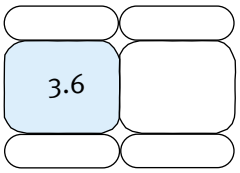
IL COMPUTER GIOCA A SCACCHI

Paolo Ciancarini

La storia delle macchine che giocano a scacchi è ancor più antica di quella del computer. Infatti la prima macchina scacchistica venne esibita nel 1770 a Vienna: era il Turco, un famoso automa che giocava con l'inganno, perché celava abilmente una persona tra i suoi ingranaggi. In questo articolo viene delineata la storia più recente delle macchine che giocano a scacchi e i principali aspetti di ricerca ad esse correlati.



1. INTRODUZIONE

Per valutare l'efficacia di un dispositivo hardware ne analizziamo le caratteristiche tecniche: possiamo per esempio misurare la velocità di calcolo di un processore, la dimensione di una memoria centrale o periferica, la risoluzione di un monitor, la banda di comunicazione disponibile su una connessione. Ma come si valuta l'efficacia di un software? Per esempio, come possiamo confrontare due sistemi operativi? Il confronto tra Windows e Linux di solito avviene su basi ideologiche, dunque fortemente soggettive. E comunque, come si dimostra che la versione 2.0 di un certo programma è migliore della 1.0 ? Non stiamo parlando di contare gli errori; supponiamo che entrambe le versioni siano prive di errori: come si confronta la loro "efficacia"?
Nel caso dei programmi che giocano a scacchi, la risposta è semplice: tra due programmi, è migliore quello che batte l'altro. In figura 1, tratta da [11], vediamo un grafico che mostra l'evoluzione di questi programmi. In ascissa ci sono tre scale correlate che descrivono rispettivamente la profondità media di analisi dell'albero di gioco (in *ply*, ossia semimosse), il numero medio di posizioni analizzate per secondo, i MIPS offerti dall'hardware. In ordinata abbiamo la scala Elo, che descrive la forza di un giocatore di Scacchi. La scala prende il nome da A. Elo, uno studioso di statistica, che tra il 1960 ed il 1970 mise a punto un sistema di valutazione della forza di gioco che è tutt'oggi usato in tutto il mondo. Per esempio, il campione del mondo "vale" di solito circa 2800 punti, un principiante vale 800 punti, mentre la curva di Gauss che descrive la "popolazione scacchistica" mondiale ha una mediana intorno ai 1400. Le variazioni di punteggio avvengono in seguito al conteggio delle vittorie e delle sconfitte in tornei ufficiali. La figura mostra le date di nascita e la forza relativa dei principali programmi di gioco, e culmina con Deep Blue, una macchina creata da IBM nei laboratori di Yorktown appositamente per battere il campione del mondo.
In effetti, Domenica 11 maggio 1997 è una data che gli storici dell'informatica segnano

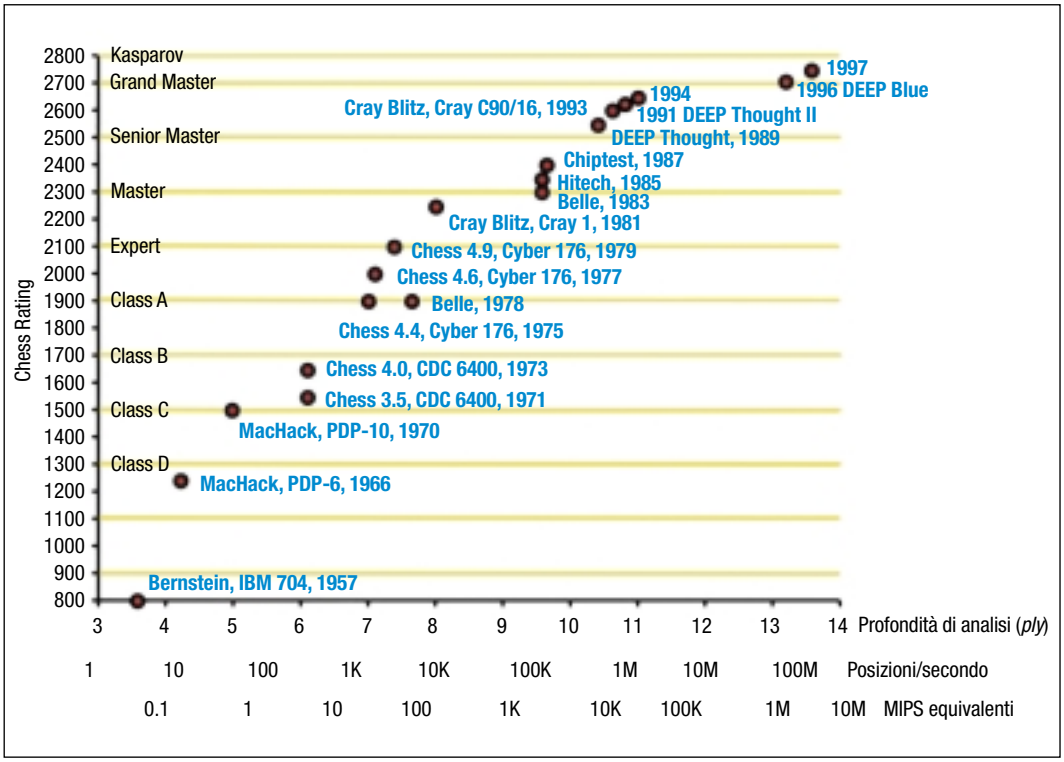


FIGURA 1
Il progresso dei programmi di gioco

con un sassolino bianco. Quel giorno Deep Blue ha sconfitto il campione del mondo di scacchi, il russo Garry Kasparov, dopo un match di sei partite (due vinte, una persa e tre patte). L'evento è stato seguito e commentato in diretta da decine di migliaia di persone connesse a Internet [10]. Tale impresa va considerata come uno dei grandi successi scientifici di questo secolo, cui hanno contribuito in varia misura sin dalla fine degli anni '40 parecchi studiosi, come Alan Turing, Claude Shannon, Herbert Simon, John McCarthy e Ken Thompson, solo per citarne alcuni.
Per quali ragioni si costruiscono macchine capaci di giocare? La giustificazione scientifica di tali sforzi è stata espressa quasi 50 anni fa in un articolo di Newell, Shaw e Simon:

"Gli Scacchi sono il gioco intellettuale per eccellenza. Senza far uso di strumenti casuali (come i dadi o la roulette), che inquinerebbero la contesa, due intelletti vengono contrapposti in una situazione così complessa che nessuno dei due può sperare di comprenderla completamente, ma sufficientemente analizzabile di modo che ciascuno dei due può sperare di sconfiggere l'altro. Il gioco è tanto profondo e sottile che ha permesso la nascita di giocatori professionisti, ed ha sopportato senza esaurirsi 200 anni di partite e di studi analitici intensivi. Tali caratteristiche rendono gli Scacchi un'arena naturale per i tentativi di meccanizzazione. *Se si potesse sviluppare un giocatore artificiale vincente, si potrebbe affermare di aver penetrato il nucleo dell'attività intellettuale umana.*"

Gli Scacchi permettono di studiare le procedure astratte che usano gli esseri umani quando prendono decisioni. Se riusciamo a far giocare bene un computer, possiamo sperare di insegnargli a ragionare "come noi", e forse meglio, grazie alla grande velocità con cui la macchina elabora immense quantità di dati. Alcuni problemi pratici che avrebbero tratto giovamento dallo studio del progetto di giocatori artificiali vennero elencati da Shannon in [15]:
- macchine capaci di progettare componenti elettronici;
- macchine capaci di sostituire le centrali telefoniche elettromeccaniche;
- macchine capaci di tradurre frasi da una lingua in un'altra;
- macchine capaci di decisioni strategiche in campo militare o economico;

- macchine capaci di orchestrare una melodia;
- macchine capaci di deduzione logica.

È davvero mirabile osservare come quasi tutti gli obiettivi proposti da Shannon siano stati raggiunti, e questo grazie anche al contributo delle ricerche sui giocatori artificiali.

2. PICCOLA STORIA

Già all'inizio degli anni '50 Shannon [15] e Turing [17] avevano descritto algoritmi per giocare a scacchi, ma non avevamo macchine adatte a realizzarli. Verso il 1955 A. Newell e H. Simon della *Carnegie Mellon University* iniziarono la progettazione del programma CP-1 (*Chess Player* 1). Il loro scopo era quello di costruire una macchina "intelligente", capace di dimostrare teoremi matematici o di progettare sistemi complessi. Newell e Simon decisero che gli scacchi fornivano un buon terreno di prova per progettare una "macchina intelligente". Nella sua autobiografia Simon racconta di come una mattina del gennaio del 1956 entrò in classe annunciando ai suoi studenti: "Durante le vacanze di Natale io e Newell abbiamo progettato una macchina che pensa" [16]. Negli stessi giorni scrive:

"... Allen Newell e io abbiamo fatto progressi sostanziali sulla macchina che gioca a scacchi, solo che per il momento non sa giocare a scacchi ma solo cercare e scoprire dimostrazioni di teoremi di logica simbolica ...".

Il programma che avevano progettato si chiamava Logic Theorist ed era scritto in IPL-2, un linguaggio di programmazione appositamente sviluppato. Il programma di scacchi venne effettivamente scritto solo qualche anno dopo, in collaborazione con C. Shaw, per cui venne chiamato NSS. Il programma di gioco era abbastanza sofisticato, ma l'uso di un linguaggio sperimentale di programmazione penalizzò talmente le prestazioni della macchina che occorreva un'ora per fare una mossa. A causa della sua lentezza, il programma venne usato per giocare una sola partita, ma comunque entusiasmò i suoi autori che ottimisticamente profetizzarono che entro 10 anni il Campione del Mondo sarebbe stato un giocatore artificiale.

Un altro programma famoso fu scritto al MIT di Boston da A. Bernstein e altri alla fine degli anni '50. Il programma di Bernstein usava una strategia che Shannon definì di tipo B: sceglieva in una posizione le sette mosse migliori usando alcuni principi generali, analizzava per ciascuna mossa le sette migliori risposte, e ripeteva il procedimento per ciascuna delle 49 posizioni così ottenute. Il programma veniva eseguito da un elaboratore IBM 704 ed impiegava circa otto minuti per analizzare $7^4 = 2401$ varianti prima di effettuare una mossa. L'esame di una singola posizione comportava l'effettuazione di 7.000 operazioni di macchina, che realizzavano alcune semplici valutazioni strategiche: bilancio del materiale, misura della mobilità dei pezzi, analisi del controllo dello spazio e valutazione del grado di sicurezza del Re. Il programma giocava come un principiante, ma venne enormemente pubblicizzato negli Stati Uniti, contribuendo a far conoscere al grosso pubblico scopi e prospettive dell'Intelligenza Artificiale.

Il MIT per diversi anni fu il più importante centro di ricerca sul gioco artificiale. Il principale ricercatore impegnato nel tentativo di costruire macchine intelligenti era J. McCarthy. Egli aveva ottenuto la disponibilità di uno dei primi mainframe IBM, un modello 704, che i suoi studenti programmavano usando un nuovo linguaggio, il LISP, inventato da McCarthy stesso. L'interesse di McCarthy nei programmi di scacchi era puramente teorico, ma alcuni suoi studenti si entusiasmarono del progetto e lo portarono avanti. Nel 1962 A. Kotok, uno di tali studenti, terminò di scrivere un programma di scacchi e lo presentò come tesi di laurea. Per ragioni di efficienza il programma era scritto in FORTRAN e assembler. Il programma di Kotok venne ulteriormente sviluppato da McCarthy che si era trasferito a Stanford, in California, e nel 1967 divenne il rappresentante americano in una famosa sfida contro una macchina sovietica. I sovietici avevano iniziato le loro ricerche da alcuni anni, e si avvalevano dell'aiuto di alcuni esponenti della loro grande scuola scacchistica. Era la prima volta che le ricerche sovietiche venivano allo scoperto, e il risultato fece sensazione: l'URSS vinse per 3-1 quella sfida, con due vittorie e due patte. L'incontro fu

particolarmente interessante dal punto di vista scientifico, perché i due programmi usavano strategie di ricerca opposte. Il programma sovietico usava una strategia di Shannon di tipo A (o "forza bruta"): esplorava tutte le varianti possibili fino alla profondità di 5 semimosse. Il programma americano usava una ricerca di tipo B: non tutte le varianti possibili venivano esplorate, ma solo quelle definite "plausibili" da una funzione euristica.
Negli anni della corsa alla Luna, una sconfitta tecnologica di questo genere venne vista negli USA come una vera e propria onta nazionale. Il risultato dell'incontro stimolò nuove ricerche, e R. Greenblatt costruì poco dopo una nuova macchina. Il programma si chiamava MacHack, usava un Digital PDP-6 del MIT ed ebbe l'onore di essere il primo giocatore artificiale a debuttare in un torneo a Boston nel 1967. La prestazione di MacHack al torneo fu pessima. Tuttavia, col suo debutto agonistico la lunga corsa verso il Campionato del Mondo era davvero cominciata: infatti al programma fu permesso di iscriversi alla federazione scacchistica americana (USCF), stabilendo così un precedente. A parte il risvolto commerciale, è importante che partecipando a tornei ufficiali un programma ottiene un punteggio Elo. MacHack raggiunse una valutazione Elo pari a 1500 punti: il livello di un dilettante di media forza. MacHack non riuscì mai a giocare bene, ma quando fu mostrato ad una conferenza che si tenne ad Edimburgo nel 1968 ebbe il merito di attirare l'attenzione di un giocatore scozzese, D. Levy, che era anche uno studioso di Intelligenza Artificiale. MacHack aveva scatenato l'euforia dei congressisti presenti, alimentando rosee speranze per il futuro. Fu allora che Levy, colpito dall'entusiasmo ingiustificato che avevano acceso le mediocri

Tecniche di Intelligenza Artificiale applicata agli scacchi

Nei programmi di scacchi sono state messe alla prova parecchie tecnologie tipiche dell'Intelligenza Artificiale.

Rappresentazione della conoscenza. La conoscenza scacchistica è di vario tipo. Una forma di conoscenza *dichiarativa* riguarda la rappresentazione dei pezzi e della scacchiera, e poi la formazione di concetti in base a dei principi (mettere al sicuro il Re) e la conseguente definizione di piani e tattiche di gioco (arrocca, per mettere al sicuro il Re). Una forma di conoscenza *procedurale* riguarda invece il modo in cui si calcolano le conseguenze di una serie di scambi di pezzi (conviene sempre iniziare gli scambi dal pezzo di valore minore). Per rappresentare questi tipi di conoscenze sono state usate varie tecniche, in particolare per la conoscenza dichiarativa la logica e i linguaggi logici come Prolog, mentre la conoscenza procedurale è stata rappresentate mediante euristiche, di solito all'interno della funzione di valutazione.

Esplorazione dello spazio degli stati. Esistono diversi algoritmi di esplorazione dello spazio degli stati di una partita a scacchi. Sono stati infatti inventati raffinamenti algoritmici che si comportano mediamente meglio di Alfabeta, ed altri se ne continuano ad inventare.

Machine Learning. L'apprendimento automatico viene usato per aumentare le conoscenze dei programmi specie nella fase di apertura. I cosiddetti libri di aperture vengono da tempo creati in modo automatico, a partire da partite giocate da Grandi Maestri. È interessante notare che ci sono stati molti tentativi di costruire programmi che apprendono giocando contro se stessi, ma sono tutti falliti (nessuno ha ottenuto un giocatore efficace usando questa tecnica).

Pattern recognition. Una tecnica importante in alcuni programmi è il riconoscimento di schemi di gioco, di solito allo scopo di applicare una strategia che in altre partite ha dato buona prova. Il riconoscimento avviene definendo prima degli schemi (in inglese: pattern, o chunks) associati a una o più euristiche da applicare quando viene rilevato lo schema nella posizione in analisi.

Algoritmi genetici. Questa tecnica permette di far evolvere i programmi di gioco, raffinandone le funzioni di valutazione. Vengono create alcune varianti della funzione di valutazione, e poi le si fanno "competere" usando un insieme di posizioni di test. Le funzioni migliori (quelle che risolvono meglio le posizioni di test) vengono poi leggermente alterate in modo casuale, ripetendo il ciclo. Alla lunga (dopo migliaia di cicli) si ottiene una sorta di "selezione naturale" evolutiva delle funzioni di valutazione. Gli algoritmi genetici hanno dato buona prova nel miglioramento delle funzioni di valutazione.

Reti neurali. Esistono programmi le cui funzioni di valutazione sono reti neurali. Tali programmi, giocando o osservando le partite di altri, sono in grado di sviluppare una "teoria subsimbolica" (cioè non basata su conoscenza dichiarativa) degli scacchi e di metterla alla prova. Questa tecnologia non ha dato buoni risultati nel caso degli scacchi.

prestazioni delle macchine che aveva esaminato, scommise che nessun programma sarebbe riuscito a batterlo nei 10 anni seguenti. Sfidare Levy divenne un punto d'onore per tutti i principali ricercatori. La scommessa venne accettata da due ricercatori presenti alla conferenza, J. McCarthy e D. Michie. Nel corso degli anni la scommessa venne ribadita ed altri ricercatori scommisero contro Levy; nel 1976 valeva in totale 1.250 sterline, che Levy vinse facilmente battendo nel 1978 il miglior programma dell'epoca.
La sfida di Levy ebbe il merito di spronare i membri della comunità scientifica, che decisero di incontrarsi annualmente per misurare i progressi conseguiti. Nel 1970 a New York iniziò infatti una competizione riservata ai giocatori artificiali che continua ogni anno ancora oggi, raccogliendo sempre una partecipazione molto qualificata: il torneo dell'*Association for Computing Machinery* (ACM), in seguito denominato torneo NACC (*North American Computer Championship*). Nel 1974, a Stoccolma, venne introdotta su suggerimento di Levy una nuova manifestazione ufficiale: il Campionato del Mondo per giocatori artificiali. La novità del torneo di Stoccolma fu la partecipazione di un programma sovietico, Kaissa, diretto discendente del programma che aveva vinto la sfida USA-URSS del 1967. Kaissa era stato scritto da M. Donsky e V. Arlazarov, dell'Istituto per le Scienze Sistemiche di Mosca. Ebbe l'onore di diventare il primo giocatore artificiale a fregiarsi ufficialmente del titolo di Campione del Mondo, vincendo tutte le partite. Questa prima affermazione però non venne seguita da altri successi. Infatti tre anni dopo, a Toronto, Kaissa non riuscì a ripetersi e giunse solo secondo dopo il programma americano Chess 4.6. Andò anche peggio nel 1980, nel torneo di Campionato del Mondo svoltosi a Linz: Kaissa giunse solo sesto, perché l'hardware impiegato era diventato obsoleto rispetto alle macchine più moderne a disposizione dei suoi competitori. La sconfitta fu dovuta soprattutto al fatto che il tempo di macchina in URSS era costosissimo, a causa della scarsità di calcolatori, quindi gli autori del programma non avevano potuto migliorarlo quanto i suoi concorrenti. Kaissa sparì del tutto dalle competizioni, e da allora nessun programma sovietico ha più partecipato ad alcuna competizione ad alto livello.

3. LA NUOVA GENERAZIONE: MACCHINE BASATE SU HW SPECIALE

In quegli stessi anni venivano gettate le basi teoriche e tecnologiche per la costruzione della nuova generazione di programmi di gioco, che cominciò a dominare i tornei all'inizio degli anni '80. Le novità tecnologiche più importanti della fine degli anni '70 furono due: l'introduzione del microprocessore, ed i grandi miglioramenti delle tecniche di progettazione di hardware specializzato (VLSI).
L'invenzione del microprocessore mise alla portata di tutte le borse un calcolatore personale; nel 1974 esistevano già i primi microprocessori per hobbysti. Nel 1977 venne commercializzata la prima scacchiera elettronica (Figura 2): si chiamava Chess Challenger, era stata progettata da R. Nelson e venne prodotta dalla Fidelity International. Costava poche centinaia di dollari e venne venduta in decine di migliaia di esemplari.
Per comprendere il significato economico di questa invenzione, basterà riportare alcune cifre. I sedici computer partecipanti al campionato mondiale di Toronto nel 1977 erano tutti grossi calcolatori; venne calcolato che il loro costo totale era di 40 milioni di dollari, e un'ora di gioco costava 10.000 dollari. Al

FIGURA 2
Fidelity Chess Challenger, la prima scacchiera elettronica

campionato del 2004 hanno partecipato invece normali personal computer che costano al più qualche migliaio di dollari.
L'altra grande novità che rivoluzionò profondamente la ricerca sui giocatori artificiali fu l'introduzione di hardware specializzato per la generazione di mosse. Mentre i primi giocatori artificiali utilizzavano calcolatori "normali", ovvero di uso generale, ad un certo punto si cominciò a costruire hardware progettato specificatamente per giocare. Nel 1977 uno studente di Berkeley, O. Babaoglu (oggi docente dell'Università di Bologna) progettò un circuito capace di funzionare come generatore di mosse. Poco dopo venne progettata nei Laboratori Bell da J. Condon e K. Thompson una nuova macchina, chiamata Belle. Nel 1972 i due ricercatori avevano scritto per uno dei primi sistemi Unix un programma che si era comportato senza infamia e senza lode ai vari tornei cui aveva partecipato. Dopo la pubblicazione della tesi di Babaoglu i due ricercatori decisero che valeva la pena di sviluppare una macchina concepita appositamente per gli scacchi, in cui lo sviluppo delle varianti veniva calcolato direttamente da alcuni circuiti in hardware.
Belle conseguì rapidamente in tornei ufficiali della USCF il grado di Maestro di scacchi (2200 punti Elo). Una prima pietra miliare nella storia del gioco artificiale era stata raggiunta.
I successi di Belle, e la simultanea nascita di un ricco mercato per le macchine commerciali, diedero finalmente una certa credibilità sportiva alle ricerche sul gioco artificiale. Tra il 1982 ed il 1985 vennero organizzate a Milano da M. Somalvico e B. Pernici tre conferenze sui giocatori artificiali intitolate *L'Intelligenza Artificiale ed il Gioco degli scacchi*. A Londra venne creata la *International Computer Chess Association* (ICCA), un'associazione di scienziati che si interessano di informatica scacchistica. Oggi ICCA ha cambiato nome in ICGA (*International Computer Games Association*) e si occupa di organizzare manifestazioni agonistiche per giocatori artificiali. È a cura dell'ICGA l'organizzazione delle conferenze *Computer Games* e *Advances in Computer Games*.
All'inizio degli anni '80 i progettisti di giocatori artificiali cominciarono ad esplorare le possibilità di una nuova tecnologia, quella del *calcolo parallelo*. L'idea del calcolo parallelo è ingannevolmente semplice: due processori sono meglio di uno perché possono effettuare una doppia quantità di calcoli. In realtà esistono grandi problemi di coordinamento delle operazioni: in molti casi non si ottiene nessun guadagno usando più calcolatori invece di uno solo, perché essi perdono molto tempo a cercare di mettersi d'accordo!
Nel 1983 a New York il Campionato del Mondo fu vinto da Cray Blitz, un programma che si avvantaggiò della potenza del calcolatore parallelo Cray-1, che era allora il sistema di elaborazione più potente del mondo. Cray Blitz fu il secondo giocatore artificiale capace di guadagnarsi il titolo di Maestro, e dominò la scena per la prima metà degli anni '80.
Sia Belle che Blitz usavano programmi basati sul concetto di *espansione cieca dell'albero di gioco* (strategia A): gli anni '70 erano stati dominati da programmi basati su espansione euristica, ma ormai l'hardware era diventato così veloce da permettere lunghe analisi per forza bruta. Nel 1985 però si affacciò sulla scena Hitech, un programma di H. Berliner, che si basava ancora su euristiche. Sin dal suo apparire Hitech guadagnò una valutazione record di 2220 punti Elo Usa, e subito sconfisse Cray Blitz. L'anno successivo, nel 1986, Hitech valeva già 2360 punti Elo. Per qualche tempo sembrò che l'approccio "intelligente" scelto da Berliner fosse quello giusto, lento ma sicuro verso il Campionato del Mondo assoluto. E invece nel 1987 apparve una nuova macchina basata sul principio dell'analisi per forza bruta, Chiptest. Era stata progettata da uno studente cinese, FH. Hsu, in collaborazione con uno studente di Berliner, M. Campbell. L'anno successivo Hsu e Campbell battezzarono il nuovo prototipo di ChipTest con un nuovo nome: Deep Thought, dal nome di un personaggio di un libro di fantascienza. Analizzando 700.000 posizioni al secondo grazie al suo generatore hardware, Deep Thought era in grado di analizzare tutte le varianti possibili ad una profondità media di 10 semimosse in apertura (con un libro di aperture estremamente ridotto) e 9 semimosse nel mediogioco.
Nel 1990 sia Hsu che Campbell vennero assunti dai laboratori di ricerca di IBM a Yorktown, e lì svilupparono la macchina che oggi conosciamo col nome di Deep Blue.

4. ANALISI DEL PROBLEMA

Analizziamo brevemente i dati del problema, e vediamo come è stato "risolto" dai progettisti di Deep Blue. Nel gioco degli scacchi:

- ci sono due avversari che alternano le mosse e conoscono in ogni istante le stesse informazioni, cioè hanno *informazione completa* sullo "stato del gioco";
- ad ogni turno di gioco le mosse ammesse dalle regole sono in numero *limitato* e ben definite;
- ogni partita *termina* con la vittoria di uno dei due giocatori, oppure in parità.

Nella posizione iniziale del gioco degli scacchi si hanno 20 mosse possibili per il Bianco. Per ogni mossa del Bianco il Nero può rispondere in 20 modi diversi, per un totale di 400 posizioni possibili dopo sole due semimosse. Per quanto gli scacchi siano un gioco finito, e quindi in teoria completamente analizzabile, siccome il numero medio di mosse in ogni posizione è circa 33 e una partita dura mediamente 40 mosse, ovvero 80 semimosse, un programma che volesse "risolvere" il gioco dovrebbe esplorare 33^{80} partite, un numero di 120 cifre! Immaginando un computer talmente veloce da poter analizzare un miliardo di mosse al secondo (un compito leggermente al di sopra della tecnologia attuale), servirebbero circa 10^{105} anni per analizzare tutte la partite possibili. E se ci si limita ad analizzare le posizioni, queste sono di meno, circa 10^{43}, ma certo è impensabile analizzarle tutte per poi memorizzarne l'esito.

La soluzione universalmente adottata nei programmi di gioco è di fermare la generazione delle mosse ad un certo livello di profondità dell'albero, proporzionalmente alla potenza di calcolo disponibile, di applicare un'euristica sulle foglie dell'albero e poi di valutare i nodi interni mediante l'**algoritmo minimax** (o meglio mediante la sua ottimizzazione chiamata algoritmo *alfabeta*) [3].

Un esempio del funzionamento di Minimax è descritto nelle figure 3 e 4. L'algoritmo inizia valutando con la funzione di valutazione tutte le posizioni del livello più profondo dell'albero. A questo punto l'algoritmo "trasmette" verso l'alto i valori delle posizioni successive usando la regola minimax.

Per esempio, nella figura 3 la posizione d, in cui deve muovere MAX, ottiene il valore 4, che è il massimo tra le valutazioni delle posizioni sottostanti. La posizione e ottiene il valore 5 per lo stesso motivo. La posizione b, in cui deve muovere MIN, ottiene invece il valore 4, che è il minimo tra i valori delle sue posi-

Algoritmo minimax

Tutte le possibili evoluzioni di una partita, a partire da una certa mossa, vengono rappresentate con un albero i cui nodi sono caratterizzati da un certo peso. L'elaboratore viene programmato per ricercare la miglior mossa possibile all'interno di una determinata strategia. Una possibile strategia è quella realizzata con gli algoritmi minimax: ogni mossa viene scelta in modo da ridurre al minimo il massimo dei vantaggi che l'avversario potrà conseguire. L'algoritmo *alfabeta* consente di velocizzare quello minimax, "potando" l'albero dei rami che risultebbe inutile esaminare.

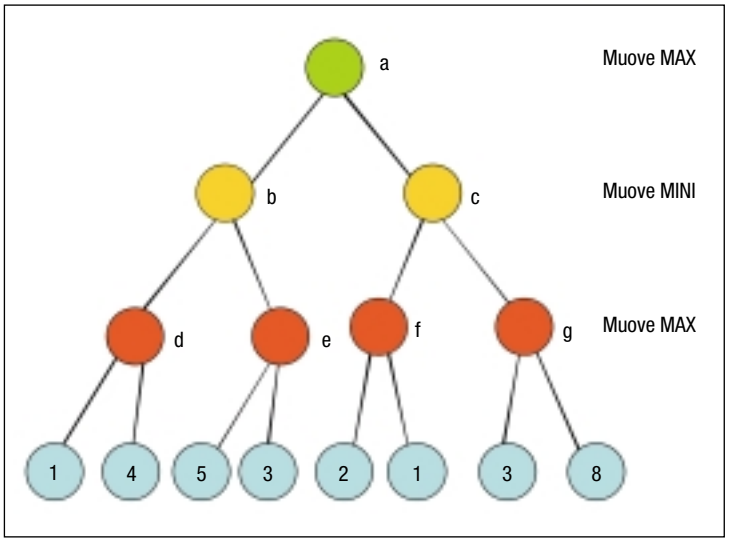


FIGURA 3
Situazione iniziale dell'algoritmo Minimax

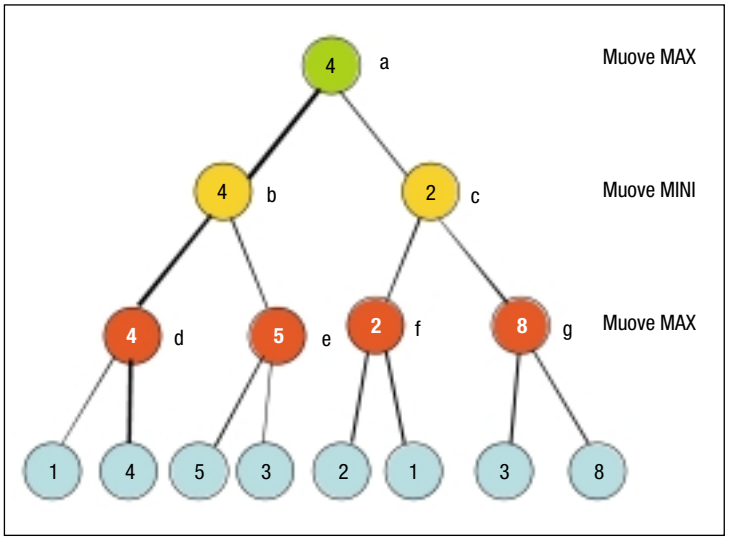


FIGURA 4
Risultato di Minimax; le linee in evidenza indicano la variante principale (a-b-d)

zioni sottostanti. La situazione finale, in cui la radice ha ottenuto un valore, è schematizzata nella figura 4.
La variante principale è quindi composta dalle mosse che producono in successione le posizioni a-b-d, e la mossa da giocare è quella che porta da a in b.
L'ipotesi grazie alla quale questo metodo funziona è che più profondo è l'albero da valutare, più è precisa la valutazione della mossa da giocare. Dal punto di vista teorico questa ipotesi non è mai stata dimostrata, ed anzi alcuni studi tenderebbero addirittura a dimostrare l'ipotesi contraria, che cioè più profondo è l'albero, meno precisa risulta la valutazione. In effetti in un albero molto grosso le posizioni del livello finale saranno molto diverse tra loro, e quindi sarà difficile paragonarle. D'altra parte i risultati empirici confermano che un programma X che usa uno hardware più veloce Y batte sistematicamente lo stesso programma che usa uno hardware meno veloce Z, perché esplora più approfonditamente ed estesamente l'albero di gioco.
A questo proposito K.Thompson effettuò un famoso esperimento con la sua macchina Belle. Fece giocare un torneo tra sei diverse versioni; l'unica differenza tra i programmi era la profondità di analisi dell'albero di gioco, che variava tra 4 e 9 semimosse. Ogni incontro comprendeva 20 partite. La tabella 1 riassume i risultati del torneo.
Come si vede, un incremento di profondità nell'analisi pari a una semimossa comporta in media un incremento nella forza di gioco di quasi 200 punti Elo. Si confronti questa tabella con quella ottenuta da un diverso esperimento di J. Schaeffer [13], che non variava la profondità di analisi del programma, ma solo la sua "conoscenza", cioè l'importanza della funzione di valutazione. Schaeffer effettuò una serie di esperimenti per valutare il bilanciamento relativo dei vari fattori componenti una funzione di valutazione. Egli usò il suo programma Phoenix per giocare un torneo cui partecipavano varie versioni del programma stesso, che differivano per la funzione di valutazione utilizzata. La versione di riferimento (che chiameremo versione 1), tentava semplicemente di massimizzare il materiale. Le altre erano via via arricchite come segue:

versione 2: mobilità e spazio: mosse pseudo-legali + controllo delle case nella metà avversaria della scacchiera.
versione 3: controllo del (grande) centro;
versione 4: struttura pedonale: penalizzazioni per pedoni isolati o doppiati;
versione 5: valutazione incrementale di mosse: catture di pedone verso il centro, torre in settima, sviluppo di un pezzo, arrocco;
versione 6: sicurezza del Re;
versione 7: case deboli e pedoni passati;
versione 8: pianificatore strategico.

Le otto versioni del programma giocarono lungamente tra di loro, ed i risultati sono riassunti nella tabella 2.
Confrontando i due esperimenti si deduce che si guadagna molto di più migliorando la velocità di analisi di un programma che non la sua conoscenza specifica della strategia scacchistica.

	P4	P5	P6	P7	P8	P9	Elo
P4	-	5	0.5	0	0	0	1235
P5	15	-	3.5	3	0.5	0	1570
P6	19.5	16.5	-	4	1.5	1.5	1826
P7	20	17	16	-	5	4	2031
P8	20	19.5	18.5	15	-	5.5	2208
P9	20	20	18.5	16	14.5	-	2328

TABELLA 1
Incremento di forza in funzione della profondità di analisi

versione 1: solo materiale	1110
versione 2: vers. 1 + mobilità e spazio	1420
versione 3: vers. 2 + controllo del centro	1530
versione 4: vers. 3 + struttura pedonale	1600
versione 5: vers. 4 + valutazione mosse	1630
versione 6: vers. 5 + sicurezza del Re	1750
versione 7: vers. 6 + case deboli e P passati	1760
versione 8: vers. 7 + pianificazione	1780

TABELLA 2
Confronto Elo tra diverse funzioni di valutazione

Infatti i grandi successi dei giocatori artificiali si sono verificati a partire dagli anni '80, quando i progressi delle tecnologie VLSI hanno permesso il progetto di hardware specializzato per la generazione di mosse [5].
Deep Blue aveva un'architettura parallela: un modello SP/2 con 24 processori, che controllano 512 processori specializzati per la generazione delle mosse. Ciascun processore specializzato analizza circa 1 milione di posiz/s, e complessivamente i 512 processori arrivano a circa 200 milioni di posiz/s utili. Questo permetteva a Deep Blue di esplorare alberi profondi 12/14 semimosse in circa 3 min, il tempo medio per mossa di torneo.

5. LO STATO DELL'ARTE

Un'importante misura dell'efficienza di un algoritmo parallelo è lo *speed-up* (accelerazione). Questo parametro è definito come il rapporto tra il tempo di esecuzione di una implementazione sequenziale *efficiente* di un algoritmo e quello di una sua versione parallela. L'obiettivo ideale è di ottenere valori per lo speedup che aumentino linearmente con il numero di processori usati. Purtroppo questo risultato è difficile da ottenere per gli algoritmi paralleli di ricerca su alberi di gioco. Esistono infatti diversi motivi di degrado delle prestazioni di questa classe di algoritmi che vengono indicati con il termine *overhead* (degradazione) [14].
L'*overhead in tempo* (OT) esprime una misura quantitativa della degradazione totale dell'algoritmo: è la perdita percentuale di speedup confrontata con lo *speed-up* ideale. È espressa come:

$$OT = (\text{tempo con n CPU}) \cdot \frac{n}{\text{tempo con 1 CPU}}$$

L' overhead totale è in realtà composto da più componenti:
❑ *overhead di ricerca* (OR): in ambiente sequenziale tutte le informazioni ricavate fino ad un dato punto della ricerca sono disponibili per effettuare decisioni quali il taglio di un sottoalbero.
In ambiente distribuito le stesse informazioni possono essere disperse su macchine diverse e quindi non efficacemente utilizzabili: ciò può portare ad una esplorazione non necessaria di una porzione dell'albero di gioco. La crescita delle dimensioni dell'albero è chiamata overhead di ricerca. Tale forma di degrado può essere approssimata osservando che la dimensione dell'albero esplorato è proporzionale al tempo di ricerca. OR è precisamente definito dalla relazione:

$$OR = \frac{\text{nodi visitati da n CPU}}{\text{nodi visitati da 1 CPU}} - 1$$

❑ *overhead di comunicazione* (OC): è il carico addizionale che un programma parallelo deve sopportare quando è impiegato un tempo non trascurabile per la comunicazione dei messaggi fra i processi. Questo costo può essere limitato in fase di programmazione scegliendo opportunamente le dimensioni e la frequenza dei messaggi. Una stima di OC è data dal prodotto fra il numero di messaggi inviati e il costo medio di un messaggio.
❑ *overhead di sincronizzazione* (OS): è il costo che occorre quando alcuni dei processori sono inattivi. In teoria tutti i processori dovrebbero essere occupati nello svolgere lavoro utile per tutto il tempo di esecuzione. Nella realtà questo comportamento ideale viene meno quando un processo deve sincronizzarsi con un altro determinando attesa attiva. Potrebbe accadere per esempio che un processo P voglia agire su un valore condiviso in mutua esclusione, mentre un altro processo Q sta già operando su tale valore. Il processo P rimarrà bloccato (e quindi inattivo) fintanto che Q non abbia completato la sua operazione. Tale overhead può presentarsi anche quando un processo è in attesa che certi risultati, senza i quali non può continuare, gli siano forniti da un altro.
L'overhead complessivo OT è funzione delle tre forme di overhead elencate:

$$OT = f(OR, OC, OS)$$

Per massimizzare le prestazioni di un algoritmo parallelo è necessario minimizzare i vari tipi di degradazione. Purtroppo essi non sono mutuamente indipendenti e lo sforzo nella minimizzazione di uno di essi può risultare nell'aggravio di un altro. Per esempio la riduzione dell'overhead di ricerca richiede nor-

Sistema	Posizioni/sec	HW
Ananse	6.000	Intel 486/66
MChess	10.000	Pentium 90
Amy	10.000	Sun Sparc10
Arthur	20.000	Sun Sparc20
Cray Blitz	750.000	Cray C90
*Socrates	1.000.000	CM5 / 512
DeepThought 2	4.000.000	IBM/6000
Deep Blue (1997)	200.000.000	IBM SP/2 (24 processori) + 512 chip speciali
Fritz, Junior, Shredder (2004)	3-5 milioni	Quadriprocessore su PC di ultima generazione

TABELLA 3
Numero di posizioni per secondo analizzate da alcuni sistemi

malmente un aumento del numero delle comunicazioni.
Esistono comunque diverse fonti di parallelismo nella ricerca di alberi di gioco le quali hanno originato approcci completamente differenti al processo di parallelizzazione della ricerca sequenziale:

- parallelismo nella valutazione statica dei nodi terminali;
- generazione parallela delle mosse;
- decomposizione dell'albero di gioco.

5.1. Parallelismo nella funzione di valutazione statica

Una considerevole porzione del tempo di ricerca è spesa nella valutazione statica dei nodi terminali. La qualità delle scelte strategiche operate durante il gioco è fortemente legata alla affidabilità e quindi alla complessità della funzione di valutazione. Spesso si sceglie di praticare una valutazione statica semplice e grossolana preferendo impiegare il tempo di ricerca nel condurre esplorazioni più in profondità nell'albero di gioco. Il tempo dedicato alla valutazione dei nodi terminali può tuttavia essere ridotto distribuendo tale operazione su più processori, ciascuno dedicato a valutare differenti termini della funzione di valutazione. I risultati del lavoro così partizionato saranno riuniti per originare un unico valore per la posizione esaminata.
Chiaramente lo speedup massimo ottenibile è limitato dalla scomponibilità e modularità della funzione di valutazione implementata.
Deep Blue fa uso di un hardware specifico per l'esecuzione della valutazione statica. In particolare esso consiste di due componenti denominati rispettivamente valutatore veloce e lento:

- l'hardware per la valutazione veloce aggiorna i termini della funzione di valutazione che possono essere calcolati in modo incrementale, cioè a mano a mano che vengono eseguite o retratte le mosse. Tale aggiornamento avviene in parallelo alle altre fasi di esplorazione interna dell'albero; quando sarà raggiunto un nodo terminale parte dei suoi termini saranno quindi già valutati.
- i restanti termini della funzione di valutazione sono calcolati dal valutatore lento quando la ricerca arriva ad un nodo terminale. Il loro calcolo è completato molto rapidamente poiché la risorsa di elaborazione ad esso dedicata utilizza un insieme di tabelle indirizzate via hardware. Ogni tabella contiene informazioni riguardanti proprietà salienti di una posizione aggiornate durante la ricerca.

5.2. Generazione in parallelo delle mosse

La generazione delle mosse è un'altra operazione molto frequente nella ricerca su un albero di gioco ed in generale incide pesantemente sul tempo globale di esecuzione. Negli scacchi la velocità di generazione delle mosse è il fattore che limita le dimensioni della ricerca poiché le regole che governano il movimento dei pezzi sono piuttosto complicate. Un accorgimento desiderabile è dunque la generazione in parallelo di tutte le mosse relative ad un nodo.
Ogni colonna della scacchiera potrebbe essere assegnata ad un diverso processore, che genererà tutte le mosse per tutti i pezzi disposti sulla propria colonna. Poiché ogni colonna può contenere più pezzi di un'altra è necessaria una forma di bilanciamento del carico, cioè un riassegnamento dei lavori per mantenere tutti i processori attivi. In questa tecnica un motivo di degrado delle prestazioni è dovuto al fatto che per molti nodi non è necessario generare tutte le mosse (a causa di tagli) e così parte del lavoro andrà perduto. Il metodo descritto si presta per una sua im-

plementazione in hardware, cioè che utilizzi componenti architetturali progettati *ad hoc*. HITECH [5] utilizzava hardware specifico per la generazione in parallelo delle mosse. In particolare veniva impiegata una matrice 8 × 8 di processori, uno per ogni casa della scacchiera: ciascun processore calcolava in parallelo agli altri le mosse legali per il pezzo che occupa la casa cui è associato (ammesso che essa non sia vuota). Il generatore implementava inoltre l'ordinamento delle mosse: quella suggerita è la mossa con maggiore priorità; la priorità di una mossa è una stima del suo valore ed è attribuita autonomamente dal processore che l'ha generata. Ogni processore deposita la priorità della sua migliore mossa su un bus accedibile da tutti gli altri: quando il processore con la mossa a più alta priorità riconosce che nessun altro dispone di mossa migliore, esso presenta la sua mossa al modulo dedicato al controllo della ricerca.

5.3. Decomposizione dell'albero di gioco

Il metodo di decomposizione divide l'albero in sottoalberi e stabilisce che sottoalberi diversi siano esplorati da processori in generale distinti. Tale approccio al parallelismo non prevede, in linea di principio, limitazioni per lo *speed-up*. In [4] sono confrontate le prestazioni di diversi algoritmi distribuiti di decomposizione dell'albero di gioco. La prospettiva di prestazioni elevate su architetture specializzate ha fatto sì che i maggiori sforzi nella ricerca siano stati concentrati nella direzione tracciata da questa classe di algoritmi paralleli. In effetti, questo è l'approccio scelto da Deep Blue [1].

5.4. I programmi commerciali

Quelle che abbiamo delineato sono le linee di ricerca più note, in quanto pubblicate su riviste scientifiche dagli autori dei programmi. Esistono però centinaia di programmi di vari autori e di diversa forza, alcuni dei quali hanno valore commerciale e di cui nulla si sa circa il funzionamento interno. I pincipali tra questi programmi si sfidano annualmente in un Campionato del Mondo organizzato da ICGA. Nel 2004 il Campionato si è svolto presso l'Università Bar-Ilan di TelAviv, con i risultati mostrati nella tabella 4.
Questa tabella è interessante perché sembra

Nome	Nazione	Hardware	Punti (su 11 incontri)
Junior	Israele	4proc 2.2GHz, Proliant HP	9
Shredder	Germania	4proc AMD 2.0 GHz, Transtec	8.5
Diep	Olanda	4proc AMD 2.0 GHz	7.5
Fritz	Olanda	4proc AMD 2.0 GHz, Transtec	7
Crafty	USA	4proc AMD 2.0 GHz	7
Jonny	Germania	AMD64 3200+	6.5
ParSOS	Germania	AMD64 3200+	6
Falcon	Israele	AMD64 3200+	6
ISIChess	Germania	AMD64 3400+	6
Deep Sjeng	Belgio	AMD64 3400+	5.5
Woodpusher	UK	Pentium 4 2.8 GHz	3
Movei	Israele	Pentium 4 2.8 GHz	3
The Crazy Bishop	Francia	Pentium 4 2.8 GHz	2
FIBChess	Spagna	Pentium 4 2.8 GHz	0

TABELLA 4
La classifica finale del Campionato del Mondo 2004

mostrare che l'elemento dominante anche con software diversi sia l'hardware.
I programmi commerciali sono così "ottimizzati" che ormai battono regolarmente i migliori giocatori umani. Per esempio nello scorso settembre 2004 all'IRST di Trento il migliore giocatore italiano, Michele Godena, ha sfidato il programma campione del mondo 2004, Deep Junior, creato in Israele da Bushinsky e Ban (Figura 5). La partita, vinta dal programma, è la seguente:

[Event "Uomo vs Computer"]
[Site "Povo - Trento ITA"]
[Date "2004.09.27"]
[Round "1"]
[White "Deep Junior"]
[Black "Godena, Michele"]
[Result "1-0"]
[PlyCount "89"]

1. e4 e5 2. Nf3 Nc6 3. Bb5 a6 4. Ba4 Nf6 5. 0-0 b5 6. Bb3 Bc5 7. a4 Bb7 8. d3 b4 9. c3 d6 10. a5 h6 11.d4 Ba7 12. Be3 0-0 13. Nbd2 bxc3 14. bxc3 exd4 15. cxd4 Nb4 16. Qb1 Rb8 17. Re1 d5 18. e5 Nd7 19. e6 fxe6 20. Bxh6 Qf6 21. Bg5 Qf7 22. Bh4 Rfe8 23. Bg3 Nf8 24. Ba4 Bc6 25. Ne5 Qf6 26. Ndf3 Bxa4 27. Rxa4 Nc6 28. Qd3 Nxe5 29. Bxe5 Qf5 30. Qxa6 Rb1 31. Ra1 Rxa1 32. Rxa1 Rb8 33. Rf1 Rb1 34. Qxa7 Rxf1+ 35. Kxf1 Qb1+ 36. Ke2 Qb5+ 37. Ke3 Qb3+ 38. Kf4 Qc2 39. h4 Qxf2 40. Qb8 Qxg2 41. a6 Qe2+ 42. Qxc7 Ng6+ 43. Kg3 Nxe5 44. Qd8+ Kh7 45. dxe5 1-0

FIGURA 5
Da sinistra: Shay Bushinky e Amir Ban, autori di Deep Junior

6. CONCLUSIONI

La ricerca sul gioco artificiale è ritenuta tuttora da alcuni studiosi di importanza capitale per lo sviluppo della teoria e delle tecnologie della Intelligenza Artificiale. Secondo Donald Michie:

"la costruzione di giocatori artificiali di scacchi, una ricerca di tipo tecnologico ma che ha una portata che va ben al di là della pura tecnologia, è oggi la ricerca scientifica più importante del mondo. Possiamo confrontarla con le ricerche fatte da T.Morgan a New York durante la Prima Guerra Mondiale sulla Drosophila. Quelle ricerche ebbero un impatto fondamentale sulle basi teoriche della genetica moderna. Ci accorgiamo oggi delle conseguenze industriali dell'ingegneria genetica, che è figlia delle ricerche di Morgan. Stiamo oggi usando l'analisi scientifica del gioco degli scacchi come studio preliminare per l'ingegneria della conoscenza".

Tuttavia, anche se l'Intelligenza Artificiale può guadagnare ancora molto dall'approfondimento di ricerche sui giochi, non c'è dubbio che il risultato di Deep Blue va considerato in primo luogo un successo della ricerca sul calcolo parallelo.
Il successo di Deep Blue è irripetibile, perché la macchina è stata smantellata. Negli ultimi anni ci sono stati altri incontri tra i più forti giocatori del mondo e i migliori programmi commerciali che giravano su normali personal computer. La maggior parte di questi incontri si è conclusa in parità. Dunque diciamo che al momento la superiortà delle macchine sugli umani nel gioco degli scacchi non è affatto netta. Certo però ormai sono poche decine gli umani che riescono a resistere alla forza di gioco delle macchine. Per esempio i migliori giocatori dell'Internet Chess Club sono da anni i computer: al momento domina il programma Shredder.
Occorre domandarsi come si orienterà la ricerca in futuro. Esistono molti giochi più complessi degli scacchi, ad esempio il Go o gli scacchi cinesi. Noi crediamo che siano i giochi ad informazione incompleta ad n giocatori quelli che debbono essere studiati in futuro: un esempio è il Risiko. Questi giochi creano situazioni molto complesse che sono poco studiate.

Miniglossario scacchistico

Albero di gioco
In un *gioco ad informazione completa*, è la rappresentazione astratta dello sviluppo di tutte le varianti possibili, ciascuna calcolata fino ad una posizione terminale. Negli scacchi l'albero del gioco è completamente sviluppabile in teoria ma non in pratica, a causa dell'*esplosione combinatoria* delle varianti.

Effetto orizzonte
Difetto intrinseco di un algoritmo di valutazione basato su una visita parziale dell'albero di gioco. Si rimedia in parte continuando ad espandere l'albero fino ad ottenere solamente posizioni quiescenti.

Elo, punteggio
Sistema di misura e classificazione della forza dei giocatori di scacchi. Introdotto dal prof. A. Elo negli Stati Uniti durante gli anni '60, venne ufficialmente adottato dalla FIDE nel 1970. Al punteggio Elo, che varia col variare dei risultati conseguiti in tornei ufficiali, è legato il conseguimento dei titoli nazionali e internazionali, secondo la seguente tabella indicativa: Maestro = 2200; Maestro Fide = 2300; Maestro Internazionale = 2400; Grande Maestro = 2500. Il sistema Elo locale di una nazione può differire da quello internazionale. In particolare, il sistema Elo della federazione americana USCF è leggermente inflazionato rispetto a quello internazionale. Per questo motivo i punteggi Elo dichiarati dai costruttori di scacchiere elettroniche sono solitamente riferiti all'Elo USCF.

Obiettivo
Situazione di gioco da raggiungere mediante un piano strategico.

Piano
Definizione dell'obiettivo e dei mezzi per raggiungerlo.

Ply
Vedi Semimossa.

Posizione quiescente
Posizione in cui non sono possibili scacchi o prese. Una posizione quiescente viene valutata dalla funzione di valutazione senza ulteriore sviluppo dell'albero di gioco.

Posizione terminale
Nel gioco degli Scacchi una posizione è terminale se uno dei due giocatori ha dato scacco matto all'avversario, oppure se non è più possibile il matto, e quindi la partita è patta (esempio: Re solo contro Re solo).

Posizione turbolenta
Posizione non quiescente. Tipicamente, una posizione in cui sono possibili scacchi o prese.

Semimossa
Movimento di un pezzo di uno dei due giocatori. Due semimosse consecutive, una per ciascun giocatore, costituiscono una mossa nel senso tradizionale.

Strategia
Definizione ed esecuzione di un piano di gioco a lungo termine allo scopo di conseguire un certo obiettivo. Nei commenti ad una partita la descrizione di una strategia è spesso generica e non dettagliata.

Tattica
Definizione ed esecuzione di piani di gioco a breve termine, solitamente subordinati alla strategia. Un piano tattico va elaborato dettagliatamente.

Variante principale
Risultato ottenuto dalla valutazione minimax di una posizione. Contiene la mossa da giocare ed il seguito più probabile previsto dalla macchina.

Zugzwang
Una posizione di zugzwang è una posizione in cui la parte che deve muovere non può che peggiorare le proprie possibilità, mentre invece se potesse "passare" la mossa non comprometterebbe nulla.

Bibliografia

[1] Anantharaman T., Campbell M, Hsu F.: Singular Extensions: Adding Selectivity to Brute Force Searching. *Artificial Intelligence*, Vol. 43, 1990, p. 99-110.

[2] Berliner HJ., Goetsch G., Campbell M., Ebeling C.: Measuring the Performance Potential of Chess Programs. *Artificial Intelligence*, Vol. 43, 1990, p. 7-20.

[3] Ciancarini P.: *I giocatori artificiali di scacchi*. Mursia, 1992.

[4] Ciancarini P.: Distributed Searches: a Basis for Comparison. *Journal of the International Computer Chess Association*, Vol. 17:4, 1994, p. 194-206.

[5] Ebeling C.: *All the Right Moves: A VLSI Architecture for Chess*. MIT Press, 1987.

[6] Finkel RA., Fishburn J.: Parallelism in Alpha-Beta Search. *Artificial Intelligence*,Vol. 19, 1982, p. 89-106.

[7] Hsu FH.: *Behind Deep Blue*. Princeton University Press, 2002.

[8] Marsland TA., Campbell M.: Parallel Search of Strongly Ordered Game Trees. *ACM Computing Surveys,* Vol. 14:4, 1982, p. 533-551.

[9] Marsland TA., Popowich F.: Parallel Game Tree Search. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 4:7, 1985, p. 442-452.

[10] McGrew T.: Collaborative Intelligence. *IEEE Internet Computing*, Vol. 1:3, 1997, p. 38-42.

[11] Moravec H.: When will computer hardware match the human brain?. *Journal of Evolution and Technology*, Vol. 1, 1998 (anche in Moravec H:, *ROBOT: Mere Machine to Transcendent Mind*, Oxford Univ. Press, 1998).

[12] Newborn M.: *Deep Blue: An Artificial Intelligence Milestone*. Springer, 2003.

[13] Schaeffer J.: *Experiments in Search and Knowledge*. PhD Thesis, University of Alberta, 1986.

[14] Schaeffer J.: Distributed Game-Tree Searching. *Journal of Parallel and Distributed Computing*, Vol. 6, 1989, p. 90-114.

[15] Shannon CE.: Programming a Computer for Playing Chess. *Philosophical Magazine*, Vol. 41:7, 1950, p. 256-275.

[16] Simon H.A.: *Models of My Life*. Basic Books, 1991.

[17] Turing AM.: Digital Computers Applied to Games. In *Faster Than Thought* (Ed. Bowden BV.), Pitman 1953, p. 286-295.

[18] Siti web: www.research.ibm.com/deepblue - sito di IBM su Deep Blue.
www.icga.org - sito dell'associazione internazionale di giochi e computer.
www.chessclub.com - internt Chess Club, dove molti giocatori sono computer

PAOLO CIANCARINI è Ordinario di Ingegneria del Software all'Università di Bologna. È Candidato Maestro di Scacchi, anche se non gioca più in torneo da parecchi anni. È uno degli organizzatori del Campionato del Mondo di Scacchi per computer che si terrà nel 2006 a Torino nell'ambito delle Olimpiadi degli Scacchi. È socio e consigliere di AICA, nonché membro del Comitato Scientifico di Mondo Digitale.
ciancarini@cs.unibo.it