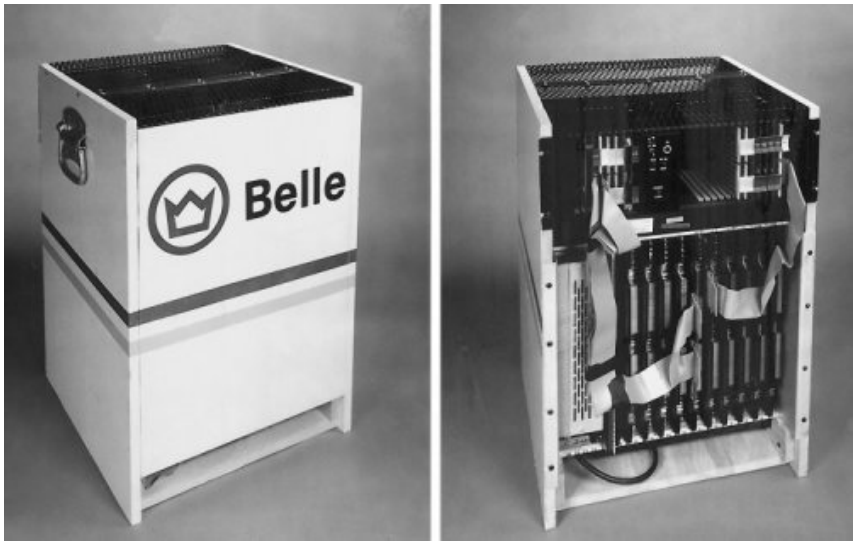
- Processore Z80A, 4MHz
- Memoria ROM 4KB
- Memoria RAM 0.5KB
- Programmatore R. Nelson



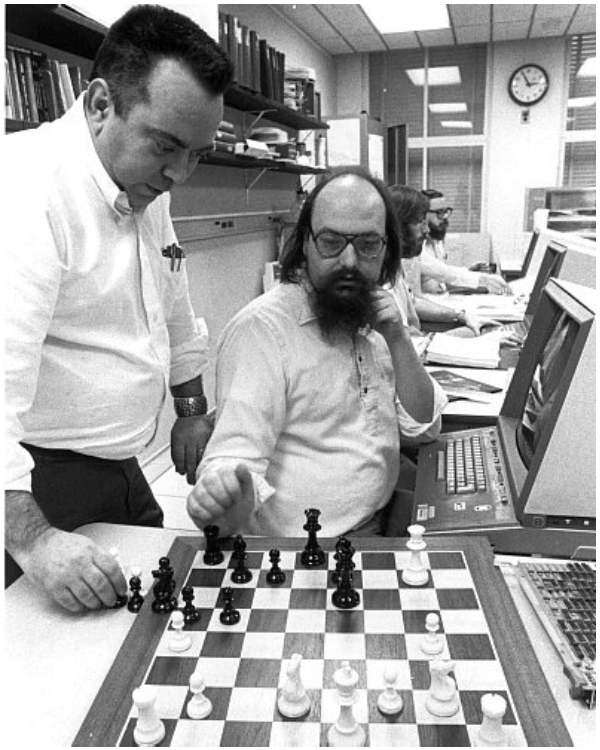
Gli anni ’80

- Belle di Condon e Thompson
- Cray Blitz di Hyatt
- Hitech di Berliner
- ChipTest e Deep Thought di Hsu e Campbell

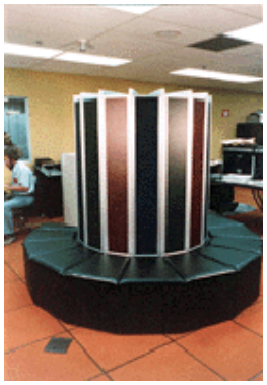
Belle



- Thompson e Condon di Bell Labs
- Fu la prima macchina basata su hardware speciale nella generazione di mosse
- Prima macchina a raggiungere il rango di Maestro (1980)



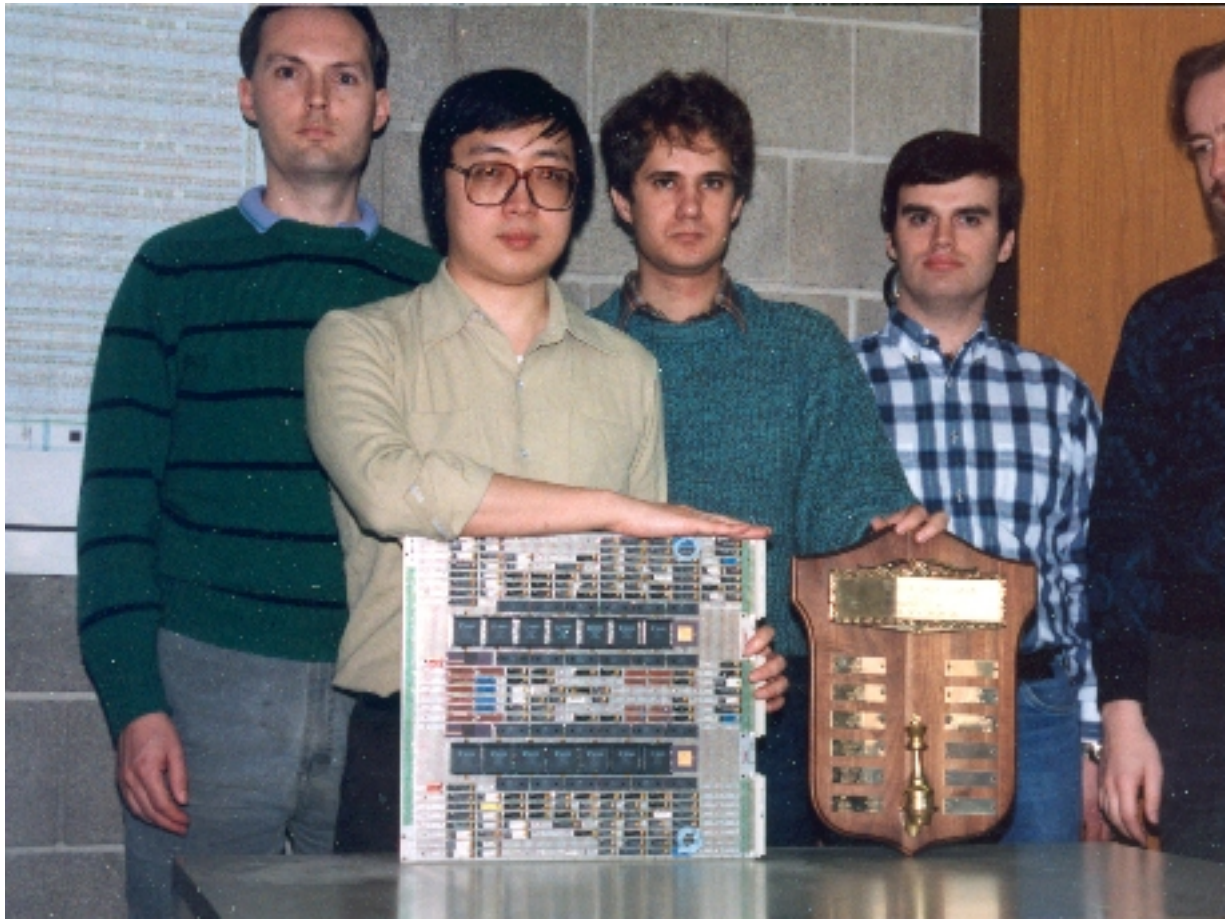
Cray Blitz



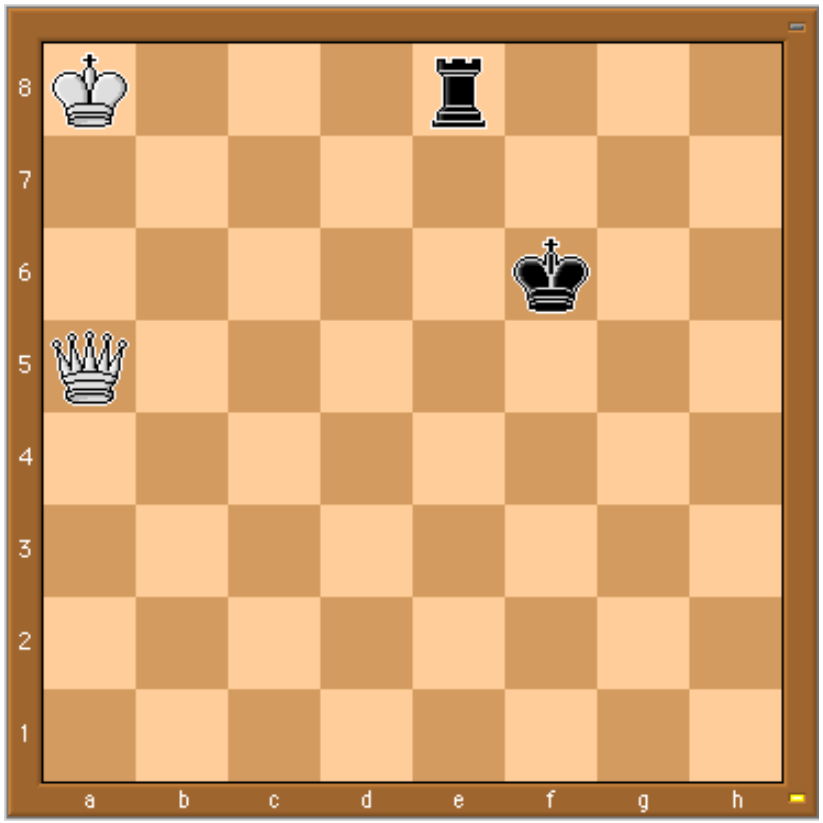
- Autore: R.Hyatt
- Programma per mainframe parallelo vettoriale
- Campione del Mondo 1983, 1986
- Oggi esiste Crafty, programma open source dello stesso autore

Chiptest e DeepThought

- Hsu e Campbell (CMU)
- Hw speciale e parallelo
- Prima macchina a battere un Grande Maestro
- Campione del mondo
- Nel 1990 acquistata da IBM, diventa Deep Blu



Uso del database dei finali

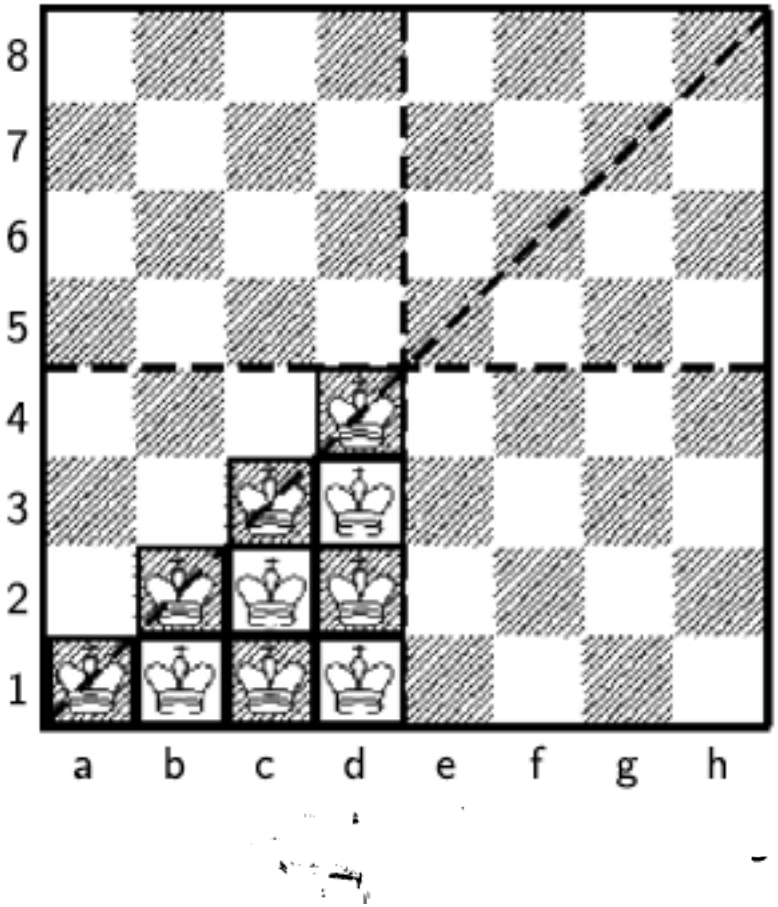


- Browne-Belle 1977
- Posizione nel DB di Belle: il Bianco vince in 30 mosse

Analisi retrograda dei finali

- Scelta di pezzi: es. KRvsK oppure KRRvsKNN
- Spazio degli stati = {WTM, BTM} x {tutte le possibili configurazioni dei pezzi scelta}
- Tabelle WTM e BTM, i cui stati sono connessi da mosse legali
- Inizia con stati terminali: matto, stallo, cattura immediata senza compenso (=riduzione ad altro problema). Marca ogni posizione vittoriosa di W (Bianco) come *vinta-in-0-mosse*
- Marca le posizioni non ancora classificate in WTM che con una mossa portano ad una *vinta-in-0-mosse* come *vinta-in-1-mossa* (e memorizza la mossa associata)
- Marca le posizioni non ancora classificate in BTM come *vinta-in-2-mosse* se esiste mossa forzata del Nero verso *vinta-in-1-mossa*
- Ripeti fino a che non è più possibile marcare alcuna posizione
- Fa' lo stesso per B (nero)
- Le posizioni che rimangono sono patte

Metodi di rappresentazione compatti



Position	Information on position
<a1-a1-a1>	0
<a1-a1-b1>	0
...	...
...	...
<a1-a1-h8>	0
<a1-b1-a1>	0
<a1-b1-b1>	0
...	...
...	...
<a1-c1-a1>	0
<a1-c1-b1>	0
...	...
...	...
<a1-c1-h8>	0
...	...
...	...
<d4-h8-h8>	0

(a)

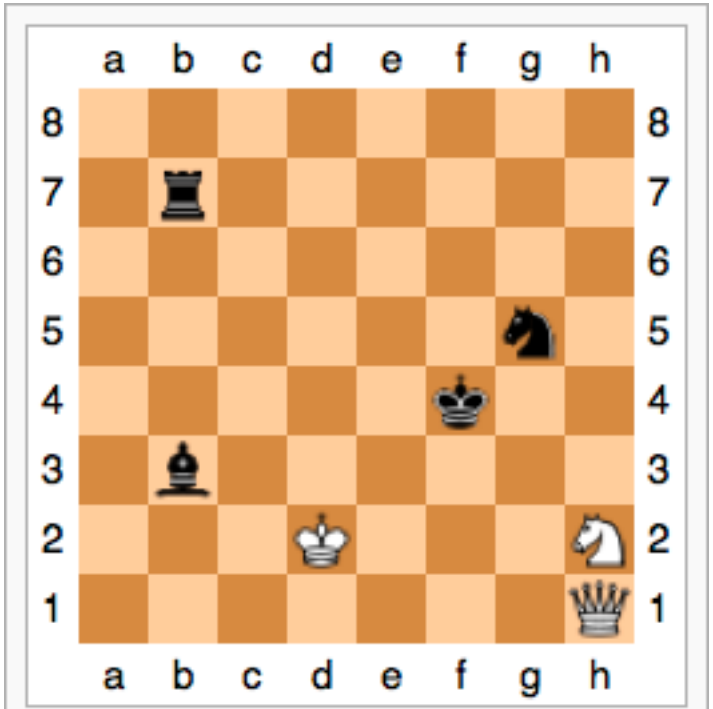
Position	Information on position
<a1-a1-a1>	Illegitimate
<a1-a1-b1>	Illegitimate
...	...
...	...
<a1-a1-h8>	Illegitimate
<a1-b1-a1>	Illegitimate
<a1-b1-b1>	Illegitimate
...	...
...	...
<a1-c1-a1>	Illegitimate
<a1-c1-b1>	In check
...	...
...	...
<a1-c1-h8>	In check
...	...
...	...
<d4-h8-h8>	In check

(b)

Building a KQK database: (a) initial contents of database, and (b) contents after performing the first step.

Risultati sui database

- Risolti nel 2006 tutti i finali con 6 pezzi e molti con 7 pezzi (completamento atteso per il 2015)
  - Record attuale: matto in 517
- Modifica delle regole
  - La FIDE ha recepito la difficoltà di alcuni finali e modificato la regola delle 50 mosse
- Conoscenza scacchistica
  - Separa la torre dal re in KRKQ
  - KRKN creduto patto, invece
    - Bianco vince 51% se ha la mossa
    - Bianco vince 87% se non ha la mossa



With Black to move, White forces a win in 517 moves.

Gli anni ’90

- I microprocessori dedicati
- Deep Blue
- I programmi commerciali

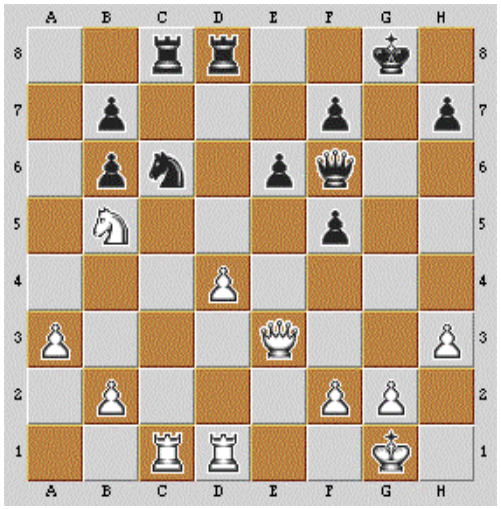

Mephisto®
ChessMachine
128 K
● Einsteckkarte für alle MS-DOS-Computer (XT/AT)
● Schachprogramm von Ed Schröder
● Der Weltmeister in Vancouver 1991
● Elo-Zahl ca. 2.200, unabhängig vom verwendeten PC-Prozessortypen
● ARM 2 RISC-Prozessor 32 bit
HG

Deep Blue

- Deep Blue (IBM) sconfisse Kasparov nel 1997
- Risultato di 10 anni di ricerche dei suoi autori
- Varie tesi di dottorato

Hsu

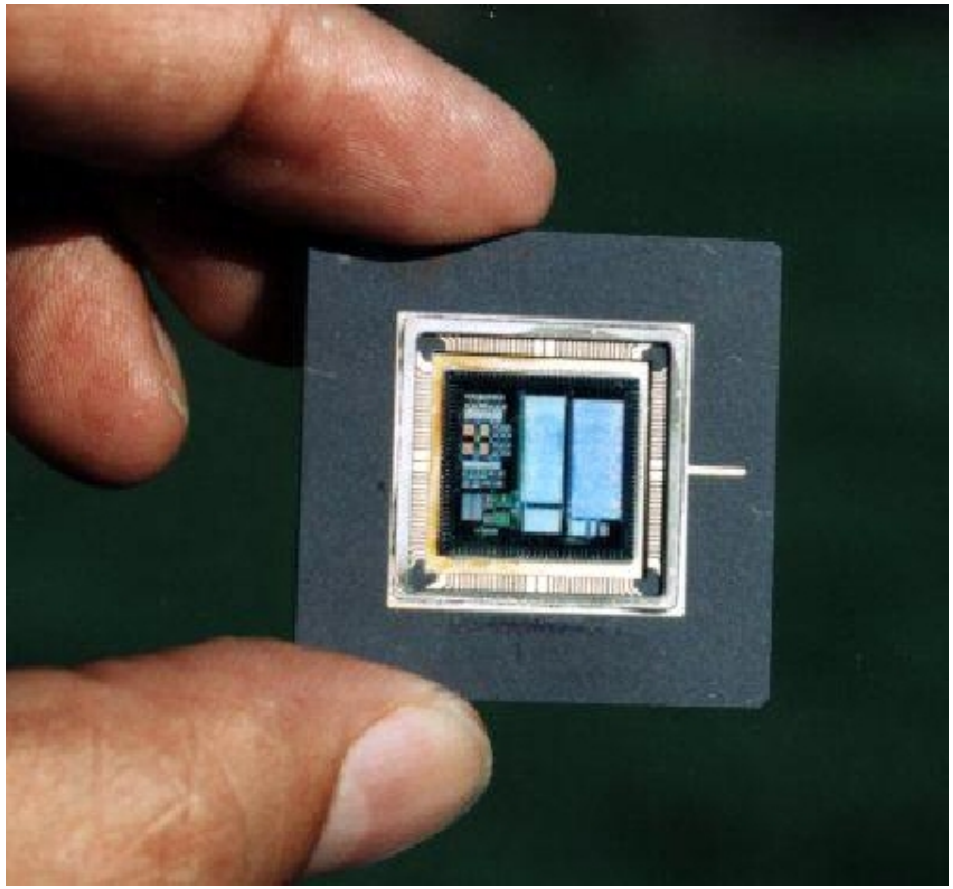
Campbell



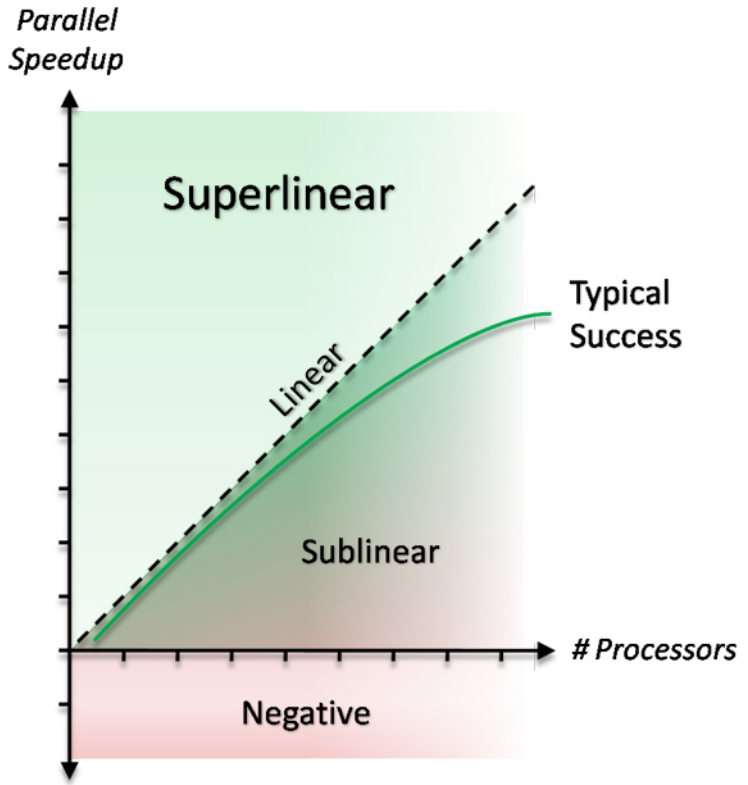
Deep Blue

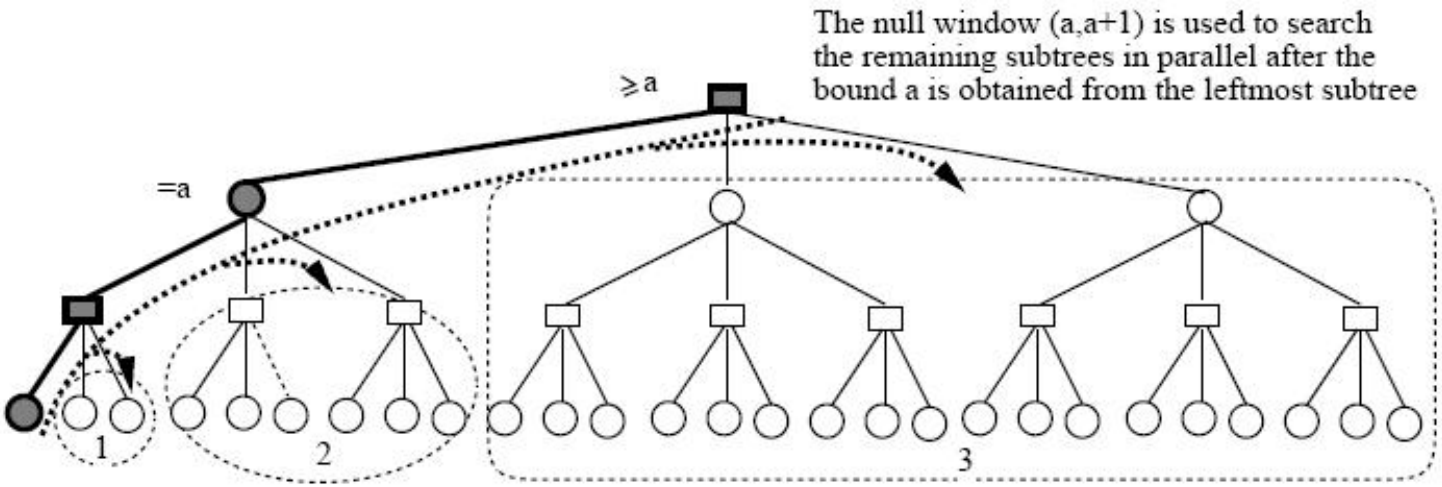
- Costruito dopo Deep Thought
- Algoritmo di analisi dell'albero:
  - iterative-deepening alpha-beta search, tabella trasposizioni, database di aperture (700,000 partite), e finali (tutti quelli con meno di 6 pezzi)
- hardware:
  - 30 processori IBM RS/6000
    - Strategie
  - 480 processori speciali
    - Generazione delle mosse
    - Valutazione della posizione (oltre 8000 termini)
- In media:
  - 126 milioni di nodi/sec., analisi 30 miliardi di posizioni per mossa, profondità media: 14 ply

Deep Blue



Ricerca parallela





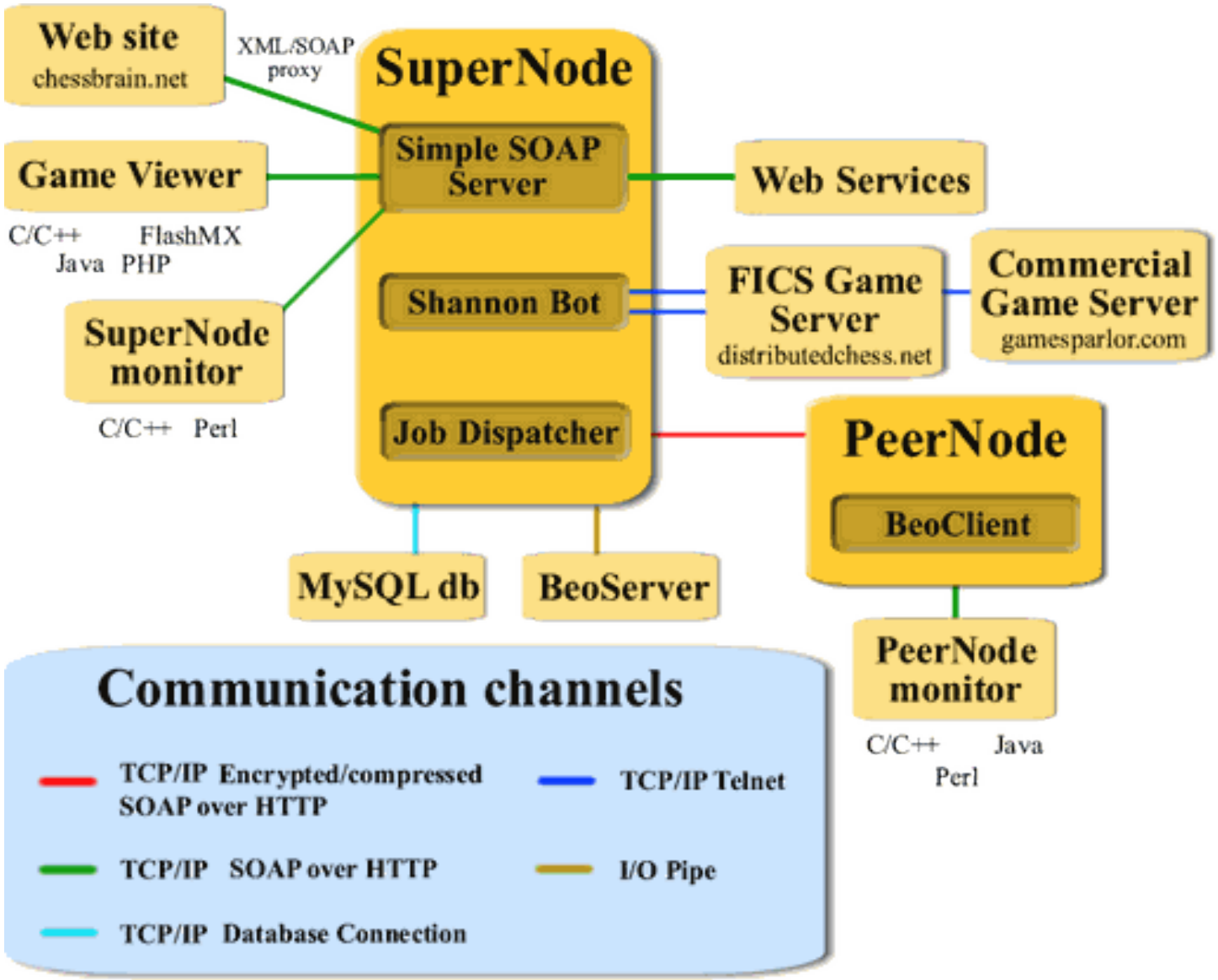
Programmi per PC

- Fritz
- Junior
- Shredder
- Chessmachine

Il nuovo secolo

- ChessBrain
- Hydra
- Risultati recenti: Rybka

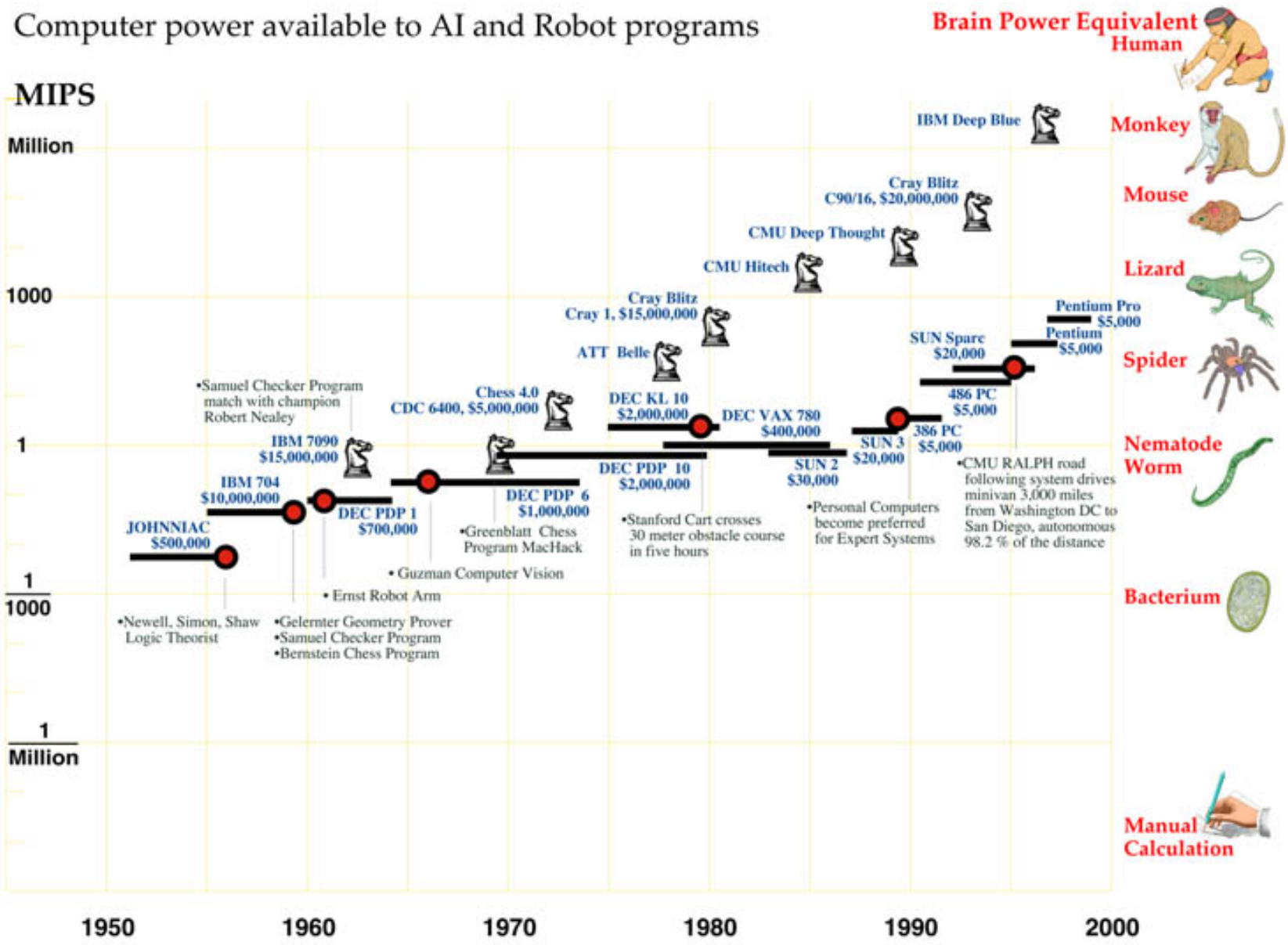
ChessBrain 2004



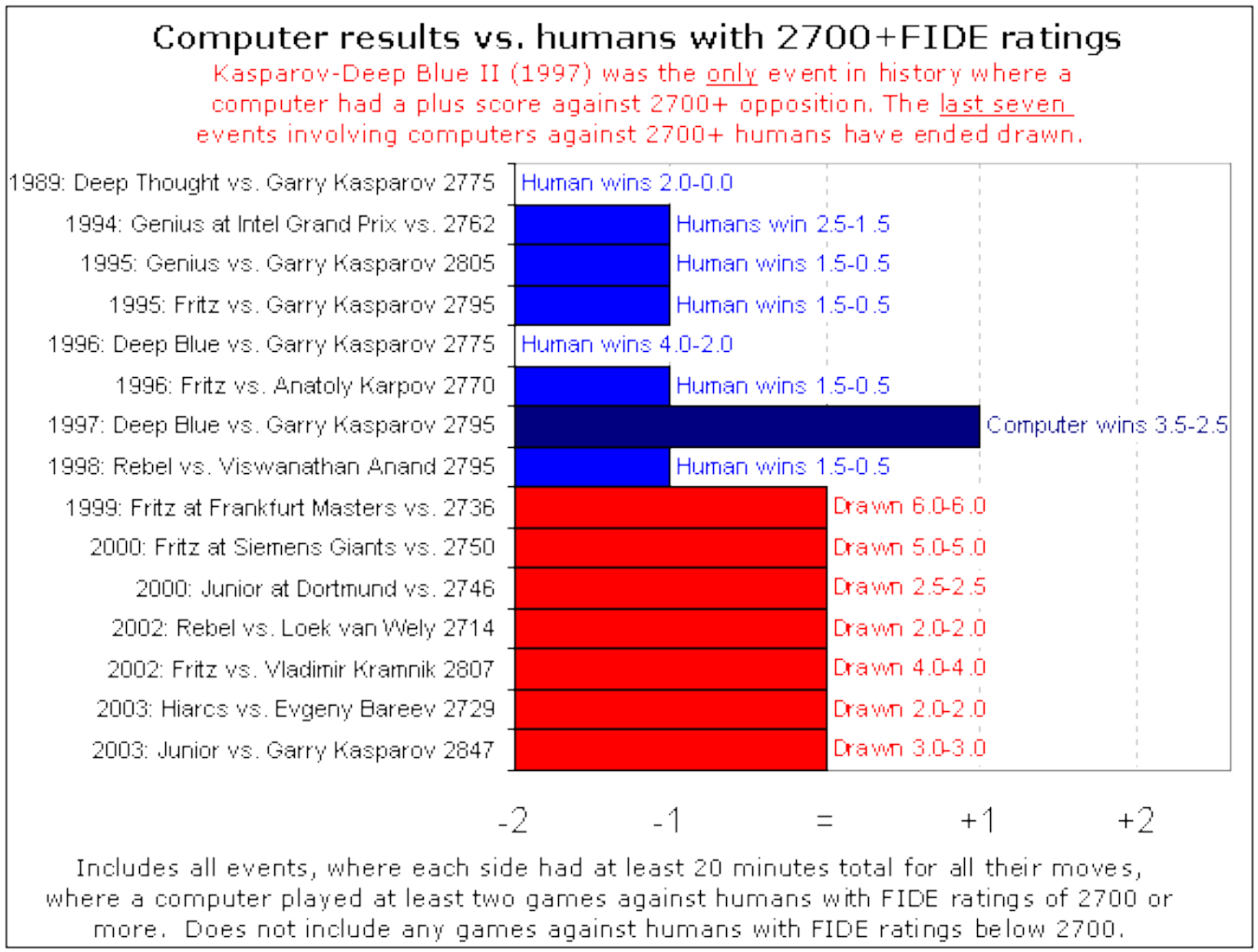
Hydra

- Donninger (EAU)
- Macchina parallela
- Tecnologia FPGA
- Ottimi risultati contro gli umani nel 2003-2005
- Progetto terminato

Legge di Moore e Scacchi



Risultati recenti



Risultati recenti dei match uomo-macchina

- 2005: Hydra-Adams 5½-½
- 2006: Fritz-Kramnik 4-2
- 2008: Rybka gioca vari match con handicap

Campioni del Mondo

• Kaissa	1974	Stoccolma
• Chess	1977	Toronto
• Belle	1980	Linz
• Cray Blitz	1983	New York
• Cray Blitz	1986	Colonia
• Deep Thought	1989	Edmonton
• Rebel	1992	Madrid
• Fritz	1995	Hong Kong
• Shredder	1999	Paderborn
• Junior	2002	Maastricht
• Shredder	2003	Graz
• Junior	2004	TelAviv
• Zappa	2005	Reykjavik
• Junior	2006	Torino
• Rybka	2007	Amsterdam
• Rybka	2008	Pechino
• Rybka	2009	Pamplona
• Rybka	2010	Giappone

Il Trofeo Shannon, che va all'autore del programma Campione del Mondo

Sommario

1950 Minimax con funzione di valutazione (Shannon e Turing)
1966 Alfabeta e forza bruta (Kotok e McCarthy)
1967 Tabelle di trasposizione (MacHack)
1957 Iterative deepening depth first search (Chess)
1977 Database sui finali con programmazione dinamica
1978 Circuiti speciali (tesi di master di Babaoglu, Belle)
1983 Algoritmi paralleli di ricerca su alberi di gioco (CrayBlitz)
1985 Valutazione parallela (Hitech)
1987 Ricerca parallela e circuiti speciali (Deep Blue)
1988 Conspiracy numbers e altri algoritmi avanzati
1990 Analisi retrograda su macchine altamente parallele
1997 DeepBlue batte il campione del mondo
2003 e seguenti: nuovi algoritmi basati su MonteCarlo search

Sviluppi futuri

E adesso?

- Altri giochi, per es.
  - Go
  - Poker
  - Kriegspiel
- Nuovi algoritmi

Bibliografia

- C.Shannon , Programming a Computer for Playing Chess, *Philosophical Magazine*, Ser. 7, 41(312) – March 1950
- S.Russel, P.Norvig, *Artificial Intelligence A Modern Approach*, 3a ed., Prentice Hall, 2009
- P.Ciancarini, *Giocatori Artificiali di Scacchi*, Mursia, 1992

Siti interessanti

- ChessBase, A short history of computer chess, `www.chessbase.com/columns/column.asp?pid=102`
- Mastering the Game, a History of Computer Chess `www.computerhistory.org/chess/`
- Sulla programmazione per gli Scacchi `chessprogramming.wikispaces.com`