UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMA
ZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA DELL'INFORMAZIONE

REINFORCEMENT LEARNING E LE SUE APPLICAZIONI: GLI SCACCHI

Relatore: Prof. Schenato Luca

Laureando: Callegaro Luca

ANNO ACCADEMICO 2022 – 2023

Data di laurea 14/11/2022

INDICE

INDICE DELLE FIGURE

Introduzione

Sin dai nostri primi passi, le prime esperienze che acquisiamo sono le conseguenze prodotte dalle nostre azioni nell'ambiente che ci circonda. Man mano che vengono ripetute si imparano le relazioni di causa ed effetto, cosa fare per non procurarci danno e quello invece che ci porta a raggiungere gli obiettivi fissati.

Imparare tramite le interazioni con l'ambiente circostante è l'idea su cui si fondano la maggioranza delle intelligenze artificiali. Tra queste, l'approccio del reinforcement learning si concentra maggiormente sui risultati che queste interazioni producono, assegnando una ricompensa per ogni passo compiuto nella direzione dell'obiettivo da raggiungere.

Lo scopo della tesi è quello di presentare e descrivere la strategia del reinforcement learning come metodo di risoluzione di problemi.
Il gioco degli scacchi è un' ottimo esempio in cui applicare questo approccio, dato che:

1) Il problema è ben definito, sia nelle operazioni permesse (le varie mosse), che l'obiettivo finale (lo scacco matto);
2) Non è né così semplice da risultare banale, né troppo complicato da non avere soluzioni apprezzabili;
3) Negli scacchi è presente il concetto di ragionamento, ciò che le intelligenze artificiali dovrebbero riuscire a riprodurre tramite modelli e, in questo caso, ricompense;
4) La struttura discreta degli scacchi la rende ottima da digitalizzare.

Nel primo capitolo di questo documento verrà introdotta la storia del Reinforcement Learning, da quali idee è nata e come si è sviluppata negli anni.
Questo metodo verrà poi approfondito nel capitolo successivo, descrivendo le caratteristiche matematiche e strategiche che lo caratterizzano.
Nel terzo capitolo verrà trattata la storia del rapporto tra le intelligenze artificiali e gli scacchi, fino ad arrivare nell'ultimo capitolo ad AlphaZero, un motore che, grazie al Reinforcement Learning, è arrivato ad essere al vertice della scena scacchistica mondiale, capace di vincere contro i migliori giocatori, sia reali che artificiali.

1.Storia del Reinforcement Learning

1.1 Programmazione Dinamica e Apprendimento

La strategia del Reinforcement Learning è nata dalla fusione di due grandi temi studiati nel secolo scorso.
Il primo tema riguarda il problema dell' *optimal control*, termine nato alla fine degli anni ‘50 per descrivere il problema di progettare un controllore per ottimizzare il comportamento di un sistema dinamico nel lungo periodo. Uno degli approcci a questo problema fu quello studiato da Richard Bellman nel 1957, approccio che utilizzava il concetto di stato del sistema e di funzione di valore, o “optimal return function”, concetti che usò per definire delle equazioni che furono le fondamenta di ciò che oggi viene chiamata programmazione dinamica, e che verranno chiamate nel prossimo capitolo equazioni di Bellman. Sempre nello stesso anno Bellman introdusse la versione stocastica discreta del problema noto come Processo Decisionale di Markov.

I concetti di programmazione dinamica e apprendimento rimasero separati per molti anni, probabilmente a causa delle diverse materie in cui venivano studiati e i diversi obiettivi a cui erano indirizzati. Tra i primi a collegare la programmazione dinamica con l’apprendimento fu Paul Werbos nel 1977 che propose un approccio approssimato alla programmazione dinamica, chiamato *heuristic dynamic programming.* Nel 1987 sostenne che un collegamento più saldo tra programmazione dinamica e apprendimento era necessario per comprendere meccanismi cognitivi e neurali. Nel 1989 il lavoro di Chris Watkins sul Reinforcement Learning diffuse i formalismi del Processo Decisionale Markoviano. Da qui questo collegamento venne studiato sempre più approfonditamente, particolarmente da Dimitri Bertsekas e da John Tsitsiklis che nel 1996 coniarono il termine *neurodynamic programming.*
Ogni progresso ottenuto nella programmazione dinamica è un passo avanti anche nel Reinforcement Learning, grazie a questa interconnessione naturale sviluppatasi tra i due argomenti negli ultimi decenni.

1.2 L'Apprendimento per Trial-and-Error

L'altro tema principale è l'apprendimento per tentativi, o Trial and Error, inizialmente studiato nella psicologia dell'apprendimento animale negli anni a cavallo del secolo scorso. Forse il primo a descrivere il concetto generale di apprendimento per tentativi fu Edward Thorndike che nel 1911 chiamò *Law of Effects* l'apprendimento rinforzato da parte degli animali per compiere delle scelte specifiche dopo un certo tipo di avvenimento.
L'uso del termine *rinforzo* venne utilizzato per esprimere il rafforzamento di un pattern comportamentale in risposta ad uno stimolo, un rinforzatore, proveniente dall'ambiente. Il rafforzamento porta ad un cambiamento nel comportamento dell'agente, in questo caso gli animali, che persiste anche quando l'azione dello stimolo termina.

La strategia di apprendimento per Trial and Error venne studiata sin da subito come una possibile strada per lo sviluppo delle intelligenze artificiali. Nel 1948 Alan Turing descrisse il design di ciò che egli chiamava *pleasure-pain system*, che sfruttava alcuni aspetti della Law of Effects. Nel 1952 Claude Shannon seguì i movimenti di un topo mentre tentava di sfuggire da un labirinto, registrando le scelte ottimali tramite dei magneti sul pavimento.
Negli anni '60 i termini *Reinforcement* e *Reinforcement Learning* vennero usati per la prima volta in campo ingegneristico. Nella sua pubblicazione "Steps Toward Artificial Intelligence" (1961), Marvin Minsky discusse molti problemi riguardante il Reinforcement Learning, tra cui il *problema dell'assegnamento dei crediti*: Come devono essere distribuiti i crediti del successo tra le varie decisioni che possono essere state coinvolte nel produrlo?

In questi anni venne sviluppato anche il *Supervised Learning*, un metodo parallelo al Reinforcement ma con l'attenzione spostata verso il riconoscimento di pattern rispetto all'apprendimento per tentativi. Questi due approcci simili generarono confusione negli articoli e libri di testo scritti in quegli anni, in cui capitava che il termine apprendimento per tentativi fosse associato ad algoritmi che imparavano grazie ad esempi. Negli anni '60 e '70 per questo motivo vennero fatte poche ricerche in vero stile Trial and Error.
Un'eccezione fu il lavoro svolto da Donald Michie nel 1961 e 1963; egli descrisse un semplice sistema che lavorava con la strategia dell'apprendimento per tentativi per imparare a giocare a tris chiamato MENACE (Matchbox Educable Naughts and Crosses Engine). Questo sistema consisteva nel riconoscere lo stato in cui si trovava la partita, e da qui scegliere la prossima mossa.

John Holland, dopo un lavoro nel 1975 in cui delineò una teoria generale su sistemi capaci di adattarsi grazie alle scelte effettuate, nel 1986 introdusse i *Classifier Systems*, dei sistemi basati sulla strategia del Reinforcement Learning in cui la novità fu l'inserimento di un algoritmo genetico (*Genetic Algorithm*), il cui scopo era quello di far evolvere il sistema.
La persona che diede il contributo maggiore al Reinforcement Learning fu Harry Klopf; secondo lui l'aspetto che mancava e che permetteva di distinguere efficacemente questa strategia dal Supervised Learning era la possibilità di controllare l'ambiente con lo scopo di ritrovarsi in situazioni più ottimali per raggiungere l'obiettivo finale.

Un grande contributo allo sviluppo del Reinforcement Learning fu lo studio su metodi di apprendimento a più intervalli temporali, anche detto Temporal-difference learning methods. Le origini di questa strategia derivano dalla nozione di *secondary reinforcers,* proveniente dal campo dell'apprendimento animale; in sostanza quando un secondary reinforcer, ossia uno stimolo, viene associato ad un primary reinforcer (fame o dolore), il primo assume caratteristiche simili a quest'ultimo. Il temporal difference method e il metodo trial-and-error method divennero inseparabili quando Klopf nel 1972 e 1975 investigò dei grandi sistemi di Reinforcement Learning suddividendoli in tanti sottosistemi, ognuno con i propri obiettivi e ricompense, in grado di rafforzarsi l'un l'altro.

1.3 Sviluppi Correnti e Futuri

Dagli anni '70 ad oggi il Reinforcement Learning ha trovato moltissime applicazioni e contemporaneamente moltissime sfide, soprattutto nei giochi da tavolo. Nel 1992 Tesauro fu in grado di implementare questa strategia nel suo programma TD-Gammon, che fu in grado di raggiungere il *Master Level* nel gioco di Backgammon. La IBM sviluppò il software DeepBlue con il solo scopo di giocare a scacchi e nel 1997 riuscì a battere il campione mondiale Garry Kasparov. Questo software però utilizzava un albero di ricerca molto complesso, che richiedeva moltissimo spazio e tempo di elaborazione per il calcolo delle future mosse. Dopo anni di tentativi di insegnare ad un motore ad imparare a giocare a Go, un gioco molto più complesso degli scacchi, nel 2016 il software AlphaGo sviluppato da Google

riuscì a sconfiggere il campione mondiale, il coreano Lee Sedol. A differenza di DeepBlue, AlphaGo utilizza una combinazione di simulazioni di Monte Carlo, albero di ricerca di Monte Carlo, ottimizzazione di Bayes e osservazione diretta delle partite dei migliori giocatori mondiali, in modo da in modo da aumentare sempre più il livello contro cui sfidarsi.

Al giorno d'oggi gli ambiti in cui il Reinforcement Learning può venire applicato sono numerosissimi, come ad esempio: controllo dei semafori, chimica, pubblicità, robotica, giochi. Lorica nel 2017 divise le potenzialità di questa strategia in 3 temi di applicazione, che possono essere adoperati da qualsiasi azienda: per ottimizzare, per il controllo, e per il monitoraggio e mantenimento.

A differenza di forme di Intelligenza Artificiale però, il Reinforcement Learning non spiega il perché delle sue scelte, ed il modo in cui raggiunge il suo obiettivo non sempre è chiaro agli occhi dell'uomo. Considerando la visione del pubblico non è facile quindi affidare vite umane ad un motore che lavora con questo tipo di strategia, soprattutto l'apprendimento per tentativi. Un esempio potrebbe essere l'auto a guida autonoma di Nvidia, la cui particolarità è quella di non aver mai imparato direttamente a guidare ma si muove per strada osservando il comportamento degli altri veicoli.
Il reinforcement Learning ha quindi bisogno di alcuni miglioramenti nell'ambito della interpretabilità e responsabilità prima di poter essere applicato ad aspetti più delicati e che richiedono fiducia da parte delle persone.

2. Reinforcement Learning

2.1 Il Problema Del Reinforcement Learning

Il problema classico che contraddistingue il Reinforcement Learning ha come caratteristica principale massimizzare un contatore numerico tramite una serie di scelte compiute all'interno di un ambiente, di cui ognuna porta con sé un valore, chiamato ricompensa o reward.
Questi problemi possono essere modellati con dei loop chiusi, poiché ogni azione intrapresa influenza i futuri input. Al motore, o agente, questi comandi non vengono introdotti dall'esterno, ma è lui stesso che prova ad interagire con l'ambiente e valutare le varie opzioni disponibili, in modo da capire qual è l'insieme di scelte che portano alla ricompensa maggiore nel lungo periodo.

Il bilanciamento tra l'esplorazione e l'uso delle informazioni già acquisite dall'esperienza (*exploration and exploitation*) è un problema caratteristico che insorge nel reinforcement learning, ed è anche uno tra i suoi maggiori punti di forza. Il motore deve utilizzare le conoscenze già in memoria in modo da riconoscere i modelli ed ottenere velocemente le ricompense migliori nel breve termine, ma deve anche esplorare strade nuove, così da compiere scelte migliori nel futuro.

A differenza di altre intelligenze artificiali che dividono il problema a cui vengono poste in vari sottoproblemi da risolvere separatamente, il reinforcement learning ha un solo obiettivo da raggiungere, ossia quello finale. Tutti gli agenti che utilizzano questa strategia hanno davanti degli obiettivi espliciti, sono sensibili all'ambiente che li circonda e sanno compiere scelte che influenzano questo ambiente.

Alcuni esempi in cui la strategia del reinforcement learning può venire applicata potrebbero essere:

- Un giocatore di scacchi che compie una mossa. La scelta viene fatta sia anticipando le possibili risposte, che considerando la posizione che si andrebbe a raggiungere, se è favorevole o meno.
- Un cucciolo di gazzella che in poche decine di minuti dalla sua nascita riesce già a correre sulle sue zampe.

- Un robot casalingo che sceglie se entrare in una nuova stanza in cerca di sporco da raccogliere o cominciare a tornare verso la sua stazione di ricarica, in base a quanta batteria gli rimane.

Ognuno di questi esempi comprende l'interazione tra un agente e il suo ambiente, nel quale prova a raggiungere l'obiettivo, analizzando e prendendo scelte.

2.2 Gli Elementi del Reinforcement Learning

Il sistema del reinforcement learning è composto, oltre che da agente ed ambiente, da quattro principali sottoelementi: una policy, un segnale di reward, una funzione di valore, e, facoltativamente, un modello dell'ambiente stesso.

La policy definisce il modo in cui un agente impara ed il suo comportamento in un certo istante. Una policy può essere vista come il collegamento tra gli stati percepibili dall'ambiente e le azioni da prendere quando si è in tali stati. In alcuni casi la policy può essere espressa come semplice funzione o tabella di lookup, mentre in altri può coinvolgere una computazione estensiva. La policy è il cuore di un agente poiché da sola determina il suo comportamento. In generale, le policy possono essere deterministiche, ovvero dipendere solamente dallo stato, o stocastiche, ovvero definite attraverso la distribuzione di probabilità sulle azioni, dato uno stato.

Un segnale di reward definisce l'obiettivo in un problema di reinforcement learning. In ogni time step, l'ambiente invia all'agente un valore numerico chiamato reward, o ricompensa. L'unico obiettivo dell'agente è quello di massimizzare il reward totale che riceve nel lungo periodo. Il segnale di reward definisce così quali sono i buoni ed i cattivi comportamenti per l'agente. Questo segnale è la base di partenza per modificare in modo appropriato la policy; se l'azione selezionata dalla policy è seguita da un reward basso, allora la policy potrebbe cambiare per selezionare qualche altra azione in una situazione futura, influenzando la ricompensa in modo diretto con le sue scelte, o indiretto, cambiando lo stato dell'ambiente.

Mentre il reward indica cosa è buono nell'immediato, la funzione di valore specifica cosa è buono nel lungo termine. Il valore dello stato è l'ammontare totale dei reward che un agente si aspetta di accumulare nel futuro, partendo da quello stato. Il valore quindi determina la preferibilità a lungo termine degli stati dopo aver considerato gli stati migliori da seguire e i reward disponibili in questi stati. Per esempio un'azione compiuta in un certo stato potrebbe sempre portare un reward

immediato basso ma avere ancora un valore alto poiché è regolarmente seguito da altri stati che portano reward alti, e viceversa. Per fare un'analogia umana, i reward sono come il piacere (reward alto) o il dolore (reward basso).
I reward sono in un certo senso, elementi primari, mentre i valori, come predizione dei reward, sono secondari. Senza reward non ci possono essere valori e l'unico scopo di stimare i valori è quello di raggiungere reward più alti. Tuttavia, sono i valori gli elementi di confronto utilizzati quando prendiamo e valutiamo le decisioni. Le scelte delle azioni sono fatte sulla base di giudizi sui valori. Si cercano azioni che producano stati a massimo valore, non a massima ricompensa, perché queste azioni ottengono maggior ricompensa nel lungo periodo. Sfortunatamente, è molto più difficile determinare i valori che determinare i reward. I reward sono dati direttamente dall'ambiente mentre i valori devono essere stimati e ri-stimati dalle sequenze di osservazioni che un agente compie durante la sua intera esistenza. La componente più importante di quasi tutti gli algoritmi di reinforcement learning è un metodo per stimare in modo efficiente i valori.

Il quarto ed ultimo elemento è il modello dell'ambiente. Per modello si intende un'entità in grado di simulare il comportamento dell'ambiente. Per esempio, data una coppia stato-azione, il modello può predire il risultato della prossima coppia stato-azione. I modelli sono usati per pianificare, ovvero decidere prima quali azioni saranno eseguite sulla base degli stati che potranno essere raggiunti.

2.3 Il Problema dell' n-Armed Bandit

Il problema dell'n-armed bandit (un'analogia alla leva della slot machine) ha la seguente forma caratteristica: si consideri di dover effettuare ripetutamente una scelta tra n diverse opzioni, o azioni. Dopo ogni scelta si riceve un reward numerico che dipende da essa; l' obiettivo è massimizzare il reward dopo un lungo periodo, ad esempio dopo 1000 azioni selezionate, anche dette time steps. Potendo conoscere le rewards di ogni azione sarebbe logico pensare che scegliendo sempre quella con la ricompensa maggiore anche il reward finale sia il maggiore possibile. Questo tipo di scelta viene detta *greedy*, o avida, e si basa sullo sfruttare (exploiting) le conoscenze già possedute dall'agente. Se invece viene scelta un'azione diversa rispetto a quella con il reward maggiore si sta esplorando (exploring). Quest' ultima strada restituisce reward minori nel breve periodo, ma può portare a stati dell'ambiente che rendono possibili scelte ancora non esplorate, e una ricompensa finale che può risultare maggiore rispetto ad un procedimento caratterizzato da scelte solamente greedy.

Un possibile bilanciamento tra exploitation ed exploration è scegliere azioni greedy la maggior parte del tempo, e ogni tanto, con una piccola probabilità ε, prendere casualmente una qualsiasi delle altre strade. Questo metodo viene chiamato ε-greedy.
Il grafico riportato in seguito mostra come diverse scelte di ε portino a risultati molto diversi. In questo esempio è stato usato un set di 2000 problemi, ognuno con un diverso compito, di n-armed bandit generati casualmente, con n=10. In ogni step, l'agente aveva a disposizione 10 possibili strade, di cui conosceva la ricompensa generata da ognuna. Facendo la media dei risultati ottenuti nei vari problemi è stato possibile tracciare il grafico delle performance e del comportamento dei vari metodi mentre migliorano con l'esperienza.

La figura 1 compara il rendimento di metodo greedy con due metodi ε-greedy (ε=0.1, ε=0.01) nel tipo di esperimento descritto precedentemente. Il grafico superiore mostra l'aumento di reward ottenuto grazie all'esperienza. Il metodo greedy ha dei risultati notevolmente peggiori nel lungo periodo perché si trova spesso a compiere azioni non ottimali. Il grafico inferiore mostra che il metodo greedy trova più difficilmente le azioni ottimali, mentre i metodi ε-greedy, grazie alla continua esplorazione, hanno delle performance migliori. Tra questi, il metodo con ε=0.1 esplora di più e trova l'azione migliore più facilmente, ma proprio a causa di un ε elevato è alta la possibilità che non la scelga. Il metodo con ε=0.01 migliora più lentamente ma nel lungo andare avrà risultati migliori del' altro metodo ε-greedy.

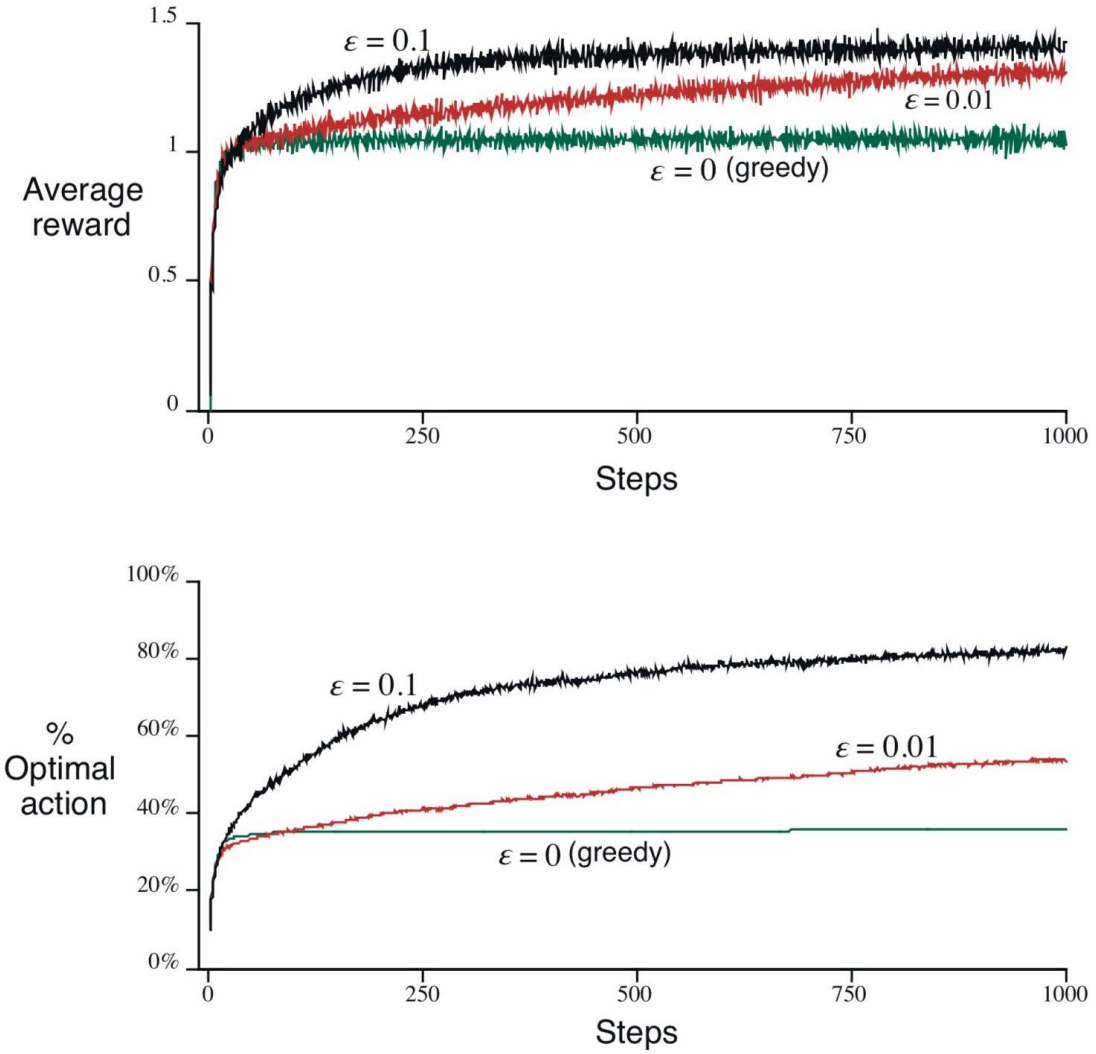

Figura 1: performance medie di vari metodi ε-greedy su 2000 problemi n-armed con n=10;

2.4 Processo Decisionale di Markov

L'agente e l'ambiente interagiscono tra loro ad intervalli di tempo discreti, t=0, 1, 2, 3, …. Ad ogni intervallo l'agente riceve una rappresentazione dello stato dell'ambiente, $S_t \in S$, dove S è l'insieme

dei possibili stati, e su questa base sceglie un'azione, $A_t \in A(S_t)$, dove $A(S_t)$ è l'insieme di azioni disponibili nello stato S_t. Nell' intervallo di tempo successivo l'agente riceve una ricompensa numerica $R_{t+1} \in R \subset \Re$, ritrovandosi in un nuovo stato S_{t+1}.

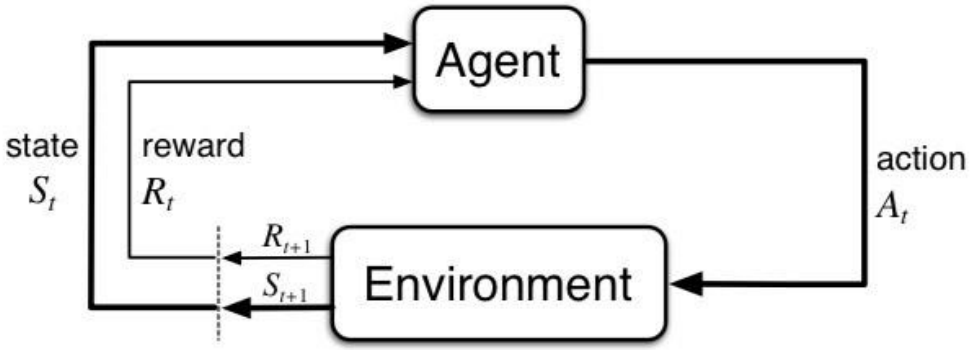

Figura 2: interazione agente-ambiente.

Ad ogni interazione l'agente crea una collegamento tra uno stato e la probabilità che ogni azione venga presa dato lo stato. Questo collegamento viene chiamato policy π_t, dove $\pi_t(a|s)$ indica la probabilità che $A_t = a$, se $S_t = s$.

2.4.1 La Proprietà Markoviana

Definita informalmente, la proprietà Markoviana di uno stato indica la capacità di un agente (che si trova in questo stato) di distinguere, riassumere e memorizzare le informazioni utili dall'insieme degli stati in cui si è trovato precedentemente.
In generale, la probabilità che un agente prenda una certa decisione dato un certo stato deve tenere conto di tutto ciò che è successo precedentemente, ossia:

$$\Pr\{R_{t+1}=r, S_{t+1}=s \mid S_0, A_0, R_1, \ldots, S_{t-1}, A_{t-1}, R_t, S_t, A_t\}.$$

Se lo stato possiede la proprietà Markoviana, tutto ciò che succede all'istante t+1 dipende solo dalle caratteristiche di stato e azione intrapresa allo stato t:

$$p(s', r|s, a)=\Pr\{R_{t+1}=r, S_{t+1}=s' \mid S_t=s, A_t=a\}$$

per ogni r, s, S_t e A_t.

2.4.2 Valore di Ritorno

L'obiettivo di un agente di RL è quello di scegliere una policy che massimizzi la somma dei reward attesa. La somma dei reward viene chiamata valore di ritorno (G_t) ed è data da:

$$G_t = r_t + r_{t+1} + r_{t+2} + ... + r_{N-1}$$

Per compiti a valore continuo, viene definito il discount return che è dato da:

$$G_t = r_t + \gamma\, r_{t+1} + \gamma^2 r_{t+2} + ... = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$$

dove $\gamma \in [0,1)$ è chiamato fattore di discount.

2.4.3 Funzione Valore

Per decidere quale azione intraprendere in un certo istante, è importante per l'agente conoscere quanto è "buono"/ottimale essere in un particolare stato. Un modo per misurare questa bontà dello stato è la funzione valore. Viene definita come la somma dei rewards attesa (E_π) che l'agente riceverà mentre segue una particolare policy π partendo da un particolare stato s. La funzione valore, $V_\pi(s)$ per la policy π è data da:

$$V_\pi(s) = E_\pi(G_t \,|s_t = s) = E_\pi(\sum_{k=0}^{\infty} \gamma^k r_{t+k} \,|s_t = s).$$

Similmente, una funzione azione-valore, chiamata anche Q-function, può essere definita come la somma dei rewards prevista mentre si intraprende un'azione a in uno stato s, seguendo la policy π. La funzione azione-valore $Q_\pi(s, a)$ è definita come segue:

$$Q_\pi(s, a) = E_\pi(G_t \,|s_t = s,\ a_t=a) = E_\pi(\sum_{k=0}^{\infty} \gamma^k r_{t+k} \,|s_t = s,\ a_t=a).$$

2.4.4 Equazioni di Bellman

Le equazioni di Bellman formulano il problema di massimizzazione della somma dei reward attesa in termini di relazione ricorsiva con la funzione valore. Una policy π è considerata migliore di un altra policy π^* se il ritorno atteso di quella policy è maggiore di π^* per tutti gli $s \in S$, che implica $V_\pi(s) \geq V_{\pi^*}(s)$ per tutti gli $s \in S$. Quindi la funzione valore ottimale $V_*(s)$ può essere definita come

$$V_*(s) = \max_\pi V_\pi(s),\ \forall s \in S.$$

Similmente, la funzione azione-valore ottimale $Q_*(s, a)$ può essere definita come

$Q_*(s, a) = \max_\pi Q_\pi(s, a), \ \forall s \in S, a \in A.$

Inoltre, per una policy ottimale, possiamo definire l'equazione

$V_*(s) = \max_{a \in A(s)} Q_{\pi*}(s, a).$

Sistemando le equazioni si può ottenere

$$V_*(s) = \max_a \sum_{s'} p(s'|s, a)[r_t + \gamma V*(s')]$$

dove $p(s'|s, a)$ indica la possibilità di arrivare allo stato s' partendo dallo stato s e compiendo l'azione a. Quest'ultima equazione è conosciuta come equazione di ottimalità di Bellman per $V_*(s)$. Mentre per Q_* l'equazione di ottimalità è

$$Q_*(s, a) = \sum_{s'} p(s'|s, a)[r_t + \gamma \max_{a'} Q_*(s', a')]$$

3. Rapporto tra Intelligenza Artificiale - Scacchi

Il gioco degli scacchi è tra le prime sfide in cui le varie intelligenze artificiali si sono cimentate, grazie alle sue regole semplici e alla sua caratteristica complessità, che richiede un'elevata capacità di vedere in profondità e simulare in poco tempo i possibili scenari a partire da varie posizioni iniziali.

3.1 Scacchi e AI nella Storia

I primi approcci nell'affrontare il gioco degli scacchi con l'uso di un elaboratore si verificarono nel 19° secolo: nel 1825 un dispositivo chiamato *Maelzel Chess Automaton*, inventato da W. von Kempelen nel 1769, venne introdotto negli Stati Uniti, suscitando scalpore. Furono in molti a scrivere studi ed articoli, incluso Edgar Allan Poe (*Maelzel's Chess Player*, 1836), con lo scopo di proporre una spiegazione su come funzionasse questo dispositivo rivoluzionario. La maggior parte degli autori concluse correttamente che l'Automaton era manovrato da un maestro di scacchi nascosto al suo interno.

Un tentativo più onesto nel progettare una macchina che giocasse a scacchi venne ideato da L. Torres y Quevedo nel 1912, che costruì un dispositivo capace di giocare finali di re e torre contro re. Il motore giocava con re e torre e forzava lo scacco matto in poche mosse qualsiasi fosse la posizione iniziale delle pedine. Il problema è relativamente semplice, dato che il risultato è ottenibile semplicemente seguendo un insieme di regole, ma l'idea era molto avanzata per l'epoca.

Il primo software a partecipare in un torneo fu *Mac Hack* nel 1967, sviluppato da Richard Greenblatt. Questo programma conosceva, oltre alle regole basilari, le prime 5 mosse delle 20 aperture più diffuse nelle partite ad alto livello. Ed in questo evento si verificò anche la prima vittoria dell'intelligenza artificiale sull'uomo:il primo giocatore sconfitto da un programma fu il filosofo americano Hubert Dreyfus, che perse per scacco matto alla 37esima mossa.

Nel 1979 Ken Thompson sviluppò un motore chiamato Belle, capace di generare fino a 180'000 per secondo e raggiungere una profondità di analisi di 9 mosse e mezzo. Nel 1983 Belle fu il primo programma ad essere premiato con il titolo di *National Champion* dalla US Chess Federation.

Nel 1989 Feng-hsiung Hsu, detto Crazy bird per il suo stile eccentrico, si unì a Murray Campbell della IBM per studiare e ricercare il concetto di *parallel computing*, ossia un tipo di architettura in cui diversi processori compiono semplici operazioni contemporaneamente in modo da risolvere un problema complesso. Da questo lavoro nacque il supercomputer scacchistico Deep Blue, capace di battere il campione mondiale nel 1997.

3.2 Digitalizzare gli scacchi

Lo stato di una partita di scacchi può essere definita con i seguenti dati:

1) Lo stato della posizione di ogni pedina sulla scacchiera
2) Un segnale che indica a chi tocca compiere la prossima mossa, se il bianco o il nero
3) Un segnale che indica se il re e le torri si sono mosse. Questo indica la possibilità di poter arroccare o meno in futuro.
4) Un segnale che indica l'ultima mossa compiuta. Può essere importante per esempio nel caso in cui si voglia compiere una cattura *en passant*, dato che si ha la possibilità di catturare in questo modo solo la prima mossa successiva al movimento di pedone dell'avversario.
5) Un segnale che indica il numero di mosse fatte dall'ultima cattura o spostamento di pedone. Questo è necessario per la regola della patta dopo 50 mosse.[1]

Da ogni posizione inoltre si può ricavare il vantaggio di un giocatore rispetto ad un altro, ossia:

1) Una posizione vincente per il bianco. Il bianco può forzare la vittoria, in qualsiasi modo il nero difenda.
2) Una posizione pari. Il bianco può forzare almeno un pareggio, in qualsiasi modo il nero giochi, e allo stesso modo il nero può forzare un pareggio, in qualsiasi modo il bianco giochi. Se ogni giocatore gioca correttamente la partita finirà in pareggio.
3) Una posizione vincente per il nero. Il nero può forzare la vittoria, in qualsiasi modo il bianco difenda.

Ciò che ha richiesto anni prima che i computer potessero essere un degno avversario contro sfidanti umani era la scelta della miglior mossa da prendere.

[1] La regola delle cinquanta mosse negli scacchi afferma che un giocatore può dichiarare la partita patta se, nelle ultime cinquanta mosse consecutive, non vi è stata nessuna cattura e nessuna mossa di pedone.

Il primo articolo che parlava di questo argomento fu scritto nel 1950 da Claude Shannon[2]. Prima ancora che i primi programmi per giocare a scacchi uscissero, lui predisse le due principali strategie di ricerca che verranno successivamente usate, e le chiamò "Tipo A" e "Tipo B".

Programmi di Tipo A avrebbero usato un approccio a *forza bruta*, esaminando ogni possibile posizione per un certo numero di mosse avanti usando un semplice algoritmo Minimax[3]. Shannon credeva che questo approccio fosse irrealizzabile per due motivi:
Il primo era la dimensione dell'insieme di mosse da valutare: con in media trenta mosse possibili in ogni posizione, lui calcolò che anche con processori capaci di elaborare milioni di posizioni ogni secondo (elaboratori esistiti solo 40 anni dopo), per valutare 3 mosse avanti per ciascun giocatore ci sarebbero voluti decine di minuti.
Il secondo era il problema della quiescenza: il programma avrebbe effettuato una valutazione solo alla fine di una serie di scambi di pezzi o altre importanti sequenze di mosse. Valutare anche le mosse "quiescenti", ossia preparatorie o posizionali, avrebbe aumentato di molto il tempo di elaborazione.
Questo portò naturalmente a ciò che viene chiamato *ricerca selettiva* (selective search) o i programmi che Shannon chiamò di Tipo B; usando la conoscenza del gioco degli scacchi per ogni posizione si possono limitare le scelte a poche mosse ottime. Questo rendeva possibile guardare "in profondità" le future mosse in tempi più ragionevoli. Nei successivi 25 anni però non si riuscì a trovare un algoritmo capace di identificare le mosse migliori.
Nel 1974 la ricerca a forza bruta fu implementata per la prima volta nel programma Chess 4.0 della Northwestern University. Con questo approccio ogni strada veniva esaminata senza scartarne nessuna; i ricercatori in questo modo hanno notato che il tempo necessario per costruire l'albero delle possibili future scelte al seguito di ogni mossa era molto minore rispetto al tempo richiesto per analizzare e scegliere le mosse migliori, e il beneficio di non scartare per errore le mosse migliori (cosa che succedeva con i metodi di Tipo B) portava a risultati decisamente migliori.
Negli anni '80 e '90 furono infine compiuti progressi nel campo della ricerca selettiva con lo sviluppo della ricerca quiescente ed altri metodi selettivi euristici[4] moderni. Questi nuovi algoritmi erano in grado di compiere molti meno errori rispetto ai precedenti, permettendo ai metodi di Tipo

[2] Programming a Computer for Playing Chess, 1950,
[3] Algoritmo che valuta la propria posizione e quella dell'avversario, cercando in ogni mossa di massimizzare la propria valutazione e minimizzare quella avversaria.
[4] Procedimento Euristico: procedimento non rigoroso, o intuitivo, analogico, che consente di prevedere un risultato, che dovrà poi essere convalidato in seguito. (Treccani)

B di esprimere il loro potenziale di ricerca in profondità, e di essere preferibili ai metodi a forza bruta.
Nel 2006 Rémi Coulom creò il *Monte Carlo tree search*, un metodo di ricerca selettiva di tipo B. Nel 2007 Levente Kocsis e Csaba Szepesvàri modificarono questo metodo chiamandolo *Upper Confidence bounds applied to Trees*, o UCT in breve. Nel 2011 Chris Rosin sviluppò una nuova versione dell'UCT chiamata Predictor + Upper Confidence bounds applied to Trees, o PUCT in breve. PUCT fu poi usato in AlphaZero nel 2017, e successivamente in Leela Chess Zero nel 2018.

3.3 Monte Carlo Tree Search

Nei giochi si usa ciò che viene chiamato albero del gioco, in cui ogni nodo rappresenta uno stato della partita e i suoi nodi figli sono i vari possibili futuri stati che si possono ottenere.
La ricerca ad albero di Monte Carlo, o Monte Carlo tree search (MCTS) usa il metodo di Monte Carlo per stabilire la stima del valore di ogni nodo, indirizzando il giocatore verso alcune direzioni desiderabili all'interno dell'albero di gioco.
L' MCTS si può riassumere nella ripetizione iterativa (idealmente infinita) di 4 fasi: selezione, espansione, simulazione e backup.

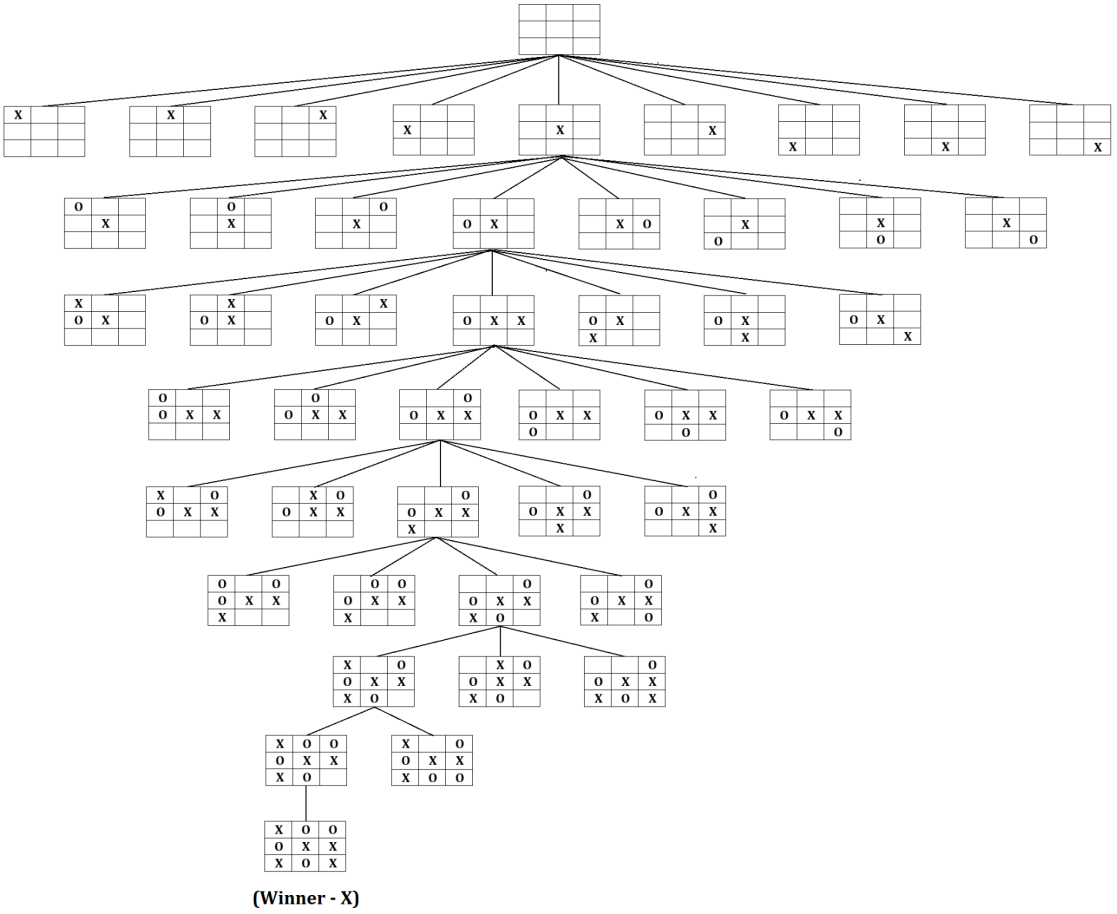

Figura 3: Un possibile albero di gioco di una partita di tris

- Selezione

In questo step si usa la tree *policy* (o regola adottata) per costruire il percorso dalla radice fino alla foglia con lo stato più promettente rispetto a ciò che si vuole ottenere. In questo caso una foglia indica un nodo i cui figli sono altri nodi non ancora esplorati.
Una caratteristica importante di questa policy è il bilanciamento tra esplorazione e sfruttamento delle conoscenze già possedute (exploration and exploitation).

- Espansione

Nello step dell'espansione si sceglie randomicamente un nodo figlio non ancora esplorato di una foglia.

- Simulazione (o Roll-out)

In questo step vengono effettuate diverse simulazioni in cui si va ad analizzare la funzione valore (somma delle ricompense di ogni azione) per ognuna. Solitamente la policy adottata per il Roll-out è piuttosto semplice, in modo da essere veloce da eseguire; ad esempio una vittoria può registrare una ricompensa di +1, un pareggio di 0, una sconfitta di -1.

- Backup

Dopo aver effettuato le diverse simulazioni si riportano le ricompense accumulate di ogni percorso in modo da poter compiere la prossima scelta.

4. AlphaZero

AlphaZero è un algoritmo di intelligenza artificiale sviluppato da Google DeepMind. È una generalizzazione di AlphaGo Zero, predecessore sviluppato specificatamente per il gioco del go. Questo programma ha raggiunto livelli sovrumani di gioco grazie alla sua strategia di ricerca ad albero di Monte Carlo (MCTS) guidata da una rete neurale addestrata tramite Reinforcement Learning. Per vincere contro altri fortissimi motori, come il programma campione del mondo di scacchi Stockfish, a questo algoritmo sono semplicemente bastate le regole del gioco e un po' di tempo per giocare contro se stesso.
A differenza di AlphaZero, Stockfish utilizza metodi di ricerca tradizionali, ossia quello che, allo stato dell'arte, viene chiamato *Alpha-beta Pruning*. Questo è un algoritmo che tenta di diminuire il numero di nodi valutati dall'algoritmo Minimax nel suo albero di ricerca; una caratteristica di questo programma è che smette di valutare una specifica mossa se riesce a dimostrare che la mossa è peggiore rispetto ad una valutata precedentemente.
Quando applicato ad un albero di ricerca Minimax standard, questo algoritmo riporta la stessa mossa che Minimax ritornerebbe, ma scarta i rami dell'albero che non influenzano quella decisione. Questa differenza di strategie dei due algoritmi (Reinforcement Learning e Alpha-beta Pruning) la si nota soprattutto guardando il numero di posizioni studiate: AlphaZero analizza infatti 60'000 posizioni al secondo, mentre Stockfish 60 milioni. La minore quantità di analisi di AlphaZero viene compensata dal fatto che le sue mosse studiate sono selezionate tra quelle che portano agli stati della partita più favorevoli., un approccio più umano di ricerca, come originariamente proposto da Shannon con i suoi metodi di Tipo B. La figura 4 mostra le performance di AlphaZero mentre apprende tramite Reinforcement Learning giocando contro sé stesso. Con il passare del tempo (steps) il suo livello Elo[5] aumenta molto velocemente e dopo appena 4 ore (300'000 steps) è riuscito a superare il livello di Stockfish.

[5] Il sistema di valutazione Elo è un metodo per calcolare i livelli di abilità relativi dei giocatori in giochi a somma zero come gli scacchi, ossia una situazione in cui il guadagno o la perdita di un partecipante è perfettamente bilanciato da una perdita o un guadagno di un altro partecipante in una somma uguale e opposta.

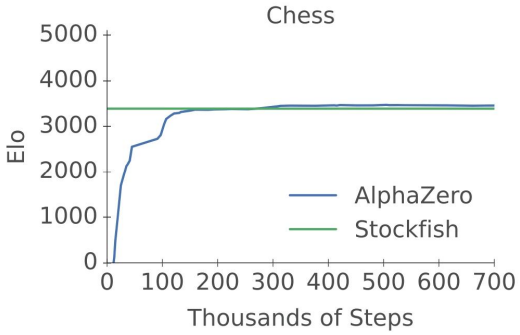

Figura 4: performance di AlphaZero paragonata a Stockfish 2016.

Nel loro articolo “A general reinforcement learning algorithm that masters chess, shogi and Go through self-play”, David Silver, Thomas Hubert e Julian Schrittwieser dell’ University College di Londra hanno analizzato insieme al team di DeepMind la rapida ascesa di AlphaZero mentre parte da una tabula rasa fino a sfidare il programma campione mondiale Stockfish. Vennero fatte 1000 partite, ognuna con tempo limite 3 ore e 15 secondi aggiuntivi per ogni mossa. AlphaZero sconfisse Stockfish, vincendo 155 partite e perdendone 6 su 1000. Per verificare la robustezza di AlphaZero vennero giocate altre partite partendo dalle più comuni posizioni iniziale giocate dalle persone. AlphaZero sconfisse Stockfish in ogni apertura, dimostrando che AlphaZero riesce a destreggiarsi tra le diverse situazioni come se fosse sempre più a suo agio man mano che gioca. Vennero anche giocate partite partendo dalle posizioni iniziali giocate nel campionato mondiale TCEC[6] del 2016 (che vide Stockfish vincitore), vinte da AlphaZero. La tabella T1 mostra 20 partite giocate da AlphaZero contro Stockfish. In diverse di queste AlphaZero sacrifica i propri pezzi per un vantaggio strategico nel lungo termine, suggerendo che ha una valutazione strategico-posizionale più fluida rispetto ai programmi di scacchi precedenti.

[6] Top Chess Engine Championship

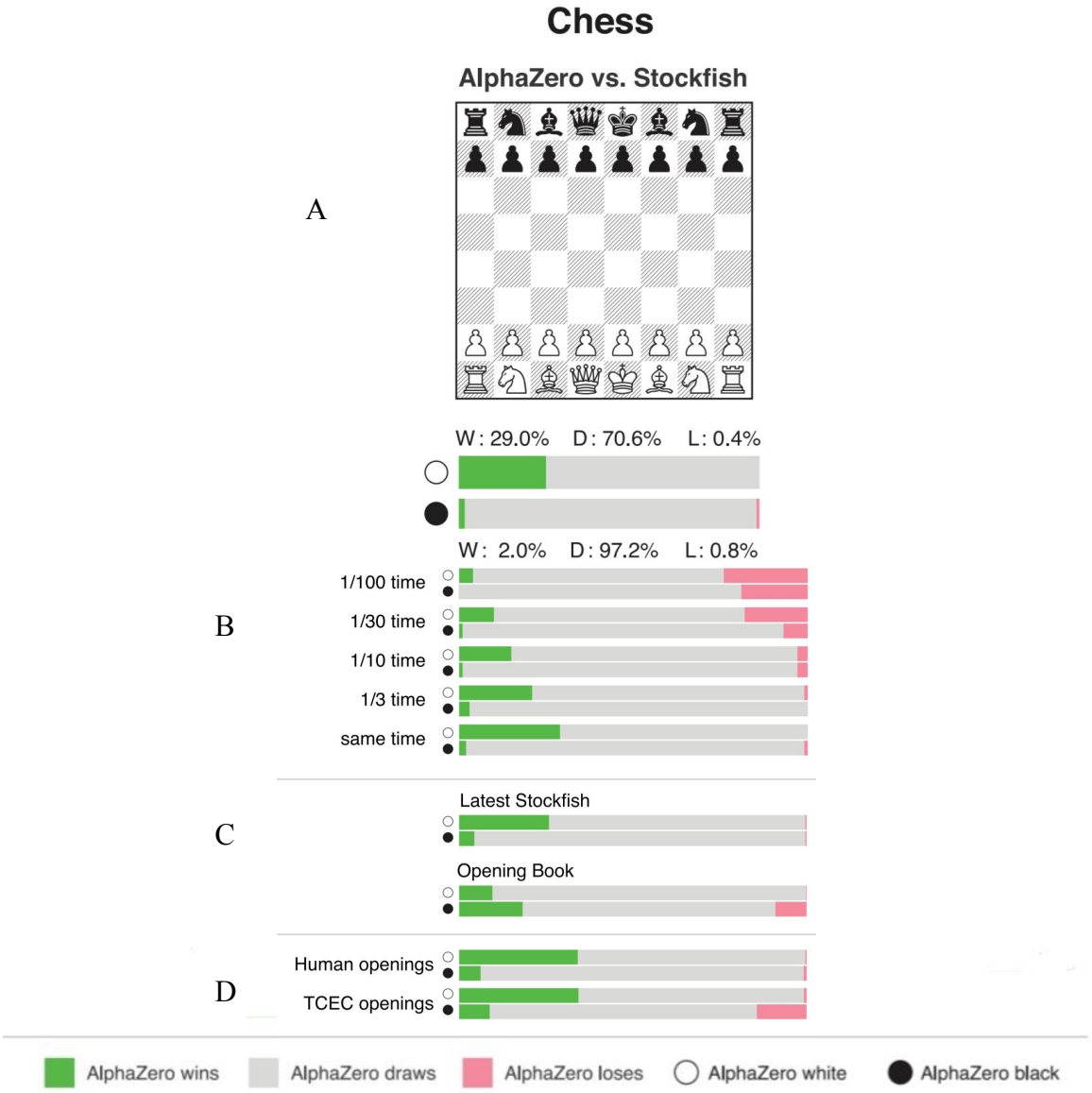

Figura 5: Paragone delle prestazioni con Stockfish.

(A) Valutazione del torneo contro Stockfish. Nella prima barra Alphazero gioca con il bianco, nella seconda con il nero. Ogni barra mostra i risultati dalla prospettiva di AlphaZero: vittoria ('W' in verde), pareggio ('D' in grigio), sconfitta ('L' in rosso).

(B) Scalabilità di AlphaZero a cui viene dato un diverso tempo per pensare alla mossa successiva paragonato a Stockfish. Stockfish ha sempre potuto usare l'intero tempo (3 ore per partita più 15 secondi aggiuntivi dopo ogni mossa), mentre il tempo di AlphaZero è stato diminuito come indicato in figura. (C) Prestazioni di AlphaZero contro una versione aggiornata di Stockfish (Stockfish 9) e contro una versione di Stockfish a cui è stato dato un grande insieme di mosse d'apertura. (D) Risultati medi delle partite di scacchi partendo da diverse posizioni iniziali, sia posizioni comuni giocate dalle persone, che posizioni proposte nel campionato mondiale TCEC del 2016.

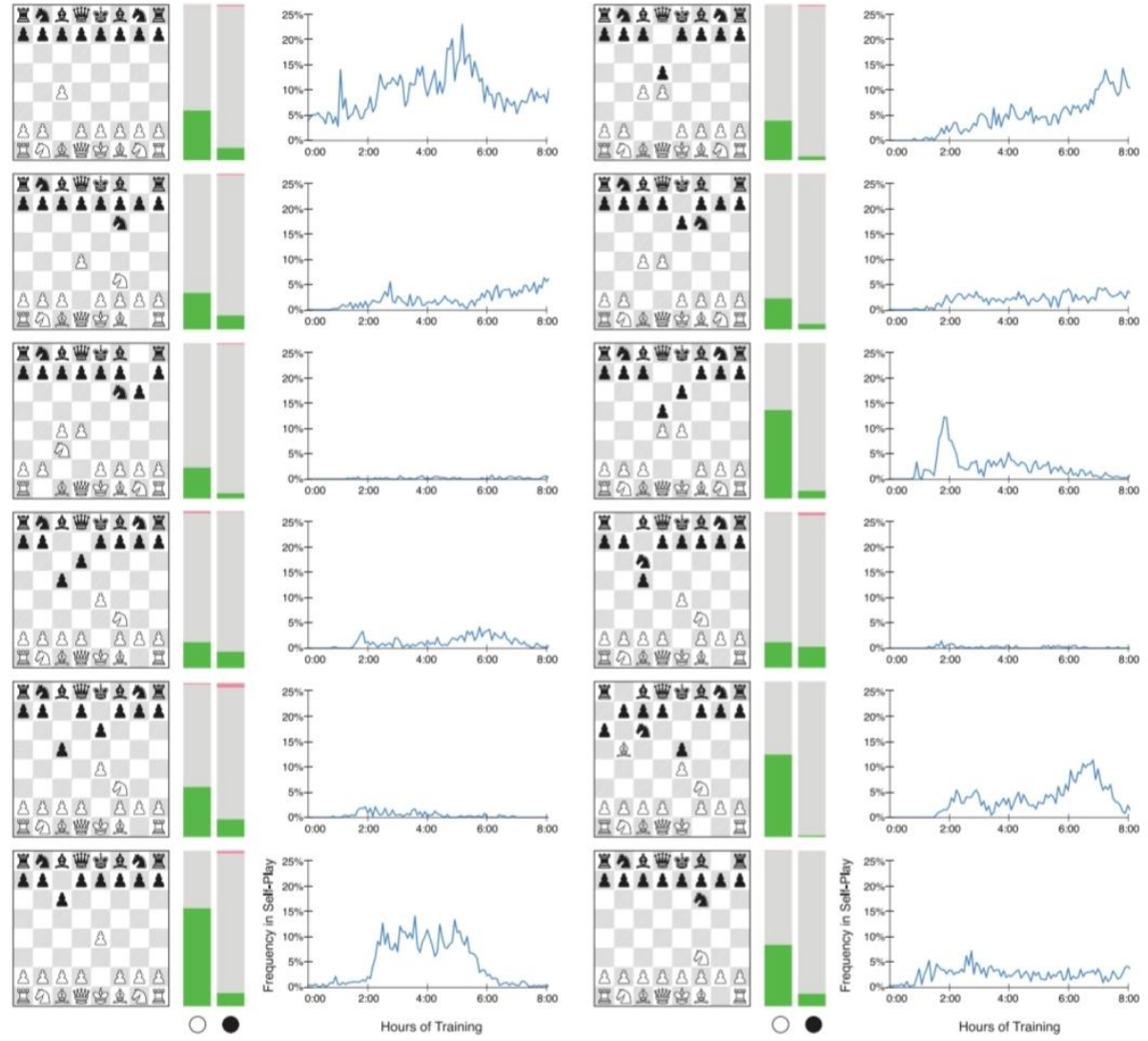

Figura 6: Partite iniziate dalle aperture più popolari. Nella barra a sinistra AlphaZero gioca con il bianco, partendo dalla posizione in figura, mentre nella barra a destra AlphaZero gioca con il nero. Ogni barra mostra i risultati dalla prospettiva di AlphaZero: vittoria (‘W’ in verde), pareggio (‘D’ in grigio), sconfitta (‘L’ in rosso). A lato di ogni scacchiera è mostrata quante volte l’apertura considerata è stata analizzata in percentuale da AlphaZero contro sé stesso, rispetto al tempo di allenamento in ore.

Figura 7: partite iniziate dalle posizioni proposte nel campionato mondiale TCEC del 2016. Molte delle posizioni iniziali sono sbilanciate secondo entrambi i software, provocando un numero maggiore di sconfitte da parte di entrambi.

Tabulati

Partita 1 - Bianco: AlphaZero Nero: Stockfish
1. Nf3 Nf6 2. c4 e6 3. Nc3 Bb4 4. Qc2 O-O 5. a3 Bxc3 6. Qxc3 a5 7. b4 d6 8. e3 Ne4 9. Qc2 Ng5 10. b5 Nxf3+ 11. gxf3 Qf6 12. d4 Qxf3 13. Rg1 Nd7 14. Be2 Qf6 15. Bb2 Qh4 16. Rg4 Qxh2 17. Rg3 f5 18. O-O-O Rf7 19. Bf3 Qh4 20. Rh1 Qf6 21. Kb1 g6 22. Rgg1 a4 23. Ka1 Rg7 24. e4 f4 25. c5 Qe7 26. Rc1 Nf6 27. e5 dxe5 28. Rhe1 e4 29. Bxe4 Qf8 30. d5 exd5 31. Bd3 Bg4 32. f3 Bd7 33. Qc3 Nh5 34. Re5 c6 35. Rce1 Nf6 36. Qd4 cxb5 37. Bb1 Bc6 38. Re6 Rf7 39. Rg1 Qg7 40. Qxf4 Re8 41. Rd6 Nd7 42. Qc1 Rf6 43. f4 Qe7 44. Rxf6 Nxf6 45. f5 Qe3 46. fxg6 Qxc1 47. gxh7+ Kf7 48. Rxc1 Nxh7 49. Bxh7 Re3 50. Rd1 Ke8 51. Ka2 Bd7 52. Bd4 Rh3 53. Bc2 Be6 54. Re1 Kd7 55. Kb2 Rf3 56. Re5 Rg3 57. Re3 Rg2 58. Kc3 Rg4 59. Rf3 Ke8 60. Rf2 Rg3+ 61. Kb4 Rg4 62. Rd2 Bd7 63. Ka5 Rf4 64. Be5 Rf3 65. Rd3 Rf2 66. Bd1 Bc6 67. Kb6 1-0

Partita 2 - Bianco: AlphaZero Nero: Stockfish
1. d4 Nf6 2. c4 e6 3. Nf3 b6 4. g3 Bb7 5. Bg2 Be7 6. Nc3 O-O 7. O-O Ne4 8. Bd2 d5 9. cxd5 exd5 10. Qb3 c5 11. Bf4 Na6 12. Rfd1 c4 13. Qc2 Nb4 14. Qc1 Qd7 15. h4 Rac8 16. a3 Nxc3 17. bxc3 Nc6 18. Qb1 Rce8 19. Re1 Na5 20. Ng5 f5 21. Nf3 Bf6 22. Ra2 h6 23. a4 Qe6 24. Kh2 Bc8 25. Rh1 Nc6 26. h5 Kh8 27. Ng1 Qf7 28. Bf3 Rd8 29. Nh3 Kg8 30. Bc1 Rfe8 31. Qb5 Bb7 32. Rd1 Na5 33. Qb1 Bc8 34. Nf4 Bg5 35. Ng6 Bxc1 36. Qxc1 Be6 37. Ne5 Qc7 38. Rb2 Nb7 39. Rb5 Na5 40. Qf4 Nb3 41. Ng6 Qc6 42. Qe5 Qd7 43. e3 Na5 44. Bg2 Nb7 45. Ra1 Kh7 46. Nf4 Bg8 47. Rxd5 Bxd5 48. Qxd5 Nd6 49. Bh3 Re7 50. Ng6 Rf7 51. Ne5 Qb7 52. Bg2 Qxd5 53. Bxd5 Rc7 54. Kg2 Ne4 55. Bxe4 fxe4 56. f3 exf3+ 57. Kxf3 Re7 58. Ng6 Rb7 59. e4 b5 60. axb5 Rxb5 61. Nf4 Rb3 62. Ne2 Ra8 63. e5 a5 64. d5 a4 65. d6 a3 66. d7 Kg8 67. Rd1 Rbb8 68. e6 Kf8 69. Nd4 1-0

Partita 3 - Bianco: AlphaZero Nero: Stockfish
1. d4 Nf6 2. c4 e6 3. Nf3 b6 4. g3 Bb7 5. Bg2 Bb4+ 6. Bd2 Be7 7. Nc3 O-O 8. Qc2 Na6 9. a3 c5 10. d5 exd5 11. Ng5 Nc7 12. h4 h6 13. Nxd5 Ncxd5 14. cxd5 d6 15. a4 Qd7 16. Bc3 Rfe8 17. O-O-O Bd8 18. e4 Ng4 19. Bh3 hxg5 20. f3 f5 21. fxg4 fxg4 22. Bf1 gxh4 23. Bb5 Qf7 24. gxh4 Bf6 25. Rhf1 Rf8 26. Bxf6 gxf6 27. Rf4 Qg7 28. Be2 Qh6 29. Rdf1 g3 30. Qd3 Kh8 31. Qxg3 Rae8 32. Bd3 Bc8 33. Kb1 Rf7 34. Qf2 Bd7 35. h5 Ref8 36. Bc2 Be8 37. Rf3 Re7 38. Rxf6 Qxf6 39. Qxf6+ Rxf6 40. Rxf6 Kg7 41. Rxd6 Bxh5 42. Kc1 Re5 43. a5 bxa5 44. Kd2 Be8 45. Ra6 Rh5 46. Bd3 a4 47. d6 Bf7 48. d7 Rh8 49. e5 1-0

Partita 4 - Bianco: AlphaZero Nero: Stockfish
1. Nf3 e6 2. c4 Nf6 3. Nc3 Bb4 4. Qc2 O-O 5. a3 Bxc3 6. Qxc3 d6 7. b4 e5 8. Bb2 Nbd7 9. e3 Re8 10. d3 Nf8 11. Be2 a5 12. O-O Bg4 13. h3 Bh5 14. Qc2 h6 15. Bc3 b6 16. b5 N6d7 17. Rad1 Nc5 18. Ba1 Bg6 19. Qb2 Na4 20. Qa2 Nc5 21. d4 exd4 22. Nxd4 Be4 23. Bf3 Bxf3 24. gxf3 Nfe6 25. Kh2 Nxd4 26. Rxd4 Kh7 27. Qc2+ g6 28. Rf4 Qe7 29. Rg1 Rg8 30. h4 h5 31. Rg5 Kh6 32. e4 Ne6 33. Rf6 Nxg5 34. hxg5+ Kh7 35. f4 Rae8 36. Qd3 Rg7 37. f3 Kg8 38. Qd4 Kf8 39. Bc3 Rg8 40. a4 Rd8 41. Kh3 Rd7 42. f5 gxf5 43. Rxf5 Qe6 44. Kh4 Re7 45. Qd5 Rg6 46. Kxh5 Re8 47. Bf6 Qd7 48. Kg4 Rc8 49. Qc6 Qe8 50. Qxe8+ Kxe8 51. Rd5 Rxf6 52. gxf6 Kd7 53. Kf5 c6 54. bxc6+ Kxc6 55. f4 Rh8 56. e5 1-0

Partita 5 - Bianco: AlphaZero Nero: Stockfish
1. d4 Nf6 2. Nf3 e6 3. c4 b6 4. g3 Bb7 5. Bg2 Bb4+ 6. Bd2 Be7 7. Nc3 c6 8. Bf4 O-O 9. e4 d5 10. e5 Ne4 11. cxd5 cxd5 12. O-O Nxc3 13. bxc3 Ba6 14. Re1 Nc6 15. h4 Rc8 16. Re3 Rc7 17. Ng5 h6 18. Nh3 Kh7 19. Rf3 Na5 20. Qc2+ Kg8 21. Re1 Kh8 22. Qd1 Nc6 23. Be3 Bc4 24. Qd2 Kh7 25. Nf4 Qe8 26. g4 Rh8 27. Nh5 Kg8 28. Rh3 Rb7 29. Bf4 Bf8 30. Qd1 Ne7 31. Bc1 b5 32. f4 Rb6 33. Ba3 Ng6 34. Bxf8 Nxf8 35. Qd2 Qc8 36. Rf3 Qd8 37. Qf2 b4 38. cxb4 Rxb4 39. f5 Qc7 40. Rd1 Qb7 41. Rd2 Qc7 42. Qg3 Bb5 43. Kh2 Bd7 44. Qf2 Rb6 45. Rc2 Rc6 46. Rb2 Rb6 47. Bf1 Qb8 48. Rxb6 axb6 49. f6 g6 50. Ng3 Be8 51. Qb2 Qd8 52. h5 Nd7 53. Kg2 g5 54. Rc3 Nxf6 55. exf6 Qxf6 56. Rf3 Qd8 57. Qb4 Kg7 58. Be2 Bc6 59. Rb3 Bd7 60. Qd6 Ba4 61. Qxd8 Rxd8 62. Rxb6 Kf8 63. Kf2 Rc8 64. Ra6 Bd7 65. Ra7 Ke8 66. Bd3 Rc3 67. Ke2 Kd8 68. Kd2 Rc7 69. Rxc7 Kxc7 70. Kc3 Ba4 71. Kb4 Bd1 72. Be2 Bc2 73. Nf1 1-0

Partita 6 - Bianco: AlphaZero Nero: Stockfish
1. Nf3 Nf6 2. c4 c5 3. Nc3 e6 4. g3 Qb6 5. d3 d5 6. Bg2 Be7 7. cxd5 exd5 8. e4 d4 9. Nd5 Nxd5 10. exd5 O-O 11. O-O Bg4 12. h3 Bxf3 13. Qxf3 Na6 14. h4 Bd6 15. h5 Qd8 16. h6 g6 17. Re1 Nc7 18. Bd2 a5 19. a4 b6 20. Re2 Re8 21. Rxe8+ Nxe8 22. Re1 Rb8 23. b3 Nc7 24. Re2 Ra8 25. Kf1 Rb8 26. Bh3 Ra8 27. Re4 Rb8 28. Re1 Ra8 29. Bc1 Rb8 30. Re2 Bf8 31. Re5 Bd6 32. Re1 Bf8 33. Bg2 Bd6 34. Bd2 Rc8 35. Bf4 Qd7 36. g4 Re8 37. Re4 Rd8 38. Bg5 Re8 39. Qf6 Bf8 40. Qc6 Qxc6 41. dxc6 Bd6 42. f4 Kf8 43. f5 gxf5 44. gxf5 Rxe4 45. Bxe4 Be5 46. f6 Kg8 47. Bf5 Ne8 48. Kf2 Bc7 49. Kf3 Bb8 50. Bf4 Bxf4 51. Kxf4 1-0

Partita 7 - Bianco: AlphaZero Nero: Stockfish
1. Nf3 Nf6 2. c4 e6 3. Nc3 d5 4. d4 c6 5. Bg5 Be7 6. e3 h6 7. Bf4 O-O 8. Qc2 Nbd7 9. g4 dxc4 10. Rg1 Nd5 11. g5 Nxf4 12. gxh6 Nh5 13. hxg7 Nxg7 14. O-O-O Qa5 15. Bxc4 Qf5 16. Qe2 Qh7 17. Rg3 Kh8 18. Rdg1 Nf5 19. Qf1 Nxg3 20. Rxg3 Rg8 21. Rh3 Rg7 22. Rxh7+ Rxh7 23. Bd3 Rg7

24. Qd1 Nf6 25. Ne5 Bd7 26. Qf3 Rf8 27. Ne2 Kg8 28. a3 Be8 29. Kb1 a5 30. e4 b5 31. Bc2 b4 32. a4 Kh8 33. Ka2 Kg8 34. Ng3 Rg5 35. Qd1 Kh8 36. Bb3 Rg7 37. Qc2 Ng4 38. Nxc6 Bxc6 39. Qxc6 Rh7 40. Qc7 Bd8 41. Qf4 Rg8 42. Bd1 Nf6 43. h4 Nd7 44. h5 Nf6 45. d5 Re8 46. d6 Rg7 47. h6 Rh7 48. e5 Nd5 49. Qd2 Rg8 50. Bb3 Bg5 51. Qd1 Nb6 52. Ne4 1-0

Partita 8 - Bianco: Stockfish Nero: AlphaZero
1. e4 e5 2. Nf3 Nc6 3. Bb5 Nf6 4. O-O Nxe4 5. d4 Nd6 6. Bxc6 dxc6 7. dxe5 Nf5 8. Qxd8+ Kxd8 9. Rd1+ Ke8 10. Nc3 Be7 11. b3 Nh4 12. Nxh4 Bxh4 13. Be3 Be7 14. Ne2 h5 15. c3 h4 16. Rd2 Rh5 17. h3 a5 18. Re1 Be6 19. f4 a4 20. Nd4 Bd7 21. b4 c5 22. bxc5 Bxc5 23. Nc2 Be7 24. Rb1 b6 25. Nb4 Be6 26. Nc6 a3 27. Kh2 f6 28. Re1 f5 29. Nd4 Bd7 30. Bf2 Rd8 31. Ree2 c5 32. Nc2 g5 33. Nxa3 g4 34. Kg1 g3 35. Be3 Ra8 36. Nc4 Rh6 37. Rb2 Ra6 38. Bc1 b5 39. Ne3 Ra4 40. c4 bxc4 41. Nd5 c3 42. Nxc3 Rc4 43. Bd2 Rc6 44. Kf1 Be6 45. Rb1 Rb4 46. Ree1 Bc4+ 47. Kg1 Rc8 48. Rbc1 Bd3 49. Nd5 Rb2 50. Bc3 Rxa2 51. Ra1 Rxa1 52. Bxa1 c4 53. Nf6+ Kd8 54. Bc3 Rb8 55. Bd4 Bb4 56. Rd1 Rb5 57. Kh1 Bc5 0-1

Partita 9 - Bianco: Stockfish Nero: AlphaZero
1. e4 e5 2. Nf3 Nc6 3. Bc4 Bc5 4. d3 a6 5. Ng5 Nh6 6. O-O d6 7. a4 Bg4 8. Nf3 O-O 9. h3 Bh5 10. c3 Kh8 11. Bxh6 gxh6 12. Nbd2 Ba7 13. Bd5 Ne7 14. Bxb7 Rb8 15. Bxa6 f5 16. Kh1 Ng6 17. exf5 Nf4 18. d4 Rxf5 19. Qc2 Rf8 20. Rae1 Qf6 21. Re3 Qg7 22. Rg1 Nd5 23. Bb5 Bg6 24. Qc1 Nxe3 25. fxe3 Bf7 26. Rf1 Bd5 27. Bc4 Ba8 28. a5 e4 29. Nh2 Qg5 30. b4 Qxe3 31. Ng4 Qg5 32. Qe1 h5 33. Ne3 h4 34. Be6 Bc6 35. Bc4 d5 36. b5 Bb7 37. Bb3 Rbc8 38. a6 Ba8 39. Ba4 Bb6 40. Kg1 Qg3 41. Qxg3 hxg3 42. Ra1 Rf2 43. Ndf1 Re2 44. Nf5 Rg8 45. N1xg3 Rd2 46. Rf1 Ba5 47. Rf2 Rxf2 48. Kxf2 Bxc3 49. h4 Rf8 50. Ke3 Be1 51. Ke2 Bxg3 52. Nxg3 Rg8 53. Kf2 Rg4 54. Bd1 e3+ 55. Kf3 Rxd4 56. Be2 Rb4 57. Kxe3 d4+ 58. Kd2 Rb2+ 59. Ke1 Rb3 60. Nh5 d3 61. Bd1 Rxb5 62. Nf4 Ra5 63. g3 Rxa6 64. Nxd3 Bd5 65. Kd2 Bf7 66. g4 Kg7 67. Nf2 Ra8 68. Be2 Ra4 69. Bd1 Rd4+ 70. Ke3 Rb4 71. Nd3 Rb1 72. Kd2 Rb6 73. Ke3 Re6+ 74. Kd2 Rd6 75. Kc3 Bg6 76. Nf2 Rf6 77. Nh3 Bf7 78. Be2 Re6 79. Kd2 Rb6 80. Nf2 Bd5 81. Bd3 Rb4 82. Bf5 h6 83. Ke3 Bf7 84. Nd3 Rb5 85. Bd7 Ra5 86. Bc6 Kf8 87. Be4 Ke7 88. Bf3 Kd6 89. Nf2 Bg6 90. h5 Bb1 91. Nd1 Bh7 92. Nb2 Ke7 93. Nc4 Ra6 94. Ne5 Kf6 95. Nd7+ Kg5 96. Be2 Re6+ 97. Kf2 Re7 0-1

Partita 10 - Bianco: Stockfish Nero: AlphaZero
1. e4 e5 2. Nf3 Nc6 3. Bb5 Nf6 4. O-O Nxe4 5. d4 Nd6 6. Bxc6 dxc6 7. dxe5 Nf5 8. Qxd8+ Kxd8 9. Nc3 Be7 10. Rd1+ Ke8 11. Ne4 Be6 12. b3 b6 13. h3 Rd8 14. Bb2 h5 15. Rxd8+ Bxd8 16. Rd1 h4 17. Nh2 c5 18. c4 a5 19. Nc3 Nd4 20. Ng4 Rh5 21. Kf1 Bd7 22. f3 Ne6 23. Nd5 Bg5 24. Nf2 Bd8 25. Nd3 Rf5 26. Ne3 Rg5 27. Bc3 Rh5 28. Kf2 Bc8 29. Nd5 Bg5 30. f4 Bd8 31. Ne3 Bd7 32. Nd5 Bc8 33. Ne3 g6 34. Bd2 Bb7 35. Nd5 Kd7 36. Nc1 Kc8 37. Ne2 Bc6 38. Be3 Ng7 39. Nec3 Rh8 40. Ne4 a4 41. Kf3 Nf5 42. Rd2 Re8 43. Bf2 Rg8 44. b4 cxb4 45. Nxb4 Bb7 46. Nd5 Re8 47. Kg4 Nh6+ 48. Kf3 Nf5 49. Kg4 Nh6+ 50. Kf3 Re6 51. Rd3 Nf5 52. Rd2 a3 53. Kg4 Nh6+ 54. Kf3 Nf5 55. Kg4 Nh6+ 56. Kf3 Rc6 57. Ke3 Nf5+ 58. Kd3 Re6 59. Re2 Ne7 60. Ndf6 Nf5 61. Nd5 Re8 62. Kc3 b5 63. Nc5 Bc6 64. Ne4 Bb7 65. Nc5 Bc6 66. Ne4 bxc4 67. Kxc4 Bb7 68. Re1 Ba6+ 69. Kb3 Bb7 70. Kc4 Ba8 71. Bc5 Re6 72. Ng5 Bxg5 73. fxg5 Re8 74. Nf4 Ng7 75. Bxa3 Rd8 76. Re2 Rd1 77. Rf2 Ne6 78. Nxe6 Bd5+ 79. Kb5 Bxe6 80. Bc5 Rb1+ 81. Kc6 Rd1 82. Kb5 Ra1 83. a3 Re1 84. Bd4 Rb1+ 85. Kc5 Rc1+ 86. Kb5 Bd7+ 87. Kb4 Be6 88. Kb5 Rb1+ 89. Kc5 Rd1 90. a4 Rc1+ 91. Kb5 Kb7 92. Rd2 Bb3 93. Rb2 Bc4+ 94. Kb4 Be6 95. Be3 Re1 96. Bd4 Rc1 97. Kb5 Bd7+ 98. Kb4 Ka6 99. Rb3 Rc2 100. g4 Rd2 101. Kc5 Be6 102. Rf3 Kb7 103. a5 Ra2 104. Bc3 Ra3 105. Kb5 Rb3+ 106. Kc5 Ra3 107. Re3 Ra2 108. Be1 Ra1 109. Bd2 Kc8 110. Rd3 Rd1 111. Kb5 Bd7+ 112. Kb4 Be6 113. Kc5 Ra1 114. Kd4 Ra4+ 115. Kc5 Ra1 116. a6 Rxa6 117. Be1 Kb7 118. Bxh4 Ra5+ 119. Kd4 c5+ 120. Ke4 Kc6 121. Be1 Ra2 122. Rd6+ Kb5 123. Rd3 Rh2 124. Re3 Rg2 125. Kf3 Rc2 126. Rc3 Rh2 127. Kg3 Re2 128. Bf2 Kb4 129. Rc1 c4 130. Re1 Rc2 131. Be3 c3 132. h4 Ra2 133. h5 Kb3 134. h6 Ra8 135. Rc1 c2 136. h7 Rh8 137. Rh1 Kc3 138. Kf4 Kd3 139. Rh2 Kc3 140. Ke4 Kb4 141. Kd3 Bc4+ 142. Kxc2 Be6 143. Bc1 Rc8+ 144. Kd3 Rh8 145. Ke4 Ka4 146. Kf4 Kb3 147. Rh3+ Ka4 148. Bb2 Kb5 149. Ba3 1-0

Partita 11 - Bianco: AlphaZero Nero: Stockfish
1. d4 {da libro} d5 {da libro} 2. c4 {da libro} c6 {da libro} 3. Nf3 {da libro} Nf6 {da libro} 4. Nc3 da libro e6 {da libro} 5. e3 {da libro} Nbd7 {da libro} 6. Qc2 {da libro} Bd6 {da libro} 7. g4 {da libro} Bb4 {da libro} 8. Bd2 {da libro} Qe7 {da libro} 9. Rg1 {da libro} Bxc3 {da libro} 10. Bxc3 {da libro} Ne4 {da libro} 11. O-O-O {da libro} O-O {da libro} 12. Be1 {da libro} b6 13. h4 Bb7 14. Ng5 Nxg5 15. hxg5 Qxg5 16. f4 Qe7 17. Kb1 c5 18. Bd3 g6 19. cxd5 cxd4 20. e4 Rac8 21. Qh2 f6 22. f5 exd5 23. fxg6 hxg6 24. Rh1 Qg7 25. Bc2 Ne5 26. Bb3 g5 27. Bg3 Nxg4 28. Qh3 Ne3 29. Rxd4 Kf7 30. Bf2 Rh8 31. Qd7+ Kg6 32. Qxg7+ Kxg7 33. Rxh8 Rxh8 34. Bxe3 Re8 35. Ra4 a6 36. Bxb6 dxe4 37. Rd4 Bc8 38. Kc1 e3 39. Rd8 Rxd8 40. Bxd8 Kg6 41. Bb6 Bb7 42. Bc2+ Kf7 43. Bxe3 Ke6 44. Kd2 Kd6 45. Bd3 1-0

Partita 12 - Bianco: AlphaZero Nero: Stockfish
1. d4 {da libro} Nf6 {da libro} 2. c4 {da libro} g6 {da libro} 3. Nc3 {da libro} Bg7 {da libro} 4. e4 da libro d6 {da libro} 5. f3 {da libro} O-O {da libro} 6. Be3 {da libro} c5 {da libro} 7. Nge2 {da libro} Nc6 {da libro} 8. d5 {da libro} Ne5 {da libro} 9. Ng3 {da libro} e6 {da libro} 10. Be2 {da libro} exd5 {da libro} 11. cxd5 {da libro} a6 12. Qd2 b5 13. O-O Re8 14. Bh6 Bxh6 15. Qxh6 Qe7 16. Nd1 Qf8 17. Qd2 h5 18. h4 Bd7 19. Nf2 b4 20. Rfe1 Bb5 21. f4 Ned7 22. Bf3 Qh6 23. Rad1 Rab8 24. Nh3 a5 25. Ng5 a4 26. e5 dxe5 27. f5 Bc4 28. d6 Bxa2 29. Bc6 Bb3 30. Ra1 Rf8 31. Bxa4 Bxa4 32. Rxa4 Rbe8 33. Ra7 e4 34. Rc7 Re5 35. N3xe4 Nxe4 36. Rxe4 Rxe4 37. Nxe4 Qxd2 38. Nxd2 Rd8 39. fxg6 fxg6 40. Ne4 Kf8 41. Kf1 b3 42. Ke2 Re8 43. Kd3 Ne5+ 44. Ke3 Nf7 45. d7 Re6 46. Rxc5 Ke7 47. Rd5 Ra6 48. Nc5 Ra1 49. Rd2 Rc1 50. Kd4 Kd6 51. Nxb3 Rc6 52. Na5 Rc1 53. Rf2 Nd8 54. Nc4+ Kxd7 55. Ne5+ Ke6 56. Nxg6 Nc6+ 57. Ke3 Re1+ 58. Kd3 Rd1+ 59. Ke2 Rh1 60. g3 Rg1 61. Nf4+ Ke7 62. Nxh5 Ne5 63. Kd2 Ra1 64. Kc3 Kd6 65. Nf4 Rc1+ 66. Rc2 Rg1 67. Rg2 Rc1+ 68. Kb3 Rc8 69. Rd2+ Ke7 70. Rd5 Kf6 71. Ka2 Rg8 72. Ne2 Ke6 73. Rd4 Ra8+ 74.

Kb1 Nc6 75. Rc4 Kd6 76. Re4 Rc8 77. h5 Rg8 78. Rh4 Ke7 79. h6 Kf7 80. h7 Rh8 81. Kc2 Ne7 82. Kd3 Ng6 83. Rh6 Kg7 84. Rh5 Nf8 85. b4 Nxh7 86. Nf4 Re8 87. Rf5 Nf8 88. Kd4 Re1 89. b5 Nd7 90. Kd5 1-0

Partita 13 - Bianco: AlphaZero Nero: Stockfish
1. e4 {da libro} c6 {da libro} 2. d4 {da libro} d5 {da libro} 3. e5 {da libro} Bf5 {da libro} 4. Be3 { da libro} e6 {da libro} 5. Nd2 {da libro} Nd7 {da libro} 6. Ngf3 {da libro} Ne7 {da libro} 7. Be2 {da libro } Qc7 {da libro} 8. O-O {da libro} f6 {da libro} 9. c3 fxe5 10. Nxe5 Nxe5 11. dxe5 Qxe5 12. Re1 Qc7 13. Bh5+ Ng6 14. g4 Bd3 15. Nb3 Bc4 16. Nd4 e5 17. b3 Ba6 18. Bf4 O-O-O 19. Bg3 Re8 20. a4 Kb8 21. b4 Bd6 22. Nf5 Bc4 23. Qd2 Ka8 24. Qg5 Re6 25. Nxd6 Rxd6 26. Bxg6 Rxg6 27. Qxe5 Qxe5 28. Rxe5 Rxg4 29. f3 Rg6 30. a5 Rf6 31. Rae1 Rff8 32. Kf2 g6 33. Re7 h5 34. Be5 Rhg8 35. Bg7 Rc8 36. Bd4 Bb5 37. Kg3 Bd3 38. Kh4 Bf5 39. Kg5 Rcd8 40. Bg7 Rc8 41. R1e2 Rcd8 42. h4 Rc8 43. a6 bxa6 44. Bd4 Kb8 45. Bxa7+ Ka8 46. Bc5 Kb8 47. Ra2 Bd3 48. Rd2 Bb1 49. Rd1 Bc2 50. Ba7+ Ka8 51. Rde1 Bd3 52. Bb6 Kb8 53. Bd4 Bf5 54. f4 Rcd8 55. Ra1 Bd3 56. Re3 Bb5 57. Rae1 Kc7 58. Kh6 Kc8 59. Re6 Ba4 60. Bc5 Kb7 61. Rxg6 Rxg6+ 62. Kxg6 d4 63. Bxd4 Bc2+ 64. Kxh5 c5 65. Bxc5 Kc8 66. Re5 Rh8+ 67. Kg4 Bd1+ 68. Kg3 Rh7 69. Bd4 Kd7 70. f5 Rg7+ 71. Kf2 1-0

Partita 14 - Bianco: AlphaZero Nero: Stockfish
1. d4 {da libro} Nf6 {da libro} 2. c4 {da libro} e6 {da libro} 3. Nc3 {da libro} Bb4 {da libro} 4. e3 da libro c5 {da libro} 5. Bd3 {da libro} Nc6 {da libro} 6. Nf3 {da libro} Bxc3+ {da libro} 7. bxc3 da libro d6 {da libro} 8. e4 h6 9. e5 dxe5 10. Nxe5 cxd4 11. Nxc6 bxc6 12. O-O dxc3 13. Ba3 Qc7 14. Qf3 Rb8 15. Bc5 e5 16. Rfe1 Be6 17. Qg3 Nh5 18. Qe3 Nf4 19. Bc2 f6 20. h4 Qa5 21. Rab1 Kf7 22. g3 Qxa2 23. Qe4 Qxc4 24. Qxc6 Nd3 25. Rxb8 Rxb8 26. Qc7+ Kg6 27. Bxd3+ Qxd3 28. Qxb8 Bd5 29. Kh2 c2 30. Qb2 a5 31. Qc1 a4 32. Qe3 Qxe3 33. fxe3 Kf5 34. Kg1 Ke4 35. Kf2 Be6 36. Ke2 Bg4+ 37. Kd2 Bd1 38. Rf1 f5 39. Rf2 g6 40. Ba3 Bg4 41. Kxc2 Kxe3 42. Bc5+ Ke4 43. Kd2 g5 44. Bf8 f4 45. gxf4 gxf4 46. Ke1 Be6 47. Rd2 Kf5 48. Rd8 1-0

Partita 15 - Bianco: Stockfish Nero: AlphaZero
1. d4 {da libro} f5 {da libro} 2. Nf3 {da libro} Nf6 {da libro} 3. g3 {da libro} g6 {da libro} 4. Bg2 da libro Bg7 {da libro} 5. O-O {da libro} O-O {da libro} 6. c4 {da libro} d6 {da libro} 7. Nc3 {da libro} c6 {da libro} 8. Rb1 {da libro} a5 9. Qb3 Na6 10. Rd1 h6 11. Be3 Rb8 12. Rbc1 Bd7 13. c5+ Kh7 14. Na4 Nc7 15. Bd2 Be6 16. Qc2 Ncd5 17. b3 Ra8 18. Be1 Qe8 19. e3 g5 20. Nd2 Qh5 21. Bf3 g4 22. Be2 Kh8 23. Nc4 Ne4 24. h4 Ng5 25. hxg5 hxg5 26. f3 gxf3 27. Bf1 f4 28. Rd2 fxe3 29. Nxe3 Nxe3 30. Rh2 Bh3 31. Rxh3 Qxh3 32. Bxh3 Nxc2 33. Rxc2 Bxd4+ 34. Bf2 Bxf2+ 35. Kxf2 Kg7 36. Nb6 Rad8 37. Rc3 Rh8 38. Be6 Rh6 39. Re3 Rf8 40. Nd7 Rh2+ 41. Kf1 Re2 42. Rxe2 fxe2+ 43. Kxe2 Rh8 44. Kd3 Rh6 45. Bg4 d5 46. Ne5 e6 47. Nd7 Kf7 48. Ke3 Rh1 49. Bf3 Re1+ 50. Kd2 Ra1 51. Bd1 Ke7 52. Ne5 Kf6 53. Ng4+ Kf5 54. Nh6+ Ke4 55. Nf7 g4 56. Nd6+ Kd4 57. a4 Ra2+ 58. Ke1 Kxc5 59. Nxb7+ Kb6 60. Nd8 Rg2 61. Nxe6 Rxg3 62. Kf2 Rc3 63. Bxg4 Rxb3 64. Bd1 Rb4 65. Kf3 Re4 0-1

Partita 16 - Bianco: Stockfish Nero: AlphaZero
1. d4 {da libro} Nf6 {da libro} 2. c4 {da libro} g6 {da libro} 3. Nc3 {da libro} Bg7 {da libro} 4. e4 da libro d6 {da libro} 5. f3 {da libro} O-O {da libro} 6. Be3 {da libro} c5 {da libro} 7. Nge2 {da libro} Nc6 {da libro} 8. d5 {da libro} Ne5 {da libro} 9. Ng3 {da libro} e6 {da libro} 10. Be2 {da libro} exd5 {da libro} 11. cxd5 {da libro} h5 12. O-O h4 13. Nh1 h3 14. g3 Bd7 15. Rc1 b5 16. Nf2 Re8 17. Kh1 b4 18. Nb1 Neg4 19. fxg4 Nxe4 20. Nxe4 Rxe4 21. Bf4 Qe7 22. Bf3 Bxb2 23. Bxd6 Qxd6 24. Bxe4 Bxc1 25. Qxc1 Bb5 26. Re1 Re8 27. Nd2 c4 28. Nf3 c3 29. Qf4 Qc5 30. Ne5 Re7 31. Qf6 c2 32. Nxg6 fxg6 33. Qxg6+ Kf8 34. Qf6+ Kg8 35. d6 c1=Q 36. Bh7+ Rxh7 37. Qd8+ Kf7 38. Qe7+ Kg6 39. Qe6+ Kg5 40. Qg8+ Kh6 41. Qe6+ Kg7 42. Qe7+ Kg6 43. Qe6+ Kg5 44. Qg8+ Kh6 45. Qe6+ Kg7 46. Qe7+ Kg8 47. Qd8+ Be8 48. Qxe8+ Kg7 49. Qe7+ Kg6 50. Qe6+ Kg5 51. Qg8+ Kh6 52. Qe6+ Kg5 53. Qg8+ Kh6 54. Qe6+ Kg7 55. Qd7+ Kg6 56. Qe6+ Kg7 57. Qe7+ Kg8 58. Qd8+ Kg7 59. Qd7+ Kg6 60. Qe6+ 1/2-1/2

Partita 17 - Bianco: Stockfish Nero: AlphaZero
1. e4 {da libro} c5 {da libro} 2. Nf3 {da libro} d6 {da libro} 3. d4 {da libro} cxd4 {da libro} 4. Nxd4 {da libro} Nf6 {da libro} 5. Nc3 {da libro} a6 {da libro} 6. Bg5 {da libro} Nbd7 {da libro} 7. f4 da libro e6 {da libro} 8. Qe2 {da libro} Be7 {da libro} 9. O-O-O {da libro} Qc7 {da libro} 10. g4 {da libro} b5 {da libro} 11. a3 {da libro} Rb8 12. Bg2 b4 13. axb4 h6 14. Bh4 Rxb4 15. Be1 Qb6 16. Bf2 Qb7 17. Rhg1 Qc7 18. Bf3 Nb6 19. Na2 Ra4 20. Kb1 e5 21. fxe5 dxe5 22. Nb3 Nc4 23. h4 h5 24. gxh5 Be6 25. Rxg7 a5 26. Be1 Rxa2 27. Kxa2 a4 28. Nc1 Na3+ 29. Ka1 Nxc2+ 30. Kb1 Nxe1 31. Qxe1 a3 32. b3 a2+ 33. Kxa2 Nd5 34. Rxd5 Bxd5 35. Kb1 Bb7 36. Rg2 Qd6 37. Qc3 Kf8 38. Nd3 Rg8 39. Rxg8+ Kxg8 40. Kc2 Bf8 41. b4 Qa6 42. Kb3 Bc6 43. Nc5 Qf1 44. Kc2 Bb5 45. Kb3 Bh6 46. Ka3 Bc1+ 47. Kb3 Bf4 48. Ka3 Bh6 49. Nb3 Bd7 50. Nc5 Bb5 51. Nb3 Bd7 52. Nc5 Bc1+ 53. Kb3 Bb5 54. Na4 Bh6 55. Qc8+ Bf8 56. Qc3 Bh6 57. Qc8+ Bf8 58. Qc3 Bd7 59. Nc5 Bb5 60. Kb2 Bh6 61. Nb3 Bf4 62. Qc8+ Kg7 63. h6+ Bxh6 64. Qg4+ Kh7 65. Qf5+ Kg8 66. Qg4+ Kh7 67. Qf5+ Kg8 68. Qg4+ Kh7 1/2-1/2

Partita 18 - Bianco: Stockfish Nero: AlphaZero
1. e4 {da libro} e5 {da libro} 2. Nf3 {da libro} Nc6 {da libro} 3. Bb5 {da libro} a6 {da libro} 4. Bxc6 {da libro} dxc6 {da libro} 5. O-O {da libro} f6 {da libro} 6. d4 {da libro} Bg4 {da libro} 7. c3 {da libro} Bd6 {da libro} 8. Be3 {da libro} Qe7 {da libro} 9. Nbd2 {da libro} O-O-O {da libro} 10. dxe5 fxe5 11. h3 Bd7 12. b4 Nf6 13. Qb3 g5 14. Bxg5 Rdg8 15. Kh1 Rg6 16. Rg1 Be6 17. c4 Rhg8 18. Bh4 Qf8 19. Qe3 c5 20. Rac1 b6 21. b5 Nh5 22. bxa6 Qg7 23. g3 Bxh3 24. a4 Kb8 25. a5 Bc8 26. Rg2 Ka7 27. Rh2 Bg4 28. Qd3 Qf7 29. Rc3 bxa5 30. Rb3 a4 31. Rb7+ Kxa6 32. Rb1 Qe8 33.

Bg5 Nf6 34. Bxf6 Rxf6 35. Qc3 Bf8 36. Rb5 Rb6 37. Qa3 Bd7 38. Qxa4+ Kb7 39. Rxb6+ cxb6 40. Qc2 h5 41. Kg2 Bg7 42. Rh1 Qf7 43. Nh4 Bc6 44. Ndf3 Rd8 45. Nf5 Bf6 46. Ne3 Qg6 47. Nd5 Rxd5 48. cxd5 Bxd5 49. Nd2 Bc6 50. Qa2 Bg5 51. f3 Bf4 52. Nf1 b5 53. Qb2 c4 54. Kf2 Bg5 55. Qxe5 Bd8 56. Ke2 h4 57. g4 Qg5 58. Qxg5 Bxg5 59. Ne3 Kb6 60. Rd1 Kc5 61. Nd5 b4 62. f4 Bh6 63. g5 Bf8 64. f5 c3 65. f6 Be8 66. Kd3 Bb5+ 67. Ke3 Kc4 68. Nb6+ Kb3 69. g6 c2 70. Re1 Be8 71. g7 Bc5+ 72. Kd2 Bf7 73. e5 Bf2 74. Re2 Bg3 75. Re4 h3 76. e6 h2 77. Re3+ Kb2 78. Nc4+ Kb1 79. Rb3+ Ka1 80. exf7 h1=Q 81. Kxc2 Qe4+ 82. Kd2 Bf4+ 83. Ne3 Qd5+ 84. Ke2 Qh5+ 85. Kd3 Qxf7 86. Rxb4 Bg5 87. Ng4 Ka2 88. Rd4 Qe8 89. Re4 Qd7+ 90. Ke2 Qd5 91. Kf3 Ka1 92. Ne3 Qf7 93. Ng4 Qg6 94. Ra4+ Kb1 95. Ke2 Qe8+ 96. Kf2 Qe6 97. Kf1 Qf5+ 98. Ke1 Bh4+ 99. Ke2 Qc2+ 100. Kf3 Qd1+ 101. Ke3 Qd5 102. Kf4 Kb2 103. Ke3 Kc2 104. Kf4 Bg5+ 105. Kg3 Kb2 106. Kh2 Qd2+ 107. Kg1 Qd1+ 108. Kf2 Qc2+ 109. Ke1 Qg6 110. Ne5 Qe8 111. Re4 Qg8 112. Ng4 Kc3 113. Ke2 Qa2+ 114. Kf3 Qd5 115. Ne3 Qf7 116. Re7 Qxf6+ 117. Kg4 Qg6 118. Nd5+ Kd2 119. Nf4 Bxf4+ 120. Kxf4 Kd1 121. Kf3 Qg5 122. Rd7+ Kc1 123. Ra7 Kb2 124. Rb7+ Kc3 125. Ra7 Kb3 126. Rc7 Qf5+ 127. Kg3 Qe6 128. Ra7 Qg6+ 129. Kf3 Kb2 130. Rb7+ Kc2 131. Ra7 Qf5+ 132. Kg2 Qg4+ 133. Kf2 Kb3 134. Re7 Kb2 135. Re2+ Kc1 136. Re1+ Kd2 137. Re7 Qg5 138. Ra7 Kc2 139. Rc7+ Kd3 140. Kf3 Qf5+ 141. Kg3 Qg6+ 142. Kf3 Qe4+ 143. Kg3 Qe6 144. Ra7 Qg6+ 145. Kf3 Qf5+ 146. Kg3 Qg5+ 147. Kf3 Kc3 148. Rc7+ Kb3 149. Ra7 Qg6 150. Rb7+ Kc3 151. Ra7 Qf5+ 152. Ke3 Qg4 153. Re7 Kc2 154. Rb7 Qg5+ 155. Kf3 Kc3 156. Ra7 Qf5+ 157. Ke3 Qg4 158. Re7 Kc2 159. Rb7 Qe6+ 160. Kf4 Kd2 161. Rd7+ Ke2 162. Re7 Qxe7 163. g8=Q Qe3+ 164. Kf5 Qd3+ 165. Kf4 Qf3+ 166. Ke5 Qc3+ 167. Ke4 Qc2+ 168. Ke5 Qc3+ 169. Ke4 Qc2+ 170. Ke5 Qb2+ 171. Ke4 Qb1+ 172. Ke5 Qa1+ 173. Ke4 Qb1+ 174. Ke5 Qb2+ 175. Ke4 Qb4+ 176. Kf5 Qc5+ 177. Ke4 Qe7+ 178. Kf5 Qc5+ 179. Ke4 Qe3+ 180. Kf5 Qd3+ 181. Kf4 Qd4+ 182. Kf5 Qf2+ 183. Ke4 Qf3+ 184. Ke5 Qe3+ 185. Kf6 Qf4+ 186. Kg7 Qe5+ 187. Kg6 Qg3+ 188. Kh7 Qh4+ 189. Kg6 Qg3+ 190. Kf7 Qc7+ 191. Kf6 Qf4+ 192. Kg7 Qd4+ 193. Kg6 Qe4+ 194. Kf7 Qf5+ 195. Ke8 Qb5+ 196. Ke7 Qc5+ 197. Kf6 Qc3+ 198. Kf5 Qf3+ 199. Kg6 Qg4+ 200. Kh7 Qe4+ 201. Kh6 Qh4+ 202. Kg7 Qd4+ 203. Kg6 Qg4+ 204. Kh7 Qd7+ 205. Kg6 Qc6+ 206. Kf5 Qb5+ 207. Ke4 Qa4+ 208. Kf5 Qc2+ 209. Kf4 Qc1+ 210. Kf5 Qf1+ 211. Ke4 Qf3+ 212. Ke5 1/2-1/2

Partita 19 - Bianco: Stockfish Nero: AlphaZero
1. e4 {da libro} e5 {da libro} 2. Nf3 {da libro} Nc6 {da libro} 3. Bb5 {da libro} f5 {da libro} 4. Nc3 fxe4 5. Nxe4 Nf6 6. Nxf6+ Qxf6 7. Qe2 Be7 8. Bxc6 bxc6 9. Nxe5 Bb7 10. O-O O-O-O 11. d3 Rde8 12. Nc4 h5 13. Qe3 h4 14. Qh3 Kb8 15. Bd2 d5 16. Bc3 d4 17. Bd2 Bc8 18. Qf3 Qxf3 19. gxf3 Be6 20. f4 h3 21. Ne5 Bf6 22. Nxc6+ Kc8 23. Rfe1 Bd5 24. Rxe8+ Rxe8 25. Ne5 Bxe5 26. Re1 Kd7 27. fxe5 Re6 28. c4 dxc3 29. Bxc3 Rg6+ 30. Kf1 Bxa2 31. Re3 Be6 32. Rg3 Rxg3 33. fxg3 Bg4 34. e6+ Kxe6 35. Bxg7 Kf5 36. b4 c6 37. Bd4 a6 38. Kf2 Bd1 39. Bc5 Kg4 40. Ke3 Ba4 41. Bd4 Bc2 42. Ke4 Ba4 43. Bc5 Bb5 44. Be3 Ba4 45. d4 Bd1 46. Bf4 Ba4 47. Ke5 Bb5 48. Kf6 Ba4 49. Ke6 Bb5 50. Kd6 Kh5 51. Kc5 Ba4 52. Kb6 Bb5 53. Be5 Kg5 54. Bh8 Kg6 55. Ka5 Kg5 56. Bg7 Kg6 57. Bf8 Kh5 58. Be7 Bf1 59. Kb6 Bb5 60. Bd8 Kg4 61. Bc7 Kg5 62. Kc5 Ba4 63. Kd6 Bb5 64. Bd8+ Kh5 65. Kc5 Kg4 66. Bc7 Kh5 67. Bb6 Ba4 68. Bd8 Kg4 69. Bf6 Bb5 70. Kb6 Kh5 71. Bg7 Kg5 72. Ka7 Kh5 73. Bf6 Kg4 74. Be5 Kh5 75. Kb8 Kg5 76. Kb7 Kh5 77. Ka7 Kg5 78. Bh8 Kh5 79. Kb8 Kg4 80. Kc7 Kg5 81. Kd6 Kh5 82. Bg7 Ba4 83. Bf6 Bb5 84. Ke7 Kg4 85. Bh8 Kg5 86. Ke6 Kg4 87. Be5 Ba4 88. Kf6 Bb5 89. Ke7 Kg5 90. Bc7 Kg4 91. Kd8 Kg5 92. Kc8 Kh5 93. Bd6 Kg4 94. Kb7 Kf3 95. g4 Kxg4 96. Kb6 Kf3 97. Be5 Ke4 98. Kc5 Ba4 99. Kc4 Bb5+ 100. Kc5 Ba4 101. Bg3 Bb5 102. Bd6 Ba4 103. Bc7 Bb5 104. Bd6 Ba4 105. Kc4 Bb5+ 106. Kc5 1/2-1/2

Partita 20 - Bianco: Stockfish Nero: AlphaZero
1. d4 {da libro} d5 {da libro} 2. c4 {da libro} Nc6 {da libro} 3. Nf3 {da libro} Bg4 {da libro} 4. cxd5 {da libro} Bxf3 {da libro} 5. gxf3 {da libro} Qxd5 {da libro} 6. e3 {da libro} e5 {da libro} 7. Nc3 {da libro} Bb4 {da libro} 8. Bd2 {da libro} Bxc3 {da libro} 9. bxc3 {da libro} Qd6 {da libro} 10. Qb3 Nge7 11. Qxb7 O-O 12. Qa6 Rfd8 13. Rd1 Rab8 14. h4 h5 15. Be2 Rb6 16. Qc4 Rdb8 17. Qa4 Qg6 18. Kf1 Rb1 19. e4 exd4 20. cxd4 Qd6 21. d5 Ne5 22. f4 N5g6 23. Rh3 c6 24. f5 Ne5 25. Rxb1 Rxb1+ 26. Kg2 Nxf5 27. Bxh5 Qc5 28. Bd1 Rb2 29. Rc3 Nxh4+ 30. Kf1 Qb6 31. Be3 Qd8 32. Bc1 Rb6 33. Qxa7 cxd5 34. exd5 Rb5 35. d6 Qxd6 36. Bc2 g5 37. Ba3 Qd8 38. Be7 Qe8 39. Bf6 Rb8 40. a4 Ng4 41. Qe7 Nh2+ 42. Ke2 Ng4 43. Bxg5 Qxe7+ 44. Bxe7 Re8 45. Rc7 Nf6 46. Kd3 Nd5 47. Rc4 Nf3 48. Bd6 Rd8 49. Rg4+ Kh8 50. Ke2 Nf6 51. Rc4 Ng1+ 52. Kf1 Rxd6 53. Kxg1 Ra6 54. Rc5 Nd7 55. Rd5 Nf6 56. Rf5 Rc6 57. Bd3 Rc1+ 58. Kg2 Kg7 59. a5 Rc3 60. Rf3 Rc1 61. a6 Ne8 62. Re3 Nc7 63. Bf1 Ra1 64. Re7 Nxa6 65. Ra7 Rxf1 66. Kxf1 Nc5 67. Ke2 Kg6 68. Ke3 f6 69. Rc7 Ne6 70. Rc6 Ng7 71. Kf4 Nf5 72. Ke4 Ne7 73. Rc5 Kg7 74. f4 Kf7 75. f5 Kg7 76. Kf4 Kg8 77. Kg4 1-0

Tabella 1: partite giocate da AlphaZero contro Stockfish, selezionate dal GM Matthew Sadler. Le prime 10 partite iniziano con una serie di aperture standard, mentre le rimanenti 10 partono da alcune posizioni iniziali proposte nel campionato mondiale TCEC del 2016.

Conclusioni

La strategia del Reinforcement Learning ha avuto uno sviluppo eccezionale da quando è stata concepita negli anni ‘50; negli ultimi decenni ha trovato degli ambiti di applicabilità molto vari tra loro. Dai progressi teorici e concettuali degli anni ‘90 il Reinforcement Learning ha conquistato il gioco del Go e degli scacchi. Fu proprio lo scontro del 2016 contro il campione del mondo di Go Lee Sedol a rendere famoso questo tipo di intelligenza artificiale, così astratta e distante, ma anche così simile all’uomo per il modo di ragionare. Divenne storica la mossa numero 37 effettuata nella terza partita disputata in questo match, definita dai commentatori di tutto il mondo “bellissima”, “creativa” e “unica”. Lo stesso Lee Sedol, commentando la mossa 37, ha dichiarato: “Questa mossa mi fa pensare al Go in un'altra luce.”

Sono commenti che non ci si aspetta di dare ad una azione compiuta da un motore artificiale, eppure è proprio la sua particolarità di bilanciamento tra *exploration ed exploitation* che rende il Reinforcement Learning diverso dalle rigide strutture delle altre strategie di Machine Learning.

Il Reinforcement Learning si è inoltre inserito naturalmente nelle aziende ed industrie, mostrando sempre più benefici nelle sfide della nostra società moderna. Nei prossimi anni si inserirà sempre di più nella nostra quotidianità in molti modi diversi, ma non prima di aver trovato delle soluzioni ai suoi problemi di interpretabilità e responsabilità. Il futuro del Reinforcement Learning si prospetta nondimeno luminoso e duraturo, e vedremo grandi cose da questa potente area dell’intelligenza artificiale.

L’utilizzo di motori nei giochi da tavolo, come scacchi e Go, ha cambiato l’idea dell’uomo di gioco; da istruire un agente in modo da fargli compiere le più semplici azioni si è passati ad imparare da essi. Discutendo di questo argomento si può benissimo essere a favore o contro dell’utilizzo dell’intelligenza artificiale in questi ambiti, con validi punti di argomentazione per entrambi gli schieramenti. Quello però che è indiscutibile è l’influenza e lo stupore che l’intelligenza artificiale ha portato in questi giochi.

Bibliografia

I. Richard S. Sutton and Andrew G. Barto, “Reinforcement Learning: An Introduction”, The MIT Press Cambridge, Massachusetts

II. David Silver, Thomas Hubert, Julian Schrittwieser, “A general reinforcement learning algorithm that masters chess, shogi and Go through self-play”, University College London

III. Claude E. Shannon, “Programming a Computer for Playing Chess”, Philosophical Magazine, Ser.7, Vol. 41, No. 314 - March 1950.

IV. Marvin Minsky, “Step Towards Artificial Intelligence”, Member, Ire, 1960.

V. Chess.com https://www.chess.com/terms/alphazero-chess-engine

VI. Chess.com https://www.chess.com/article/view/computers-and-chess---a-history

VII. Chess Programming https://www.chessprogramming.org/

VIII. Heavy.ai https://www.heavy.ai/technical-glossary/parallel-computing

IX. Paessler https://blog.paessler.com/the-history-of-chess-ai

X. The Atlantic https://www.theatlantic.com/

XI. Towards Data Science https://towardsdatascience.com/

XII. Towards Data Science https://towardsdatascience.com/

XIII. Treccani https://www.treccani.it/

XIV. V7labs https://www.v7labs.com/blog/reinforcement-learning-applications

XV. Wikipedia https://en.wikipedia.org/wiki/Alpha%E2%80%93beta_pruning

XVI. Wikipedia https://en.wikipedia.org/wiki/Computer_chess

XVII. Wikipedia https://it.wikipedia.org/wiki/Regola_delle_cinquanta_mosse