ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Dipartimento di Scienze

Corso di Laurea in Informatica

Analisi e comparazione delle maggiori risoluzioni algoritmiche di giochi ad informazione imperfetta applicate al poker Texas Hold'em Heads-Up Limit

Relatore:
Chiar.mo Prof.
Pietro Di Lena

Presentata da:
Matteo Toccarelli

Sessione di Dicembre 2025

Anno Accademico 2024/2025

Abstract

La risoluzione computazionale di giochi a somma zero in forma estesa rappresenta una delle sfide centrali nella teoria dei giochi algoritmica, data la complessità intrinseca dei giochi ad informazione imperfetta e la necessità di ottenere strategie approssimativamente ottimali con risorse limitate. In questo contesto, la presente tesi offre un'analisi approfondita dei principali approcci noti in letteratura per la soluzione di tali giochi. Il lavoro utilizza come modello applicativo il poker Heads-Up Limit Texas Hold'em (HULHE), uno dei giochi imperfetti più studiati per la sua rilevanza teorica e pratica. Per ciascun algoritmo considerato, è stata implementata una variante specificamente adattata a HULHE, con l'obiettivo di valutarne le prestazioni nel confronto diretto. La tesi presenta complessivamente 23 varianti algoritmiche, organizzate in base ai principi metodologici da cui derivano, fornendo una panoramica sistematica delle strategie di risoluzione più influenti e delle loro configurazioni pratiche. Inoltre, viene proposto un approccio euristico che, seppur fondato su metodi già utilizzati, rappresenta una variante innovativa, non presente in letteratura, e che costituisce il contributo originale di questa tesi al problema di gioco tra agenti artificiali nell'HULHE. Prima dell'analisi comparativa, viene fornita una descrizione dettagliata della struttura del gioco HULHE, delle nozioni fondamentali della teoria dei giochi rilevanti per il lavoro e dei criteri metodologici adottati per valutare gli algoritmi. Infine, vengono discussi gli esiti sperimentali dei confronti diretti tra i diversi approcci e viene fornita una sintesi critica dei risultati ottenuti, includendo le principali limitazioni emerse durante lo sviluppo delle implementazioni. Questa tesi si configura quindi come un contributo significativo alla valutazione empirica e all'applicazione pratica degli algoritmi per giochi ad informazione imperfetta nel contesto del poker HULHE.

Indice

Capitolo 1

Introduzione

1.1 Teoria dei Giochi nell'Informatica e poker

Sin dai primi sviluppi nel campo dell'informatica, i giochi hanno rappresentato un contesto privilegiato per lo sviluppo dell'intelligenza artificiale (IA) e dei primi modelli computazionali. Per oltre mezzo secolo, essi hanno svolto un ruolo centrale nella validazione di nuove idee e tecniche, contribuendo a traguardi fondamentali nell'avanzamento dell'IA. Tra i risultati più emblematici si annoverano Chinook, primo programma di dama a conquistare un titolo mondiale contro avversari umani [72], Deep Blue, capace di sconfiggere il campione del mondo Garry Kasparov negli scacchi [19], e Watson, vincitore contro campioni umani nel gioco Jeopardy! [28]. Tuttavia, il superamento dei migliori giocatori umani non coincide necessariamente con la "risoluzione" di un gioco, intesa come la determinazione di una strategia teoricamente ottimale che non possa essere sconfitta da alcun avversario in condizioni di gioco corrette.

Il processo di risoluzione di un gioco costituisce infatti un obiettivo fondamentale nello studio dell'IA e nell'analisi algoritmica dei giochi strategici. Tra i giochi risolti fino ad oggi, la quasi totalità appartiene alla categoria dei giochi a informazione perfetta, nei quali ogni giocatore dispone della piena conoscenza di tutti gli eventi precedenti al momento della decisione. Scacchi, dama e backgammon rientrano in tale classe. Al contrario, nei giochi a informazione imperfetta i giocatori non possono osservare completamente lo stato del gioco, come accade nel bridge, nel poker o nelle aste, dove parte delle informazioni rimane nascosta. La presenza di informazioni incomplete rende questi giochi più complessi sia dal

punto di vista teorico sia dal punto di vista computazionale, con un ritardo significativo, nella letteratura, rispetto ai progressi ottenuti per i giochi a informazione perfetta. È inoltre rilevante osservare che, mentre quest'ultima categoria risulta comune nei giochi da tavolo, l'informazione imperfetta è la norma nei processi decisionali reali. In questo senso, già J. von Neumann, in una conversazione raccontata da J. Bronowski, sottolineava come la vita reale fosse caratterizzata da bluff, inganni e inferenza sulle intenzioni altrui, riconoscendo in tali aspetti il fondamento stesso dei giochi strategici moderni [14].

L'affermazione di von Neumann allude al gioco per eccellenza dell'informazione imperfetta: il poker. Esso costituisce il paradigma dei giochi a informazione imperfetta e ha avuto un ruolo determinante nei primi sviluppi della teoria dei giochi. I lavori pionieristici di Émile Borel [10] e dello stesso von Neumann [86, 87] furono infatti motivati dalla formalizzazione matematica del bluff e delle decisioni probabilistiche tipiche di questo gioco [12], che oggi è tra i più diffusi al mondo. La variante attualmente più popolare è il Texas Hold'em; quando è disputata tra due soli giocatori, con puntate fisse e numero massimo di rilanci, prende il nome di Heads-Up Limit Texas Hold'em (HULHE).

HULHE rappresenta la più piccola variante di poker praticata a livello competitivo, pur presentando una complessità considerevole: il gioco può assumere circa $3{,}16 \times 10^{17}$ stati distinti, una dimensione superiore a quella di Forza Quattro ma inferiore a quella della dama. Tuttavia, la natura dell'informazione imperfetta riduce la possibilità dei giocatori di distinguere molti di questi stati, che differiscono solo per informazioni non osservabili, come le carte private dell'avversario. Questo porta a circa $3{,}19 \times 10^{14}$ punti decisionali effettivi, rendendo HULHE significativamente più impegnativo da analizzare o risolvere rispetto ai giochi a informazione perfetta di dimensione anche superiore [12].

Lo studio del poker da parte dell'IA, della ricerca operativa e della psicologia ha una storia ormai cinquantenaria. Solo ventisette anni fa Koller e Pfeffer affermavano che una soluzione su larga scala di giochi complessi come il poker fosse probabilmente irraggiungibile [48]. Allis [1], informatico olandese che ha ottenuto successo nella risoluzione per giochi come Forza Quattro e Qubic, fornisce tre diverse definizioni di risoluzione di una partita: un gioco è ultra-debolmente risolto se è noto il valore teorico della posizione iniziale; debolmente risolto se esiste una strategia che garantisce almeno tale valore teorico; fortemente risolto se per ogni posizione legale è nota una strategia che garantisce il valore ottimale. Tuttavia, nei giochi a informazione imperfetta tali definizioni incontrano difficoltà strutturali, poiché il valore teorico di una posizione oltre quella iniziale non è univocamente determinato.

Inoltre, la presenza di comportamenti stocastici implica valori non discreti, raggiungibili solo in media su un numero elevato di partite. Di conseguenza, i valori teorici dei giochi a informazione imperfetta sono spesso approssimati, quindi un'ulteriore considerazione nella risoluzione di un gioco è il grado di approssimazione accettabile.

Questa tesi si concentra proprio su HULHE, analizzando le tecniche di risoluzione approssimata sviluppate negli ultimi anni e valendosi dei risultati ottenuti dal gruppo di ricerca dell'Università di Alberta [12], grazie allo sviluppo di algoritmi di apprendimento per rinforzo e di metodi di riduzione dello spazio degli stati. In particolare, si focalizza sull'algoritmo di minimizzazione del rimpianto controfattuale e sulle sue varianti, che hanno permesso di avvicinarsi in modo significativo a una strategia quasi perfetta per HULHE. Inoltre, viene proposto un nuovo approccio euristico, non presente nella letteratura: *opponent modeling expectimax*.

1.2 Equilibrio di Nash e sfruttabilità

La rappresentazione classica di un contesto informativo imperfetto è il gioco in forma estesa, un modello formale di interazione strategica tra agenti razionali applicabile tanto ai contesti ludici quanto a scenari reali quali aste, negoziazioni o problemi di sicurezza. Il nucleo di un gioco in forma estesa è costituito da un albero di gioco che descrive la sequenza dei possibili eventi, comprendenti le azioni dei giocatori e gli esiti casuali. Ogni ramo dell'albero è associato allo stato del gioco in cui si verifica l'evento corrispondente, mentre a ciascun nodo decisionale è assegnato un giocatore (o il caso) responsabile della scelta dell'azione successiva. Le foglie dell'albero denotano la conclusione della partita e sono etichettate con le utilità associate ai giocatori [12].

Negli stati in cui un giocatore deve prendere una decisione, l'informazione disponibile è organizzata in insiemi informativi, ovvero insiemi di stati che il giocatore non è in grado di distinguere, come accade quando l'incertezza deriva da elementi non osservabili, ad esempio le carte private dell'avversario nel poker. Le azioni disponibili in ciascun insieme informativo sono rappresentate dai rami in uscita da qualsiasi stato compreso nell'insieme stesso (figura 1.1).

Una strategia per un giocatore è definita come l'assegnazione, per ogni insieme informativo, di una distribuzione di probabilità sulle azioni disponibili. Quando il gioco coinvolge esattamente due giocatori e le utilità terminali sommano sempre a zero, si parla di gioco

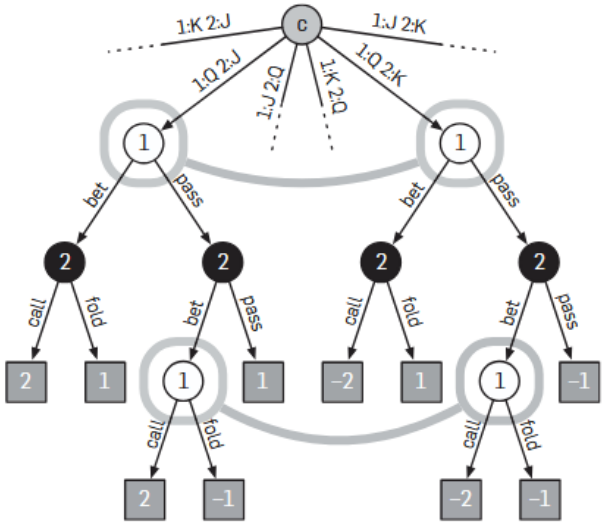

Figura 1.1: Parte della rappresentazione in forma estesa del poker Kuhn. Le frecce mostrano gli eventi tra cui il giocatore che agisce può scegliere, etichettati con il loro significato nel gioco. Le foglie sono vertici quadrati etichettati con l'utilità associata per il giocatore 1 (l'utilità del giocatore 2 è la negazione di quella del giocatore 1). Gli stati collegati da linee grigie spesse fanno parte dello stesso insieme di informazioni [12].

a somma zero. Il concetto classico di soluzione per i giochi estesi è l'equilibrio di Nash, ossia un profilo di strategie in cui nessun giocatore può migliorare la propria utilità attesa modificando unilateralmente la propria strategia. In un gioco in forma estesa finita, esiste almeno un equilibrio di Nash; nei giochi a somma zero, tutti gli equilibri producono la stessa utilità attesa per ogni giocatore, denominata valore del gioco.

Si definisce invece equilibrio ϵ-Nash una strategia per cui nessun giocatore può aumentare la propria utilità attesa di più di ϵ adottando una strategia alternativa. In accordo con le categorie introdotte da Allis, un gioco a somma zero è considerato ultra-debolmente risolto quando ne è determinato il valore teorico, mentre è debolmente risolto quando è identificata una strategia di equilibrio di Nash. Un gioco è definito essenzialmente debolmente risolto quando si ottiene un equilibrio ϵ-Nash con un valore di ϵ sufficientemente piccolo da risultare statisticamente indistinguibile da zero nell'arco di una vita umana di partite giocate [12].

La qualità dell'approssimazione prodotta può essere valutata attraverso la misura della sfruttabilità (exploitability), definita come la differenza tra il valore del gioco e l'utilità attesa ottenibile giocando la strategia considerata contro la miglior risposta dell'avversario

nel caso peggiore. Una strategia fortemente sfruttabile si discosta significativamente dall'equilibrio, mentre una strategia con sfruttabilità prossima a zero è, di fatto, una strategia quasi perfetta.

Va osservato che una strategia può risultare sfruttabile in aspettativa, ma ciò non implica necessariamente che l'avversario ottenga un guadagno significativo in un numero finito di mani, poiché gli esiti del gioco sono soggetti a variabilità stocastica e la strategia stessa include componenti di randomizzazione. Per tale motivo, si introduce il concetto di soluzione essenzialmente risolta. A titolo illustrativo, se si considera un individuo che giochi 200 mani di poker all'ora, per 12 ore al giorno, senza interruzioni, per un periodo di 70 anni, applicando sempre la miglior risposta possibile contro la strategia in esame e non commettendo alcun errore, i risultati di un numero così elevato di mani sarebbero in ogni caso soggetti al teorema del limite centrale, con la distribuzione delle vincite complessive che tende a una normale. Ne consegue che, almeno una volta su venti, l'esito totale potrà risultare 1,64 deviazioni standard al di sotto del valore atteso, anche contro una strategia perfettamente ottimale [12].

Utilizzando il valore della deviazione standard per singola mano in HULHE, riportato pari a circa 5 big blind per partita (5 bb/g), è possibile derivare una soglia di sfruttabilità tale da rendere statisticamente indistinguibile una strategia approssimata da una soluzione perfetta. In particolare, una strategia con una sfruttabilità inferiore a 1 milli-big-blind per partita (1 mbb/g) non può essere distinta da una strategia di equilibrio neppure su un orizzonte pari alla vita intera del giocatore nel caso peggiore. Essa presenta, addirittura, una probabilità non trascurabile (1 su 20) di prevalere contro la miglior risposta assoluta anche dopo milioni di mani [12].

Per questa ragione, la soglia di 1 mbb/g è adottata come criterio per dichiarare l'HULHE essenzialmente debolmente risolto. Strategie la cui sfruttabilità ricade al di sotto di tale limite sono, a tutti gli effetti pratici, indistinguibili da una soluzione esatta del gioco.

1.3 Giochi in forma estesa e tecniche di astrazione

Nel poker, una strategia può essere rappresentata come una terna di probabilità (f,c,r) associata a ciascun insieme informativo. In tale notazione, f indica la probabilità di abbandonare (fold), c la probabilità di vedere o chiamare (call), ed r la probabilità di puntare o rilanciare (raise), con il vincolo f+c+r=1. Una risposta ottimale è definita come la strategia

che massimizza l'utilità attesa di un agente contro l'insieme delle strategie adottate dagli altri partecipanti al gioco. Ogni strategia che compone un profilo di equilibrio di Nash costituisce, per definizione, una risposta ottimale alle strategie degli altri agenti presenti nel profilo. Se un singolo agente mantiene la propria strategia di equilibrio mentre gli altri deviano, esso potrebbe ottenere un'utilità maggiore o minore deviando a sua volta; tuttavia, nessuna deviazione unilaterale risulta vantaggiosa se tutti mantengono la strategia prevista dal profilo di equilibrio. Nei giochi a somma zero e a due giocatori, i profili strategici di equilibrio assumono un'importanza particolare, poiché presentano proprietà aggiuntive che li rendono strumenti fondamentali per l'analisi e la soluzione del gioco [93].

L'astrazione delle carte costituisce il metodo più diffuso per ridurre la complessità dell'albero decisionale nei giochi di poker. L'approccio più semplice consiste nell'applicare una metrica alle mani, come la forza della mano attesa (Expected Hand Strength, E[HS]) [6], e nel raggruppare all'interno dello stesso bucket le mani che presentano valori metrici simili. Nel bucket percentile, ciascun raggruppamento contiene approssimativamente lo stesso numero di mani, mentre nel bucket uniforme l'intervallo metrico [0,1] viene suddiviso in modo uniforme in N sottointervalli: tutte le mani il cui valore ricade in [0,1/N] vengono assegnate al primo bucket, quelle in [1/N,2/N] al secondo, e così via.

Nel poker esistono però molte mani che, pur non essendo particolarmente forti in una fase iniziale, possiedono un elevato potenziale di miglioramento nelle fasi successive (ad esempio, un progetto di scala o di colore). Una metrica alternativa particolarmente efficace, in grado di incorporare tale potenziale, è la forza della mano attesa al quadrato, ossia $E[HS^2]$ [41]. Anche questa metrica è definita nell'intervallo [0,1] e valuta il valore atteso del quadrato della forza della mano, enfatizzando le mani con alto potenziale futuro.

Attraverso queste tecniche di astrazione, un gioco di carte viene trasformato in un gioco basato su bucket, in cui tutte le mani appartenenti allo stesso bucket vengono trattate come indistinguibili e quindi giocate nello stesso modo. Gli insiemi informativi rappresentano quindi i bucket anziché mani individuali. Se l'astrazione è costruita in modo che ciascun giocatore mantenga memoria perfetta di tutte le azioni precedenti contenute nei rispettivi insiemi informativi, essa è definita astrazione a richiamo perfetto. Con richiamo perfetto e N bucket per ciascun round, si ottengono N sequenze preflop, N^2 sequenze al flop, N^3 al turn e N^4 al river. Nel caso di un'astrazione a 2 bucket, ciò comporta $2^4 = 16$ sequenze di bucket al river.

Per ridurre ulteriormente le dimensioni dell'albero di gioco, è possibile adottare un'astrazione a richiamo imperfetto, in cui gli insiemi informativi derivanti da bucket diversi possono confluire in uno stesso insieme informativo nei round successivi, riducendo così in maniera significativa le dimensioni dell'albero di gioco. L'albero risultante assume la struttura di un grafo aciclico orientato, più compatto rispetto all'albero completo. Nei casi più estremi, i bucket dei round precedenti vengono completamente dimenticati: il giocatore conserva esclusivamente la sequenza di puntate (betting sequence) e il bucket del round corrente. Ad esempio, invece di adottare un'astrazione a richiamo perfetto con 2 bucket per round, è possibile costruire un albero di gioco di dimensioni comparabili utilizzando un'astrazione a richiamo imperfetto con 16 bucket. Tale approccio consente una rappresentazione più fine delle mani pur mantenendo invariata, o quasi, la complessità computazionale complessiva [93].

1.4 HULHE: regole e logica di gioco

Il Texas Hold'em Heads-Up Limit [13] è una variante del poker caratterizzata da una struttura a due giocatori e da una natura ripetuta della partita: i contendenti disputano una sequenza di mani successive, alternandosi nel ruolo di dealer. In ciascuna mano uno dei due giocatori ottiene un certo numero di fiches dall'avversario, e l'obiettivo complessivo è massimizzare il proprio guadagno lungo l'intero arco della sessione.

Ogni mano ha inizio con il versamento obbligatorio dei bui: il giocatore iniziale posta lo small blind, mentre l'altro piazza il big blind, pari al doppio dello small blind. La partita procede poi attraverso quattro round distinti — preflop, flop, turn e river — ciascuno composto da una fase di distribuzione delle carte e da una fase di puntate. Nel preflop, a entrambi i giocatori vengono assegnate due carte private non visibili all'avversario; nei round successivi vengono invece rivelate carte comuni al centro del tavolo, per un totale di cinque: tre al flop, una al turn e una al river.

Dopo la distribuzione, i giocatori agiscono alternandosi tra tre opzioni: fold, call o raise. Passare (fold) significa rinunciare a pareggiare l'ultima puntata avversaria, cedendo immediatamente il piatto. Il call ("chiamare" o "vedere") consiste nell'aggiungere al piatto l'importo necessario per eguagliare la puntata corrente, consentendo il passaggio al round successivo. Il rilancio (raise), infine, comporta sia il pareggio della puntata avversaria sia l'aggiunta di ulteriori fiches per imporre una nuova puntata. All'inizio di un round, quando

non sono presenti puntate, un rilancio viene denominato bet, ossia una puntata, mentre una chiamata priva di costo prende il nome di check; se entrambi i giocatori effettuano check, il round termina.

Nel formato Limit, l'ammontare delle puntate è fisso: nei round preflop e flop l'importo è detto small bet ed è pari al big blind, mentre nei round turn e river si usa la big bet, pari al doppio del big blind. L'azione procede a partire dal giocatore non-dealer in tutti i round, tranne nel preflop, dove tocca al dealer decidere se abbandonare, chiamare o rilanciare la puntata del big blind. In ogni round sono consentite al massimo quattro puntate o rilanci complessivi; una volta raggiunto il limite, il giocatore che deve agire può soltanto chiamare o passare.

Se anche dopo il river nessun giocatore ha scelto di abbandonare, la mano si conclude con lo showdown: entrambi rivelano le proprie carte private e il vincitore è determinato dalla migliore combinazione di cinque carte ottenibile utilizzando liberamente le due carte personali e le cinque comuni. Indipendentemente dall'esito (abbandono o showdown), al termine della mano i giocatori si scambiano i ruoli e ne avviano una nuova.

Poiché il gioco può essere disputato con strutture di puntata molto diverse, le prestazioni dei giocatori vengono tipicamente misurate in unità normalizzate, ossia i milli-big-blind per mano (mbb/g). Un milli-big-blind corrisponde a un millesimo di big blind, e questa metrica consente di confrontare strategie indipendentemente dalla posta specifica. Un giocatore che foldasse sistematicamente perderebbe in media 750 mbb/g, corrispondenti ai contributi obbligatori dei bui. Nel gioco professionale, è opinione comune che un giocatore esperto debba mirare ad ottenere almeno 50 mbb/g dai propri avversari. La stessa unità di misura viene utilizzata anche per quantificare la sfruttabilità.

1.5 Approcci considerati ed obiettivo della tesi

Alla luce dei concetti esposti nei paragrafi precedenti, risulta possibile delineare con chiarezza il quadro teorico entro il quale si colloca questa tesi. L'analisi del poker Heads-Up Limit Hold'em consente infatti di ricondurre il problema della decisione ottimale a un insieme strutturato di principi propri della teoria dei giochi. Nozioni quali equilibrio di Nash, strategia ottimale, sfruttabilità e valore atteso assumono un ruolo centrale sia nella rappresentazione formale del dominio sia nella valutazione delle strategie computazionali che verranno discusse.

Il carattere sequenziale e stocastico dell'HULHE, unito alla natura imperfetta dell'informazione disponibile ai giocatori, rende il problema particolarmente adatto allo studio di algoritmi di ottimizzazione strategica. Tali algoritmi mirano ad approssimare strategie sempre più vicine all'equilibrio o, in alternativa, a ottenere politiche particolarmente adatte nello sfruttamento delle debolezze di avversari non ottimali.

L'obiettivo principale della tesi è confrontare una selezione rappresentativa di tali metodi, appartenenti a diverse famiglie concettuali. In particolare, verranno approfonditi gli algoritmi basati sulla minimizzazione del rimpianto (regret minimization), le tecniche riconducibili al fictitious play e alle sue estensioni neurali, le metodologie di ricerca e simulazione, nonché approcci alternativi quali gli algoritmi euristici e i metodi fondati sulla programmazione lineare. Il confronto tra queste differenti paradigmi consentirà di evidenziare punti di forza, limiti operativi ed eventuali particolarità e similitudini, con l'obiettivo di ottenere una visione organica delle principali strategie computazionali applicabili all'HULHE. Inoltre, verrà proposto un nuovo contributo algoritmico, basato su un modello euristico già esistente, ma adattato allo specifico contesto del poker HULHE e migliorato concettualmente.

I capitoli successivi presenteranno quindi in modo sistematico le famiglie di algoritmi considerate, illustrandone i principi teorici, le modalità di implementazione e i risultati sperimentali ottenuti. Tale percorso consentirà di valutare criticamente l'efficacia delle diverse soluzioni analizzate nel contesto specifico del poker Heads-Up Limit Hold'em, contribuendo così a una comprensione approfondita delle tecniche di risoluzione nei giochi a informazione imperfetta.

Capitolo 2

Approcci computazionali

2.1 Counterfactual Regret Minimization

La famiglia di algoritmi *Counterfactual Regret Minimization* (CFR) rappresenta oggi l'approccio più diffuso e di maggiore successo per l'approssimazione di equilibri in giochi a informazione imperfetta. Il metodo si basa su un processo iterativo di auto gioco tra due algoritmi che minimizzano il *rimpianto*, ovvero la perdita di utilità che un algoritmo subisce per non aver selezionato la migliore strategia deterministica. L'equilibrio di Nash approssimato è ottenuto attraverso la media delle strategie iterate dei giocatori, con qualità crescente al crescere del numero di iterazioni.

Nel corso degli anni, l'algoritmo originario CFR è stato oggetto di numerose estensioni e varianti — tra cui CFR+, TCFR, DDCFR, ECFR, DeepCFR, SDCFR, RCFR, CFR-BR e MCCFR — sviluppate con l'obiettivo di migliorare l'efficienza computazionale, la velocità di convergenza e la qualità delle strategie ottenute, portando inoltre alla risoluzione essenzialmente debole di HULHE nel 2015 [12].

L'analisi comparativa di tali approcci risulta pertanto cruciale per comprendere l'evoluzione delle tecniche moderne di risoluzione dei giochi ad informazione imperfetta e il loro impatto nelle applicazioni al poker competitivo.

2.1.1 CFR

Contesto e descrizione algoritmo. L'approccio di CFR (Counterfactual Regret Minimization), così come le varianti basate su di esso, ha come aspetto fondamentale, come accennato in precedenza, ciò che nella letteratura ad esso dedicata viene indicato come "rimpianto" [12].

Il concetto di rimpianto rappresenta la differenza tra l'utilità massima potenzialmente ottenibile scegliendo, in retrospettiva, l'azione migliore possibile, e l'utilità effettivamente conseguita mediante l'azione intrapresa. CFR si propone di minimizzare, per ogni insieme informativo, il rimpianto controfattuale immediato positivo [68].

Nel lavoro fondativo di Zinkevich et al. (2008) [93] è dimostrato formalmente che ridurre il rimpianto controfattuale immediato positivo equivale a minimizzare il rimpianto medio complessivo. Inoltre, in un gioco a due giocatori con somma zero e richiamo perfetto, minimizzare il rimpianto medio di entrambi i giocatori conduce a un profilo strategico che è un equilibrio ϵ-Nash.

Johanson [41] ha fornito una descrizione dettagliata dell'implementazione del CFR per giochi generali a due giocatori con richiamo perfetto e, in particolare, per il poker heads-up. Questo approccio ha segnato una svolta nel calcolo di strategie di equilibrio, soprattutto in giochi a informazione imperfetta, poiché la complessità di memoria cresce proporzionalmente al numero di insiemi informativi e non al numero totale degli stati, un vantaggio cruciale rispetto ai risolutori basati sulla forma di sequenza [12]. Grazie a tali risparmi, Zinkevich et al. [93] sono riusciti a risolvere astrazioni di poker molto più grandi rispetto alle tecniche precedenti. Un'altra proprietà importante del CFR è la capacità di calcolare la risposta ottimale (best response) in giochi astratti contro un avversario statico, come discusso da Johanson [41].

L'algoritmo CFR si fonda sull'idea di decomporre il rimpianto complessivo accumulato da un giocatore nel corso delle partite in una serie di termini elementari, ciascuno associato a un singolo insieme informativo del gioco. Tale decomposizione consente di minimizzare il rimpianto in maniera indipendente in ciascun punto decisionale, rendendo così possibile l'apprendimento progressivo di strategie che convergono verso un equilibrio approssimato. Il concetto centrale introdotto dall'algoritmo è quello di counterfactual regret, un tipo di rimpianto definito a livello dell'insieme informativo e formulato affinché contribuisca a limitare

il rimpianto globale del giocatore [93].

A partire da ciò, l'utilità controfattuale $u_i(\sigma, I)$ del giocatore i (dove σ è la strategia complessiva) viene definita come l'utilità attesa condizionata sul raggiungimento dell'insieme informativo I, assumendo che tutti i giocatori seguano la strategia σ, con l'eccezione del fatto che il giocatore i forza il gioco verso l'insieme informativo stesso. Tale quantità incorpora, quindi, la probabilità controfattuale con cui I sarebbe stato raggiunto se il giocatore avesse scelto di perseguirlo.

Per ogni azione $a \in A(I)$, si definisce inoltre la strategia modificata $\sigma|_{I \to a}$, identica a σ tranne per il fatto che in I il giocatore i seleziona determinatamente l'azione a. Il *rimpianto controfattuale immediato*, ovvero la quantità che l'algoritmo si propone di minimizzare, misura la differenza tra l'utilità ottenuta scegliendo un'azione alternativa rispetto a quella effettivamente adottata in ciascuna iterazione [93].

Un risultato fondamentale dimostrato dallo studio di Zinkevich et al. (2008) [93] afferma che il rimpianto complessivo è limitato dalla somma dei rimpianti controfattuali positivi accumulati nei singoli insiemi informativi. Ciò implica che la minimizzazione del rimpianto controfattuale in ciascun punto decisionale garantisce automaticamente la minimizzazione del rimpianto globale del giocatore, permettendo dunque l'avvicinamento a un equilibrio di Nash anche senza operare su uno spazio di decisione complessivo.

Per realizzare tale minimizzazione locale, l'algoritmo adotta un meccanismo di aggiornamento ispirato alla teoria dell'approachability di Blackwell [93]. Per ogni insieme informativo e per ogni azione, viene mantenuta una stima del rimpianto medio accumulato nelle iterazioni precedenti. Al termine di ciascuna iterazione, questi valori sono aggiornati in base alla differenza tra l'utilità controfattuale associata all'azione considerata e quella derivante dalla strategia attuale. La strategia per l'iterazione successiva viene quindi ottenuta assegnando probabilità proporzionali ai rimpianti positivi delle azioni disponibili; nel caso in cui tali rimpianti risultino nulli, le azioni vengono selezionate in maniera uniforme. Tale regola di aggiornamento consente di garantire una decrescita sub-lineare del rimpianto nel tempo, e quindi di convergere verso una strategia approssimativamente equilibrata.

Costo computazionale. Una volta definita un'astrazione appropriata del gioco, CFR viene utilizzato per calcolare una strategia di equilibrio approssimata dell'intero gioco astratto. L'algoritmo procede memorizzando e aggiornando per ogni insieme informativo i valori di rimpianto e le strategie correnti, consentendo l'apprendimento progressivo di una politica

sempre più solida. È stato descritto, inoltre, un metodo di campionamento delle azioni, che riduce il numero di stati rilevanti in ciascuna iterazione e consente un'accelerazione computazionale significativa senza comprometterne la correttezza teorica [93].

Risultati raggiunti. Sebbene le garanzie teoriche dell'algoritmo CFR siano originariamente limitate a giochi a due giocatori, somma zero e richiamo perfetto, nella pratica il CFR si è dimostrato sorprendentemente robusto anche quando alcune di queste condizioni vengono allentate. In particolare, sono stati sviluppati agenti di poker heads-up con richiamo imperfetto che si sono rivelati estremamente efficaci [68]. Ulteriori ricerche hanno esteso l'utilizzo del CFR anche ai giochi a somma non zero, generando agenti con bassa sfruttabilità che risultano particolarmente efficaci contro giocatori umani, grazie a una maggiore aggressività combinata con la tendenza umana ad abbandonare frequentemente [68].

Nonostante le sue potenzialità, il CFR non offre comunque garanzie teoriche generali per i giochi multigiocatore a somma zero. Tuttavia, nel corso di alcuni esperimenti, è stato dimostrato che il CFR può effettivamente generare agenti vincenti anche in scenari con più giocatori. In particolare, un agente a tre giocatori con richiamo imperfetto e un altro con richiamo perfetto, entrambi prodotti da CFR, ottennero rispettivamente il primo e il secondo posto nella CP Competition del 2009 [68].

2.1.2 CFR+

Contesto e descrizione algoritmo. Il *Counterfactual Regret Minimization Plus* (CFR+) costituisce una variante avanzata dell'algoritmo CFR, progettata nel 2015 specificamente per affrontare la risoluzione di giochi a informazione imperfetta caratterizzati da alberi di gioco di dimensioni estremamente elevate, come nel caso dell'Heads-Up Limit Hold'em (HULHE) [12].

L'applicazione delle varianti consolidate di CFR a giochi di tale scala incontra due sfide fondamentali: la gestione della memoria e i requisiti computazionali. Da un lato, l'algoritmo deve memorizzare sia la strategia risultante sia i rimpianti accumulati per ogni insieme informativo. Anche adottando una rappresentazione con numeri in virgola mobile a precisione singola (4 byte), lo spazio di archiviazione richiesto ammonterebbe a circa 262 terabyte [12], una quantità incompatibile con le risorse di memoria principale dei sistemi di

calcolo. Dall'altro lato, l'esperienza maturata nell'ambito dell'algoritmica per giochi estesi indica che un incremento di tre ordini di grandezza nel numero di insiemi informativi tende a tradursi, almeno, in un aumento di pari entità del tempo computazionale richiesto.

CFR+ nasce proprio per superare parte di tali limitazioni [12]: le implementazioni convenzionali di CFR si basano su un campionamento selettivo delle porzioni dell'albero da aggiornare a ogni iterazione e adottano il metodo del *regret-matching*, il quale conserva e aggiorna i rimpianti di ciascuna azione, selezionandole con probabilità proporzionale ai rimpianti positivi. Al contrario, CFR+ compie iterazioni complete sull'intero albero di gioco e utilizza il *regret-matching+*, una variante in cui i rimpianti negativi vengono troncati a zero. Questo accorgimento consente alle azioni precedentemente considerate subottimali — e quindi caratterizzate da rimpianti negativi — di ritornare selezionabili non appena si dimostrino nuovamente promettenti, senza attendere numerose iterazioni affinché il rimpianto accumulato diventi positivo.

Costo computazionale. Un grande vantaggio di CFR+ è stato osservato empiricamente: durante l'esecuzione, la sfruttabilità delle strategie intermedie tende a convergere regolarmente verso zero. Di conseguenza, non è necessario calcolare e memorizzare la strategia media, come avviene nel CFR tradizionale; si può invece utilizzare direttamente la strategia corrente come soluzione approssimata.

Dato un insieme di azioni A e una qualunque sequenza di funzioni valore

$$v_t : A \to \mathbb{R},$$

definita per $t = 1, \ldots, T$, esiste un limite L tale che

$$|v_t(a) - v_t(b)| \leq L \quad \text{per ogni } t \text{ e per tutti } a, b \in A.$$

In tali condizioni, un agente che seleziona le proprie azioni secondo l'algoritmo di *regret-matching+* presenta un rimpianto massimo limitato da un valore dipendente da L e dal numero di iterazioni T. In altri termini, regret-matching+ garantisce che il rimpianto cresca in maniera controllata, fornendo così una base teorica solida alle proprietà di convergenza di CFR+ [93].

Lo studio condotto dall'Università di Alberta [93] mostra come, grazie a una combinazione di compressione in streaming e gestione accurata dell'I/O su disco, il fabbisogno di memoria venga ridotto a circa 10.9 TiB. Inoltre, nonostante CFR+ conservi le stesse garanzie asintotiche di CFR classico (decrescita del rimpianto come $O(1/\sqrt{T})$), in pratica mostra una riduzione drastica del numero di iterazioni necessarie per raggiungere una data soglia di sfruttabilità. Nel lavoro viene riportato che il processo di soluzione di HULHE tramite CFR+ ha richiesto complessivamente circa 900 core-year di calcolo, distribuiti su un cluster di 4800 CPU, completando la computazione in 68 giorni di tempo reale.

Risultati. Nel lavoro dell'Università di Alberta, CFR+ viene applicato al gioco Heads Up Limit Hold'em (HULHE) portando a risultati di notevole rilevanza per il campo dei giochi a informazione imperfetta. Infatti, la ricerca documenta che il gioco è stato "essenzialmente risolto", nel senso che l'algoritmo ha generato una strategia la cui sfruttabilità è stata portata a un livello tale da renderla praticamente indiscutibile in un contesto competitivo.

Grazie a questo successo, CFR+ ha portato, per la prima volta nella storia, alla risoluzione di un gioco a informazione imperfetta praticato in modo competitivo da esseri umani [12].

2.1.3 CFR-BR

Contesto e descrizione algoritmo. L'algoritmo CFR-BR (CFR Best Response) nasce come conseguenza diretta degli studi di Waugh et al. [89, 90] sulle patologie di astrazione nei giochi estesi. In tali lavori è stato dimostrato che, quando si risolve un gioco in cui un giocatore opera mediante un'astrazione dello spazio degli stati mentre l'avversario utilizza il modello completo, un raffinamento più accurato dell'astrazione conduce a una diminuzione monotona della sfruttabilità del giocatore astratto. Inoltre, la strategia del giocatore astratto in tali condizioni coincide, per definizione, con la strategia meno sfruttabile rappresentabile all'interno dello spazio astratto, poiché diversamente non costituirebbe un equilibrio.

Sulla base di queste osservazioni, CFR-BR si propone come un algoritmo progettato specificamente per risolvere un gioco in cui l'avversario non è astratto, utilizzando la rappresentazione completa dello spazio informativo. Ciò viene ottenuto senza la necessità di

memorizzare esplicitamente l'intera strategia dell'avversario non astratto, evitando così un considerevole utilizzo di memoria [44].

L'algoritmo CFR-BR si articola in due componenti fondamentali. Il primo passo consiste nell'introdurre un meccanismo alternativo per generare la strategia dell'avversario non astratto. L'uso del CFR per aggiornare la strategia di un giocatore rappresenta una modalità per costruire un agente che minimizza il rimpianto. Tuttavia, una *best response* (BR) costituisce anch'essa un agente che minimizza il rimpianto: scegliendo sempre l'azione a valore massimo, infatti, la BR ottiene rimpianto nullo a ogni iterazione.

Nel contesto di CFR-BR, un agente che aggiorna la propria strategia scegliendo sempre la risposta ottimale all'azione dell'avversario è definito come agente BR. La sua strategia, a ogni iterazione, coincide quindi con una risposta ottimale alla strategia dell'avversario nella medesima iterazione [44]. Poiché il giocatore 1 utilizza CFR, si ha $R_1^T \leq \varepsilon$ dopo T iterazioni [93].

Costo computazionale. L'utilizzo di un agente BR non astratto comporta due vantaggi rilevanti. In primo luogo, la compattezza della rappresentazione: essendo la strategia BR una strategia pura, essa può essere rappresentata in modo molto più compatto rispetto a una strategia comportamentale che assegni probabilità a ogni azione. Inoltre, è stato dimostrato [44] che, quando un agente CFR gioca contro una BR, l'intera sequenza delle sue strategie correnti converge a un equilibrio di Nash con elevata probabilità. Nel CFR tradizionale converge soltanto la strategia media, ma in CFR-BR questa proprietà consente di evitare il tracciamento della strategia media, riducendo il fabbisogno di memoria per l'agente CFR di circa la metà.

Nonostante i suoi punti di forza concettuali, CFR-BR presenta ancora due criticità che ne ostacolano l'applicazione diretta a giochi di grandi dimensioni.

In primo luogo, sebbene una strategia di risposta ottimale possa essere rappresentata in forma compatta, la sua dimensione rimane comunque troppo elevata per essere gestita con risorse computazionali realistiche. In secondo luogo, il calcolo di una risposta ottimale non è un'operazione banale. Johanson et al. [44] hanno introdotto una tecnica di risposta ottimale accelerata specifica per il poker, in grado di ridurre drasticamente il costo computazionale: essa richiede circa 76 giorni di CPU ed è completamente parallelizzabile, portando il tempo effettivo ad appena un giorno. Sebbene tale contributo abbia reso praticabile un'operazione precedentemente ritenuta intrattabile, l'utilizzo di questa procedura all'interno di CFR-BR

risulterebbe comunque oneroso, poiché la risposta ottimale dovrebbe essere ricalcolata a ogni iterazione dell'algoritmo per garantire la convergenza verso una soglia desiderata.

Risultati. L'applicazione della tecnica CFR-BR al gioco HULHE permette di studiare in che modo la scelta della divisione dell'albero di gioco influenzi i requisiti di memoria e la velocità di convergenza dell'algoritmo. Nei risultati dell'Hold'em, è stato adottato un *trunk* a un round: in ciascuna iterazione vengono campionate le carte pubbliche rivelate all'inizio del secondo round, mentre il resto della mano viene esplorato come sotto-gioco [44].

Gli studi [44] mostrano che la memoria richiesta cresce rapidamente all'aumentare dei round campionati: mentre un campionamento a un round richiede pochi megabyte, uno a tre round arriva a superare i 350 GB. Per confronto, una soluzione CFR non astratta richiederebbe oltre 140 TB di RAM, rendendola di fatto impraticabile.

Gli esperimenti [44] mostrano che, in concomitanza all'aumentare delle iterazioni e alla scelta di uno specifico numero di bucket con cui creare l'astrazione, CFR-BR riesce a trovare la strategia ottimale.

Poiché l'algoritmo individua la strategia meno sfruttabile all'interno di un'astrazione, può sostituire CFR in questo compito misurando direttamente la capacità di un'astrazione di rappresentare una buona approssimazione a un equilibrio di Nash [44].

2.1.4 DDCFR

Contesto e descrizione algoritmo. L'approccio *Dynamic Discounted Counterfactual Regret Minimization* (DDCFR) nasce come estensione e perfezionamento degli algoritmi di CFR. Esso rappresenta il primo metodo di ricerca dell'equilibrio capace di applicare uno sconto alle iterazioni precedenti mediante uno schema dinamico appreso automaticamente. Tale meccanismo consente di ottenere una capacità di generalizzazione più elevata, una convergenza più rapida e, complessivamente, prestazioni migliorate rispetto alle varianti tradizionali di CFR [92], applicando lo sconto sia ai rimpianti sia alla strategia media per accelerare la convergenza. In questo caso, il peso assegnato alle iterazioni passate decresce in funzione di tre iper-parametri (α, β, γ) [92].

L'idea fondante del DDCFR consiste nell'incapsulare il processo iterativo del CFR all'interno di un ambiente, trattando lo schema di sconto come un agente che interagisce

con esso. In tale configurazione, l'agente riceve lo stato corrente del processo di iterazione e produce un'azione composta dai pesi di sconto da utilizzare nell'iterazione successiva. Il procedimento si ripete fino al raggiungimento del numero massimo di iterazioni. L'obiettivo dell'agente è apprendere una politica di sconto ottimale capace di selezionare dinamicamente pesi adeguati per ciascuna iterazione, minimizzando in tal modo la sfruttabilità delle strategie medie generate (si veda figura 2.1).

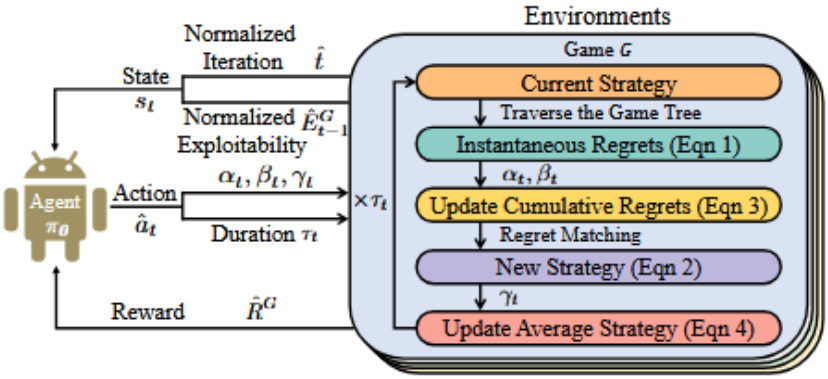

Figura 2.1: Funzionamento del ciclo iterativo di DDCFR [92].

Formalmente, l'interazione tra agente e ambiente in un dato gioco G definisce un processo decisionale di Markov (MDP), rappresentato come $(G, S, A, P_G, \hat{R}_G)$. Ogni gioco G costituisce un ambiente distinto, mentre lo stato $s_t \in S$ raccoglie le informazioni osservabili dall'agente all'iterazione t, includendo elementi quanto più generali e trasferibili possibile, così da permettere all'agente di prendere decisioni efficaci nella scelta dei pesi di sconto e di garantire al contempo la generalizzabilità dello schema appreso a giochi diversi.

A tal fine DDCFR utilizza uno spazio degli stati indipendente dal gioco, composto da due elementi: l'iterazione normalizzata $\hat{t}$, definita come il rapporto tra l'iterazione corrente t e il numero totale di iterazioni previste; e la sfruttabilità normalizzata $\hat{E}^G_{t-1}$, calcolata utilizzando E^G_1, E^G_{t-1} ed $E_{\min}$, rispettivamente la sfruttabilità delle strategie medie del gioco G all'iterazione 1 e all'iterazione $t-1$ ed il valore minimo raggiungibile di sfruttabilità, fissato pari a 10^{-12} [92].

A ogni iterazione t, l'agente osserva lo stato corrente s_t e produce un'azione $\hat{a}_t$. L'algoritmo applica quindi i pesi di sconto determinati da α_t, β_t e γ_t per un totale di τ_t iterazioni consecutive, al termine delle quali lo stato evoluto diviene $s_{t+\tau_t}$. Questo meccanismo permette di utilizzare ciascun insieme di pesi per una sequenza di iterazioni, anziché aggiornarli

continuamente. La funzione di ricompensa valuta le prestazioni dell'agente e guida l'apprendimento. Poiché l'obiettivo è ridurre la sfruttabilità, viene utilizzata una ricompensa sparsa, assegnata solo al termine dell'addestramento e basata sul miglioramento ottenuto tra l'inizio e la fine delle iterazioni. L'agente è modellato come una rete neurale parametrizzata da θ, che definisce una politica π_θ. Per ogni gioco G, l'obiettivo è massimizzare la ricompensa finale; considerando l'intero insieme dei giochi di addestramento, si massimizza la ricompensa media, ottenendo così una politica di sconto generalizzabile anche a giochi non visti [92].

Costo computazionale. Rispetto al DCFR, l'algoritmo DDCFR introduce alcuni costi computazionali aggiuntivi, riconducibili principalmente a tre componenti: il calcolo delle feature, l'inferenza della rete neurale e l'addestramento della politica di sconto. Tuttavia, tali costi aggiuntivi risultano marginali rispetto al tempo complessivo di esecuzione [92].

Il costo computazionale associato alla fase di addestramento è giustificato dal fatto che la politica di sconto appresa può essere riutilizzata direttamente in numerosi giochi differenti senza richiedere alcuna modifica. Di conseguenza, l'investimento computazionale sostenuto durante l'addestramento viene ammortizzato attraverso tutte le istanze in cui la politica viene applicata [92].

Risultati. Gli studi dimostrano come DDCFR addestrato su quattro giochi di piccola scala (Kuhn Poker, Goofspiel-3, Liar's Dice-3, Small Matrix) e testato su otto giochi complessi, inclusi varianti di Leduc Poker, Battleship, e sotto-giochi di Heads-Up No-Limit Texas Hold'em (HUNL) abbia portato a diversi risultati. Nei giochi di training, DDCFR converge molto più rapidamente, come atteso; nei giochi di test (mai visti durante l'addestramento), DDCFR dimostra una notevole capacità di generalizzazione [92].

Durante l'addestramento i parametri appresi mostrano un comportamento dinamico coerente tra i diversi giochi: α_t tende ad aumentare, mentre β_t e γ_t diminuiscono nel tempo, indicando un approccio più aggressivo nelle prime fasi e una maggiore stabilità nelle ultime [92].

2.1.5 Deep-CFR

Contesto e descrizione algoritmo. *Deep Counterfactual Regret Minimization*(Deep-CFR) viene introdotto come metodologia volta a superare i limiti dei processi di astrazione tipicamente utilizzati in algoritmi tabulari, con l'obiettivo principale di approssimare il comportamento del CFR classico senza dover calcolare e memorizzare esplicitamente i rimpianti in ciascun singolo insieme informativo. Tale risultato viene ottenuto generalizzando su stati simili mediante l'utilizzo dell'approssimazione funzionale basata su reti neurali profonde.

A ogni iterazione t, Deep CFR esegue un numero costante K di attraversamenti parziali dell'albero di gioco; ciascun attraversamento è guidato dal meccanismo di campionamento esterno (external sampling) MCCFR. In un qualsiasi insieme informativo I incontrato durante l'esplorazione, l'algoritmo adotta una strategia $\sigma_t(I)$ determinata dal regret-matching applicato all'output di una rete neurale $V : I \to \mathbb{R}^{|A|}$ parametrizzata da $\theta_p^{(t-1)}$. La rete prende in input l'insieme informativo I e restituisce in output valori $V(I, a \mid \theta_p^{(t-1)})$, i quali dovrebbero risultare approssimativamente proporzionali ai rimpianti $R^{(t-1)}(I, a)$ che il CFR tabulare avrebbe prodotto nel gioco completo [17].

Quando l'attraversamento raggiunge un nodo terminale, il valore di ritorno viene propagato verso l'alto. Negli stati di natura casuali (stati *chance*) e negli stati appartenenti all'avversario, il valore relativo all'azione campionata viene ritrasmesso senza modifiche.

Negli stati decisionali del giocatore in esame, invece, il valore propagato è la media ponderata dei valori delle possibili azioni, con pesi pari a $\sigma_t(I, a)$. Tale meccanismo produce campioni dei rimpianti istantanei per le diverse azioni nello stato considerato.

Questi campioni vengono memorizzati in una struttura dedicata $M_{v,p}$, distinta per ciascun giocatore p. Qualora la capacità della memoria venga superata, si utilizza la tecnica del *reservoir sampling* [85], che consente di mantenere un campione non distorto rispetto alla distribuzione originaria.

Completati i K attraversamenti di un giocatore, viene addestrata una nuova rete neurale, inizializzata casualmente, al fine di determinare i nuovi parametri $\theta_p^{(t)}$. L'addestramento mira a minimizzare l'errore quadratico medio (MSE) tra il vantaggio predetto $V_p(I, a \mid \theta^{(t)})$ e i campioni di rimpianti istantanei raccolti nelle iterazioni precedenti $t' \leq t$ e conservati nella memoria [17]. La media complessiva dei vantaggi istantanei campionati $\tilde{r}^{(t')}(I, a)$

risulta proporzionale al rimpianto totale campionato $\tilde{R}_t(I, a)$; pertanto ogni campione memorizzato contribuisce all'aggiornamento del modello anche nelle iterazioni successive, salvo la sostituzione determinata dal *reservoir sampling*.

Per la modellazione sia dei valori sia delle strategie è possibile utilizzare qualsiasi funzione di perdita appartenente alla classe delle divergenze di Bregman [4].

Accanto alla rete di valore, Deep CFR introduce una rete di policy separata $\Pi : I \to \mathbb{R}^{|A|}$, destinata ad approssimare la strategia media al termine dell'addestramento. Questo è motivato dal fatto che è la strategia media complessiva, aggregata su tutte le iterazioni, a convergere verso un equilibrio di Nash nei giochi a somma zero a informazione imperfetta.

Per ottenere tale approssimazione viene mantenuta una memoria specifica M_Π, dedicata ai vettori di probabilità sugli insiemi informativi campionati per entrambi i giocatori. Ogni volta che un insieme informativo I del giocatore p viene raggiunto durante l'attraversamento dell'albero condotto dall'avversario, il corrispondente vettore di probabilità $\sigma_t(I)$ viene inserito in M_Π e gli viene assegnato come peso il valore dell'iterazione corrente t.

Costo computazionale. L'architettura di rete neurale impiegata negli studi [17] presenta una profondità complessiva di sette livelli e comprende 98,948 parametri. Gli insiemi informativi sono costituiti dall'insieme delle carte private e pubbliche e dallo storico delle azioni di puntata, dove le carte vengono rappresentate come somma di tre distinti incapsulamenti applicati al rango (1–13), al seme (1–4) e all'identificativo specifico (1–52) (si veda figura 2.2).

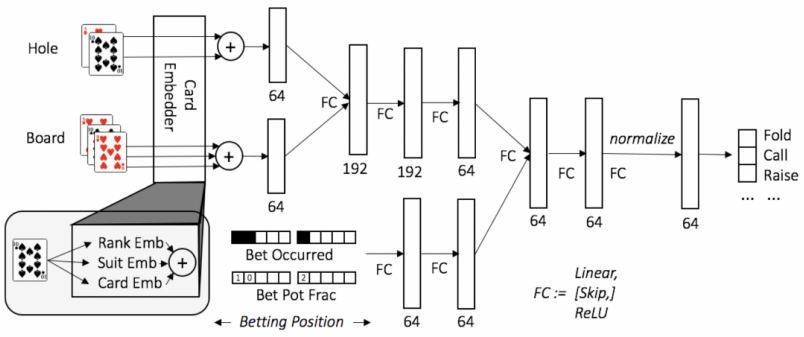

Figura 2.2: L'architettura della rete neurale utilizzata per Deep CFR. La rete accetta un insieme informativo (carte osservate e cronologia delle scommesse) come input e restituisce valori (vantaggi o logit di probabilità) per ogni possibile azione. [17].

Per quanto riguarda la gestione della memoria, è stata allocata una capacità massima pari a 40 milioni di insiemi informativi sia per la memoria dei vantaggi di ciascun giocatore $M_{V,p}$, sia per la memoria dedicata alla strategia M_{Π}. Il modello di valore è stato addestrato da zero a ogni iterazione CFR, partendo da una inizializzazione casuale dei pesi. L'addestramento ha previsto 4,000 iterazioni di *stochastic gradient descent* (SGD) con mini-batch di 10,000 campioni, utilizzando l'ottimizzatore Adam [46] con learning rate pari a 0,001 e applicando un meccanismo di clipping della norma del gradiente a 1 [17].

Nel caso del gioco Heads-Up Limit Hold'em (HULHE), gli studi indicano la necessità di un numero più elevato di aggiornamenti: vengono infatti eseguite 32,000 iterazioni di SGD con mini-batch di dimensione 20,000 [17].

Risultati. I risultati mostrano che Deep CFR risulta molto efficiente in termini di numero di nodi visitati, pur richiedendo un sovraccarico computazionale significativo dovuto all'inferenza e all'addestramento della rete neurale.

L'algoritmo è stato confrontato con NFSP [36], un altro metodo basato su reti neurali, nel contesto dell'Heads-Up Limit Hold'em (HULHE), utilizzando tre differenti astrazioni. I risultati di tale analisi confermando la maggiore efficacia di Deep CFR sia in termini di sfruttabilità sia in termini di qualità strategica complessiva [17].

2.1.6 ECFR

Contesto e descrizione algoritmo. Per affrontare le maggiori criticità di CFR, come il tempo di calcolo e la necessità di astrarre o generalizzare, viene proposto il metodo *Exponential CFR* (ECFR) [56], il cui obiettivo è accelerare la convergenza del CFR tradizionale e ottenere strategie robuste in maniera più efficiente. L'idea alla base dell'ECFR consiste nell'introdurre una tecnica di ponderazione esponenziale, volta ad attribuire un peso maggiore alle azioni caratterizzate da valori di rimpianto più elevati. Tale approccio consente all'algoritmo di concentrarsi maggiormente sulle azioni che generano rimpianto positivo, migliorando la qualità della strategia derivata. La funzione di ponderazione è definita come

$$f(x) = \begin{cases} e^{\alpha x}, & \text{se } x > 0, \\ e^{\alpha \beta}, & \text{se } x \leq 0, \end{cases}$$

dove α rappresenta un parametro che controlla la sensibilità della ponderazione al valore di x, β è un parametro di piccola entità e $f(x)$ costituisce l'output della funzione [56].

In particolare, la variabile x può assumere valori negativi durante il processo di risoluzione dei giochi. Diversamente dai metodi convenzionali, che annullano tali valori impostandoli a zero [12], l'ECFR assegna a tali variabili un nuovo valore minimo pari a $e^{\alpha\beta}$. Questa scelta è motivata dal fatto che, nelle fasi iniziali dell'addestramento, la strategia non è ancora sufficientemente accurata e alcune azioni con rimpianto negativo possono comunque essere rilevanti per l'aggiornamento strategico. Ignorare tali azioni nelle prime iterazioni risulterebbe irragionevole, mentre la loro inclusione consente una rappresentazione più completa dello spazio decisionale.

L'ECFR si fonda quindi sul CFR *vanilla* [93], ma integra la tecnica di ponderazione esponenziale per ridistribuire i pesi dei rimpianti istantanei. In questo contesto viene definita una funzione di perdita, dipendente dal parametro α, e il rimpianto istantaneo $r_t^i(I, a)$ in ciascuna iterazione assume il ruolo della variabile x nella funzione di ponderazione. Inoltre, il rimpianto istantaneo viene filtrato attraverso il valore medio EV, consentendo alla strategia dell'iterazione successiva di concentrarsi sulle azioni più vantaggiose, alle quali viene assegnato un peso proporzionalmente maggiore.

A ciascuna iterazione l'ECFR mira a minimizzare il rimpianto totale attraverso la riduzione del rimpianto su ciascun insieme informativo. Tuttavia, a differenza del CFR tradizionale, l'ECFR attribuisce un peso maggiore al rimpianto immediato. Con l'aumentare delle iterazioni, l'algoritmo concentra l'attenzione sulle azioni caratterizzate da rimpianto istantaneo più elevato, introducendo una perdita L_1 ponderata in forma esponenziale.

La strategia per l'iterazione $T+1$ può essere calcolata mediante un algoritmo di rimpianto (RM) come segue:

$$\sigma_{T+1}^i(I, a) = \frac{e^{L_1}\, R_{t,i}^{\mathrm{ECFR}}(I, a)}{\sum_{a' \in A(I)} e^{L_1}\, R_{t,i}^{\mathrm{ECFR}}(I, a')}.$$

Se il rimpianto medio di entrambi i giocatori soddisfa

$$\frac{R_T^i}{T} \leq \varepsilon,$$

allora la strategia media $\langle \bar{\sigma}_1^T, \bar{\sigma}_2^T \rangle$ costituisce un equilibrio di Nash a due giocatori in un gioco a somma zero [56].

Risultati. Sono stati condotti diversi esperimenti per valutare le prestazioni del metodo ECFR, con l'obiettivo di analizzarne l'efficacia su tre giochi di poker a due giocatori: Kuhn, Leduc e Royal poker. Tra questi, il Kuhn poker è il più semplice, composto da tre carte, un solo giro di puntate, una carta privata per giocatore e nessuna carta pubblica. Il Leduc poker utilizza sei carte e prevede due giri: nel primo ogni giocatore possiede una carta privata, mentre nel secondo viene introdotta una carta pubblica. Il Royal poker, infine, impiega otto carte e tre giri, con due carte pubbliche rivelate alla fine.

Le valutazioni sperimentali sono state effettuate confrontando quattro metodi: CFR [93], CFR+ [12], LCFR [17] e DCFR [16].

I risultati mostrano che ECFR presenta una dinamica complessiva simile agli altri metodi, con sfruttabilità decrescente al crescere delle iterazioni. Tali osservazioni confermano sperimentalmente la convergenza dell'ECFR, già dimostrata teoricamente [56].

Inoltre, nel khun poker ECFR offre prestazioni generalmente superiori rispetto agli altri metodi, sebbene in alcuni intervalli di iterazioni specifici DCFR ottenga valori leggermente migliori, similmente al Leduc poker. Infine, i risultati sul Royal poker indicano che ECFR risulta chiaramente superiore agli altri metodi, con una convergenza anticipata [56].

2.1.7 MCCFR

Contesto e descrizione algoritmo. *Monte Carlo Counterfactual Regret Minimization* (MCCFR) [51] rappresenta una variante dell'algoritmo CFR sviluppata con l'obiettivo di ridurre il tempo di attraversamento dell'albero di gioco per ogni iterazione, limitandolo a una porzione campionata dello stesso. Questo garantisce la convergenza e accelera l'avvicinamento all'equilibrio rispetto ai precedenti metodi di campionamento, pur preservando in aspettativa i rimpianti controfattuali immediati.

Sia $Q = \{Q_1, \ldots, Q_r\}$ un insieme di sottoinsiemi delle storie terminali Z; a ogni iterazione l'algoritmo campiona uno di questi blocchi e considera esclusivamente le storie terminali in esso contenute. Il valore controfattuale campionato rappresenta una stima non distorta del valore controfattuale effettivo [51]. Di conseguenza, MCCFR campiona un blocco e, per ogni insieme informativo che contiene un prefisso di una storia terminale del blocco, calcola i rimpianti controfattuali campionati di ciascuna azione. Questi rimpianti vengono

accumulati nel tempo e la strategia alla successiva iterazione viene determinata applicando la regola di *regret-matching* [51].

Esistono diverse modalità per campionare porzioni dell'albero di gioco. La procedura più semplice è rappresentata dall'*outcome sampling* (OS), in cui a ogni iterazione viene campionata una singola storia terminale e l'aggiornamento dei rimpianti avviene esclusivamente sugli insiemi di informazioni attraversati lungo tale storia [51].

Un secondo metodo di campionamento è lo *chance sampling* (CS), in cui l'insieme delle storie terminali viene suddiviso in blocchi tali che due storie non possono appartenere allo stesso blocco a meno che non differiscano unicamente per le azioni di natura probabilistica. Nel CS un blocco viene generato campionando una singola azione casuale per ogni storia, secondo la probabilità associata a tale azione. Le strategie vengono successivamente aggiornate mediante i rimpianti cumulativi [32].

Infine, l'*Average Strategy Sampling* (AS) seleziona le azioni del giocatore i in base al profilo cumulativo e a tre parametri predefiniti. L'AS può essere interpretato come uno schema di campionamento intermedio tra OS ed ES: per ciascun insieme informativo I, viene campionato un sottoinsieme delle azioni disponibili, anziché una singola azione (OS) o tutte le azioni (ES). Come nell'ES, nei nodi dell'avversario e nei nodi di natura probabilistica viene campionata una sola azione, rispettivamente secondo la politica del profilo avversario corrente $\sigma_{T,-i}$ e le probabilità fissate σ_c [32].

Costo computazionale. È stato dimostrato che il *chance sampling* (CS) riduce significativamente i tempi computazionali in giochi complessi come il poker [93]. Oltre al CS, altre varianti quali *Average Strategy Sampling* (AS), *External Sampling* (ES) ed *Outcome Sampling* (OS) [32], convergono verso l'equilibrio più rapidamente del CFR classico in diversi domini, fornendo inoltre limiti probabilistici sul rimpianto medio e quindi garanzie sulla convergenza della strategia media $\bar{\sigma}^T$ a un equilibrio di Nash.

Per quanto riguarda le risorse necessarie, l'algoritmo MCCFR richiede di memorizzare apposite tabelle per ciascun insieme informativo; ogni tabella contiene un numero di voci pari alle azioni disponibili nell'insieme corrispondente. Indicando con $|A_i|$ il numero massimo di azioni disponibili al giocatore i in tutti i suoi insiemi informativi, il requisito di spazio dell'algoritmo risulta pari a $O\big(|I_1|\,|A_1| + |I_2|\,|A_2|\big)$.

Il tempo computazionale richiesto da MCCFR, nel caso in cui venga adottato OS, dipende dai limiti sul rimpianto e dal livello di approssimazione dell'equilibrio che si desidera

ottenere. Per raggiungere un ε-equilibrio di Nash con probabilità almeno $1 - p$, il numero di iterazioni necessario è dell'ordine $O\left(\frac{2}{p\,\delta^2}\,|A|\,M^2\right)$, dove:

- δ rappresenta la minima probabilità di campionare una storia terminale tra tutte quelle possibili;
- $|A|$ è il numero massimo di azioni disponibili in qualunque insieme di informazioni;
- M è un fattore di bilanciamento che riflette il numero relativo di decisioni assunte dai giocatori durante l'intera partita e soddisfa $\sqrt{|I|} \;\leq\; M \;\leq\; |I|$.

Risultati. Uno studio recente [32] ha condotto una serie di esperimenti nel dominio dell'HUNLHE, impiegando un'astrazione a cinque carte. Per ciascuna partita sono state eseguite cinque istanze degli algoritmi CS, ES, OS e AS, misurando la sfruttabilità del gioco astratto in diversi checkpoint e calcolando successivamente la media dei risultati ottenuti. I dati sperimentali mostrano che l'algoritmo AS ha conseguito un miglioramento pari al 54% rispetto a ES nei punti di misura finali, mentre OS ha evidenziato prestazioni significativamente inferiori.

Risultati analoghi sono stati riscontrati per i giochi Bluff$(1, 1)$ e Bluff$(2, 1)$, che presentano rispettivamente oltre 24 k e 3.5 M insiemi informativi, e circa 294 k e 66 M cronologie. Anche in questo contesto, AS ha mostrato una velocità di convergenza superiore rispetto a CS, ES e OS in entrambe le istanze di Bluff analizzate.

2.1.8 Regression CFR

Contesto e descrizione algoritmo. Lo sviluppo di *Regression Counterfactual Regret Minimization* (RCFR) [27] nasce dall'esigenza di rendere più efficiente la risoluzione di giochi sequenziali con informazione imperfetta, in cui il numero di stati è così elevato da rendere impraticabile l'approccio tabellare del CFR tradizionale. Nello studio condotto dall'Università di Alberta [27] si evidenzia inoltre come l'uso dell'approssimazione di funzione costituisca una naturale estensione delle tecniche di astrazione, permettendo di stimare i rimpianti controfattuali invece di memorizzarli integralmente, mantenendo al contempo prestazioni competitive con le metodologie basate su astrazioni esplicite.

L'algoritmo RCFR si basa sull'idea di sostituire la rappresentazione tabellare dei rimpianti con una stima funzionale ottenuta tramite un approssimatore. L'approccio RCFR mantiene la struttura di minimizzazione del rimpianto, ma invece di memorizzare esattamente i rimpianti cumulativi ne apprende una stima tramite un modello, tipicamente una funzione parametrica che sfrutta caratteristiche condivise tra diversi stati del gioco. Ciò consente di trasferire informazione tra stati simili, riducendo drasticamente la necessità di memorizzazione e rendendo il processo più scalabile in giochi di grande dimensione [91].

Nel funzionamento di RCFR, l'apprendimento del comportamento dell'agente avviene in due fasi principali. La prima consiste nella predizione dei rimpianti cumulativi tramite un approssimatore $y(\phi(s, a))$, dove $\phi(s, a)$ rappresenta una codifica vettoriale dell'insieme informativo e dell'azione corrispondente.

Successivamente, viene costruita la politica a partire dai rimpianti predetti, applicando una trasformazione determinata da una *link function* f [27]. Nel caso originale di RCFR, la link function è una versione normalizzata della ReLU, coerente con il tradizionale *regret matching* [35], che assegna probabilità soltanto alle azioni con rimpianto positivo e in misura proporzionale alla loro entità. L'algoritmo produce quindi una distribuzione stocastica su ciascuno insieme informativo normalizzando i valori ottenuti dopo l'applicazione della ReLU.

Costo computazionale. Nel contesto dell'algoritmo RCFR e della sua generalizzazione f-RCFR, lo studio di Bowling et al. [27] evidenzia come l'impiego di un approssimatore funzionale comporti effetti specifici sul consumo di tempo ed energia computazionale. L'adozione di una rappresentazione lineare basata su partizioni implica che, per ogni insieme informativo, solo una feature per partizione risulti diversa da zero; tale struttura determina che il costo di predizione cresca linearmente con il numero di partizioni considerate durante l'apprendimento.

Parallelamente, l'aggiornamento del modello presenta una complessità ben più elevata: poiché l'algoritmo opera su un insieme di feature complessivamente più ampio, il costo della fase di aggiornamento cresce quadraticamente rispetto al numero totale di feature impiegate nell'approssimazione dei rimpianti [27].

Dal punto di vista della memoria, RCFR introduce un vantaggio strutturale rispetto ai metodi tabellari tradizionali, in quanto non richiede la memorizzazione persistente dei rimpianti cumulativi: una volta aggiornati i pesi del modello, i rimpianti delle iterazioni precedenti non devono essere conservati né rielaborati. Questo riduce in modo significativo

il fabbisogno di memoria rispetto al CFR classico, che necessita invece di mantenere un vettore di rimpianti per ciascun insieme informativo del gioco.

Risultati. I risultati sperimentali mostrano in modo sistematico come la scelta della *link function* e il livello di approssimazione influenzino le prestazioni dell'algoritmo f-RCFR nei diversi domini considerati. Gli esperimenti condotti in Leduc Hold'em, goofspiel e random goofspiel valutano la qualità delle strategie tramite la sfruttabilità, misurata lungo 100,000 iterazioni. Le analisi indicano che l'aumento del numero di partizioni riduce l'errore di approssimazione e conduce, in accordo con le previsioni teoriche, a strategie mediamente meno sfruttabili [27].

In condizioni di approssimazione molto accurata, invece, la softmax tende a non superare le alternative polinomiali, confermando che la scelta della parametrizzazione ottimale dipende fortemente dal livello di rumore introdotto dal modello di regressione. Complessivamente, i risultati sperimentali confermano la validità delle analisi teoriche proposte e dimostrano che f-RCFR è in grado di adattarsi in modo flessibile ai diversi livelli di approssimazione del rimpianto e offrendo prestazioni robuste nei giochi sequenziali con informazione imperfetta [27].

2.1.9 SD-CFR

Contesto e descrizione algoritmo. *Single Deep Counterfactual Regret Minimization* (SD-CFR) [79] rappresenta una variante semplificata di Deep CFR sviluppata con l'obiettivo di ridurre l'errore di approssimazione e rendere più efficiente il processo di addestramento nei giochi a informazione imperfetta di grandi dimensioni.

L'introduzione di modelli di approssimazione tramite reti neurali, come DeepStack e successivamente Deep CFR, ha permesso di superare le restrizioni dei metodi tabellari, rendendo possibile la generalizzazione anche in stati mai osservati. SD-CFR si colloca direttamente in questa linea evolutiva, migliorando l'efficienza degli approcci basati su Deep CFR e dimostrando sperimentalmente una convergenza più rapida e prestazioni superiori nelle partite uno contro uno [79].

SD-CFR mira a ridurre l'errore di approssimazione eliminando la necessità di addestrare una rete neurale dedicata alla strategia media. Nel funzionamento classico di Deep CFR,

ogni iterazione produce una *value network* che approssima il vantaggio di ciascuna azione, valore ricavato a partire dal rimpianto lineare e normalizzato tramite la *reach probability* dell'avversario [17].

SD-CFR mantiene la stessa procedura di apprendimento delle value networks, ma differisce nel modo in cui ricostruisce la strategia media. Invece di addestrare una seconda rete per approssimare la media pesata delle strategie, l'algoritmo conserva tutte le value networks generate nelle iterazioni precedenti e utilizza direttamente tali modelli per ricavare la politica media, riducendo così il numero di approssimazioni e semplificando il processo di addestramento.

Durante l'esecuzione, SD-CFR può operare in due modalità: *trajectory sampling*, in cui si seleziona una rete riferita a una delle iterazioni passate con probabilità proporzionale al peso lineare dell'iterazione e la si utilizza per l'intera traiettoria; oppure una modalità di *calcolo esplicito*, in cui la strategia media viene ottenuta computando direttamente le *reach probabilities* di ciascuna rete e applicando la formula della strategia media di CFR [79]. In entrambi i casi, l'algoritmo garantisce una ricostruzione esatta della strategia media di CFR qualora i value networks approssimino perfettamente i valori di vantaggio.

Costo computazionale. SD-CFR replica accuratamente la strategia media a partire dalle strategie di iterazione fornite durante l'addestramento. Ne consegue che, qualora tali strategie di iterazione costituissero approssimazioni perfette delle strategie effettive generate da CFR, SD-CFR risulterebbe equivalente al CFR lineare.

Come mostrato sperimentalmente [79], le prestazioni di SD-CFR tendono a deteriorarsi quando il numero di iterazioni supera la capacità del buffer destinato alla memorizzazione. Fortunatamente, la rete neurale impiegata in Deep CFR per le partite di poker di grandi dimensioni presenta una dimensione estremamente contenuta, inferiore a 100,000 parametri [17], corrispondenti a meno di 400 KB di spazio su disco. Considerando che Deep CFR viene tipicamente addestrato per alcune centinaia di iterazioni, la memorizzazione di 25,000 reti di questo tipo richiederebbe circa 10 GB di spazio, una quantità pienamente gestibile nei contesti computazionali moderni. Inoltre, nessun passaggio dell'algoritmo richiede di mantenere simultaneamente in memoria tutte le reti archiviate, eliminando così qualsiasi criticità pratica legata alla gestione delle risorse.

Risultati. Gli esperimenti condotti [79] evidenziano che SD-CFR presenta prestazioni complessivamente superiori rispetto a Deep CFR in termini di sfruttabilità e di qualità della strategia appresa. Nella variante Leduc Hold'em Poker, SD-CFR mostra una riduzione più rapida dell'sfruttabilità rispetto a Deep CFR, nonostante l'impostazione degli iper-parametri favorisca quest'ultimo, indicando una maggiore stabilità del processo di apprendimento.

Inoltre, l'analisi delle differenze tra le strategie medie prodotte dai due metodi rivela inoltre che Deep CFR tende a introdurre errori più significativi negli insiemi informativi raggiunti nelle fasi avanzate del gioco, mentre SD-CFR mantiene una coerenza strategica più elevata anche a profondità maggiori dell'albero decisionale, tendenza confermata [79] nel contesto del 5-Flop Hold'em Poker..

L'analisi dell'effetto del *reservoir sampling* sul buffer BM mostra tuttavia che l'uso di capacità limitate può causare fenomeni di plateau e oscillazioni nella convergenza, confermando l'importanza di preservare tutte le value networks generate durante l'addestramento per mantenere l'affidabilità dell'approssimazione.

2.1.10 TCFR

Contesto e descrizione algoritmo. *Targeted Counterfactual Regret Minimization* (TCFR) nasce dall'esigenza di individuare un punto di equilibrio tra i principali metodi di campionamento utilizzati nel CFR, in particolare *Outcome Sampling* ed *External Sampling* [51]. Outcome Sampling è molto veloce ma fornisce stime ad alta varianza e di bassa accuratezza, mentre External Sampling esplora ampie porzioni dell'albero a ogni iterazione, risultando molto più costoso dal punto di vista computazionale.

L'idea alla base di Targeted CFR è quindi quella di definire un approccio intermedio che, in un singolo passaggio, visiti più nodi rispetto all'Outcome Sampling ma meno rispetto all'External Sampling, mantenendo così un buon compromesso tra costo computazionale e precisione della stima. Inoltre, gli studi [39] evidenziano come, nei giochi di poker, le fasi avanzate (gli ultimi *betting rounds*) siano visitate molto raramente dagli algoritmi di sampling tradizionali, nonostante rappresentino una parte rilevante della complessità del gioco. TCFR è progettato proprio per compensare questa asimmetria, concentrando l'esplorazione in modo più frequente su tali porzioni "critiche" dell'albero di gioco.

Il funzionamento dell'algoritmo si basa sulla suddivisione preliminare dell'albero di gioco in un insieme di partizioni, ossia in regioni distinte sulle quali è possibile concentrare selettivamente l'attività di campionamento [39].

A ogni iterazione, una o più partizioni vengono designate come *mirate*. All'interno di tali aree l'algoritmo opera in modo analogo all'External Sampling: vengono considerate tutte le azioni disponibili e vengono aggiornati sia i rimpianti sia il profilo cumulativo. Al di fuori delle partizioni mirate, invece, il comportamento cambia in modo significativo: l'algoritmo esegue una *sonda*, ovvero segue una singola traiettoria dall'informazione corrente fino a uno stato terminale, campionando una sola azione per ciascun giocatore secondo la strategia corrente. In queste regioni non vengono effettuati aggiornamenti, ma si ottiene comunque una stima imparziale del valore controfattuale, utile alla successiva fase di calcolo delle funzioni di valore.

La progettazione delle partizioni e la scelta della frequenza con cui ciascuna viene targetizzata rappresentano una decisione a discrezione dell'implementatore. Nel caso del *Texas Hold'em*, questa operazione risulta particolarmente naturale, poiché la struttura del gioco prevede quattro distinti round di puntate, ciascuno dei quali può essere trattato come una partizione. In base alle esigenze specifiche, è possibile mirare un singolo round oppure una combinazione di round.

Per soddisfare il requisito di raggiungibilità — fondamentale affinché le stime controfattuali restino valide — il *Targeted CFR* prevede inoltre l'inserimento di alcune iterazioni "complete", durante le quali tutte le partizioni vengono mirate simultaneamente. Queste iterazioni coincidono esattamente con quelle dell' *External Sampling* e, poiché quest'ultimo rispetta il vincolo di raggiungibilità, anche il *Targeted CFR* lo soddisfa, a patto che tali iterazioni vengano eseguite con probabilità non nulla [39].

La frequenza con cui ciascuna combinazione di partizioni viene scelta è regolata da un parametro τ [39], che definisce la distribuzione di probabilità utilizzata per determinare la selezione delle aree target a ogni iterazione.

Risultati. La valutazione sperimentale di TCFR [39] è stata condotta su diverse varianti di giochi di poker, con l'obiettivo di confrontarlo principalmente con *External Sampling* e, in alcuni casi, con *Average Strategy Sampling*. Gli esperimenti hanno considerato giochi di dimensioni crescenti e con differenti livelli di astrazione delle carte, misurando le prestazioni secondo due indicatori complementari: la sfruttabilità e la performance "head-to-head",

determinata dal confronto diretto con una strategia di riferimento.

Nel primo scenario analizzato [39], caratterizzato da un gioco relativamente contenuto e privo di astrazione delle carte, TCFR ha mostrato risultati molto simili a quelli di *External Sampling* in termini di sfruttabilità, con un lieve vantaggio nel confronto diretto che tende tuttavia a ridursi nel tempo. Le differenze diventano più evidenti nei giochi basati su astrazioni con richiamo imperfetto: nel secondo esperimento, condotto con un mazzo completo e una complessa astrazione in circa un milione di bucket, TCFR ha ottenuto prestazioni sensibilmente migliori nelle partite dirette e una sfruttabilità reale più bassa, nonostante entrambe le tecniche mostrino una tendenza al peggioramento nelle ultime fasi dell'addestramento, fenomeno tipico delle astrazioni imperfette.

Il terzo esperimento [39], basato su un sistema di puntate molto più esteso e su un' astrazione ridotta delle carte, ha ulteriormente confermato il vantaggio del metodo: TCFR supera nettamente sia *External Sampling* sia *Average Strategy Sampling* nelle prestazioni head-to-head, evidenziando una maggiore capacità di adattarsi alle peculiarità dei grandi giochi con richiamo imperfetto. Nel complesso, i risultati mostrano che TCFR non offre benefici rilevanti nei giochi risolvibili senza astrazione, mentre si dimostra particolarmente efficace e competitivo nei contesti più complessi, in cui la struttura dell'albero di gioco e le limitazioni dell'astrazione rendono critica un'esplorazione più mirata delle informazioni rilevanti.

2.2 Fictitious Self-Play

La categoria degli approcci *Fictitious Self-Play* si fonda sul concetto di *gioco fittizio* (fictitious play), introdotto per la prima volta da Brown (1951) [15], e che costituisce uno dei modelli di apprendimento più noti nell'ambito della teoria dei giochi. In questo schema, gli individui coinvolti ripetono la stessa interazione strategica scegliendo, a ogni iterazione, la risposta ottimale alle strategie medie osservate nei loro avversari. In specifiche classi di giochi — tra cui i giochi a somma zero e i giochi a due giocatori — il profilo strategico medio generato tramite gioco fittizio converge a un equilibrio di Nash [37].

Nella famiglia di algoritmi FSP questo concetto viene ampliato mediante l'introduzione di meccanismi di approssimazione e apprendimento che permettono di estendere il gioco fittizio a domini di grandi dimensioni e ad informazione imperfetta. Tali algoritmi mantengono una stima delle strategie medie degli avversari e ne calcolano iterativamente una risposta ottimale, superando i limiti computazionali del modello originario. Varianti moderne, come *NFSP*, integrano tecniche di apprendimento supervisionato e rinforzato per rappresentare e aggiornare le strategie in modo efficiente, rendendo l'approccio applicabile a giochi complessi quali il Texas Hold'em Heads-Up Limit.

2.2.1 FSP

Contesto e descrizione algoritmo. Il modello del gioco fittizio è divenuto uno strumento consolidato nella letteratura teorica e ha stimolato un ampio dibattito su come gli equilibri di Nash possano manifestarsi nella pratica [31, 38, 55]. Inoltre, esso rappresenta un classico esempio di apprendimento dall'esperienza applicato al contesto decisionale, che ha influenzato in modo significativo lo sviluppo di algoritmi di intelligenza artificiale orientati ai giochi.

Nonostante la sua diffusione teorica, però, il gioco fittizio è stato applicato relativamente poco su larga scala [50, 59]. Una delle principali ragioni risiede nel fatto che esso richiede una rappresentazione del gioco in forma normale. Sebbene qualsiasi gioco in forma estesa possa essere convertito nel suo equivalente in forma normale, il numero di azioni risultante cresce tipicamente in modo esponenziale rispetto al numero degli stati del gioco. La forma estesa permette invece una descrizione molto più compatta grazie all'impiego di strategie

comportamentali, il cui numero di parametri aumenta in modo lineare rispetto agli stati informativi.

Nel tentativo di superare tali limitazioni, è stata introdotta la variante *Fictitious Self-Play* (FSP) [37], che propone un quadro di apprendimento automatico che implementa una versione generalizzata e indebolita del gioco fittizio, basata sulle strategie comportamentali e su campionamenti dell'esperienza di gioco. In questo modello, gli individui interagiscono ripetutamente e memorizzano gli episodi generati, impiegando strategie caute che combinano le risposte ottimali con le strategie medie accumulate. FSP campiona iterativamente episodi derivanti dal gioco individuale; tali episodi costituiscono i set di dati necessari alla stima delle risposte ottimali e dei modelli perturbati delle strategie medie.

Si consideri un gioco in forma estesa e un profilo strategico π. Per ciascun giocatore $i \in N$, il profilo strategico degli avversari π_{-i} definisce un processo decisionale di Markov (MDP), indicato come $M(\pi_{-i})$ [34, 78]. Gli insiemi informativi del giocatore i costituiscono gli stati dell'MDP, mentre le dinamiche dello stesso sono determinate dalle regole del gioco in forma estesa, dalla funzione casuale e dal profilo strategico fissato per gli avversari. Le ricompense derivano direttamente dalla funzione di payoff del gioco. Una politica ottimale per l'MDP $M(\pi_{-i})$ rappresenta pertanto una risposta ottimale approssimata del giocatore i rispetto al profilo π_{-i}.

Ne consegue che il calcolo iterativo delle risposte ottimali può essere formulato come la risoluzione approssimata di una sequenza di MDP, ottenuta ad esempio applicando algoritmi di apprendimento per rinforzo ai campioni di esperienza prelevati dai rispettivi MDP. Più precisamente, per risolvere approssimativamente $M(\pi_{-i})$, l'esperienza del giocatore i viene campionata a partire dal comportamento strategico dei suoi avversari. La strategia del giocatore deve garantire una sufficiente esplorazione dello spazio degli stati, ma può essere altrimenti arbitraria qualora si impieghi un metodo off-policy, come il Q-learning [88].

Successivamente, avviene l'aggiornamento della strategia media: ogni agente aggiorna la propria strategia media attraverso apprendimento supervisionato, utilizzando i dati memorizzati sul proprio comportamento.

Affinché il processo riproduca correttamente la logica del gioco fittizio, entrambe le operazioni di apprendimento devono essere supportate da dati campionati a partire da combinazioni specifiche di strategie. A tal fine, viene impiegato un profilo strategico di campionamento definito come $\sigma_k = (1 - \eta_k)\,\pi_{k-1} + \eta_k\,\beta_k$, dove π_{k-1} rappresenta la strategia

media all'iterazione precedente e β_k la risposta ottimale approssimata dell'iterazione corrente. Il parametro η_k regola la combinazione dei due profili; ad esempio, ponendo $\eta_k = 1/k$, il profilo σ_k coincide con la strategia media π_k corrispondente a un processo di gioco fittizio con passo $\alpha_k = 1/k$ [37].

Costo computazionale. FSP adotta un approccio basato sul campionamento che consente di contenere significativamente i costi computazionali rispetto alle altre varianti del gioco fittizio. A differenza dei metodi che richiedono di elaborare l'intero spazio degli stati a ogni iterazione, FSP concentra l'elaborazione esclusivamente sugli insiemi informativi effettivamente visitati durante la simulazione, riducendo quindi l'impatto della crescita dimensionale del gioco. Ciò permette all'algoritmo di operare in uno spazio che cresce linearmente con il numero degli insiemi informativi e non con il numero totale degli stati del gioco, che può essere esponenziale. Il costo computazionale per iterazione dipende essenzialmente dal numero di episodi campionati e gestiti nella memoria di apprendimento degli agenti [37].

Dal punto di vista della convergenza, FSP soddisfa le condizioni strutturali richieste dai processi di *fictitious play* generalizzati, poiché le risposte ottimali approssimate ottenute tramite apprendimento per rinforzo costituiscono una sequenza di politiche il cui errore ammissibile tende a ridursi nel tempo, mentre gli aggiornamenti della strategia media possono essere interpretati come stime perturbate ma coerenti del processo teorico corrispondente. La convergenza asintotica delle componenti apprese verso gli aggiornamenti corretti è garantita dal fatto che sia l'errore sulle risposte ottimali sia le perturbazioni introdotte negli aggiornamenti decrescono progressivamente. Sebbene l'analisi teorica riguardi prevalentemente il caso con risorse computazionali non limitate, e quindi resti aperta la questione della convergenza garantita sotto un budget computazionale finito per iterazione, lo studio mostra che FSP soddisfa i requisiti formali del processo di *fictitious play* indebolito e fornisce evidenza empirica di un comportamento convergente verso strategie prossime a un equilibrio di Nash [37].

Risultati. Nel gioco Leduc Hold'em a 6 carte, FSP mostra prestazioni inferiori rispetto al confronto con la variante senza apprendimento per rinforzo XFP quando entrambi possono utilizzare un budget computazionale fisso [37]. In questo contesto di dimensioni ridotte, XFP risulta più efficace nel ridurre rapidamente la sfruttabilità, ma aumentando la dimensionalità del gioco, emerge chiaramente la differenza di scalabilità tra i due algoritmi: Nel

Leduc Hold'em a 60 carte, dove il numero di stati cresce quadraticamente con il numero di carte, XFP subisce un marcato rallentamento, mentre FSP mantiene un ritmo di apprendimento stabile. FSP, guidato dal campionamento, focalizza i calcoli sui percorsi di gioco più probabili e beneficia fortemente della struttura introdotta dalle credenze dei giocatori, ottenendo nella versione informata un miglioramento della sfruttabilità superiore al 40%.

Complessivamente, i risultati sperimentali evidenziano che FSP, pur non essendo sempre competitivo nei giochi di dimensioni ridotte, scala meglio in giochi di maggiori dimensioni e sfrutta in modo efficace la struttura statistica del problema tramite campionamento.

2.2.2 NFSP

Contesto e descrizione algoritmo. *Neural Fictitious Self-Play* (NFSP) [36] ha l'obiettivo di affrontare in modo scalabile e autonomo i giochi a informazione imperfetta di dimensioni reali, un ambito in cui i metodi di apprendimento falliscono nel convergere, e gli algoritmi che possono calcolare equilibri di Nash dipendono pesantemente da astrazioni del dominio, spesso basate su euristiche o modellazione esplicita.

Per superare queste problematiche, NFSP combina l'algoritmo del *Fictitious Self-Play* (FSP) con l'approssimazione funzionale fornita dalle reti neurali. In questo modello, ogni giocatore è rappresentato da un agente NFSP distinto, il quale apprende attraverso interazioni in self-play, ossia mediante partite disputate simultaneamente contro altri agenti analoghi. Durante tali interazioni, ciascun agente registra sia l'esperienza relativa alle transizioni del gioco, sia le proprie azioni derivanti dalle risposte ottimali approssimate. Questi dati vengono archiviati in due memorie separate, indicate rispettivamente come $\mathcal{M}^{\mathrm{RL}}$ e $\mathcal{M}^{\mathrm{SL}}$, che vengono trattate come insiemi di dati indipendenti: la prima destinata all'apprendimento per rinforzo, la seconda all'apprendimento supervisionato. L'agente utilizza i dati contenuti in $\mathcal{M}^{\mathrm{RL}}$ per addestrare una rete neurale $Q(s, a \mid \theta_Q)$, la quale stima i valori delle azioni tramite apprendimento per rinforzo off-policy. La politica risultante è una strategia di risposta ottimale approssimata, definita come $\beta = \varepsilon\text{-greedy}(Q)$, che seleziona un'azione casuale con probabilità ε, altrimenti sceglie l'azione con valore stimato massimo.

Parallelamente, una seconda rete neurale, $\Pi(s, a \mid \theta_\Pi)$, viene addestrata sui dati in $\mathcal{M}^{\mathrm{SL}}$ mediante classificazione supervisionata, al fine di imitare il comportamento dell'agente nelle

passate risposte ottimali. Tale rete associa a ciascuno stato una distribuzione di probabilità sulle azioni e definisce la strategia media dell'agente, indicata con $\pi = \Pi$.

Nel corso del gioco, l'agente seleziona le proprie azioni mediante una combinazione delle due strategie, β e π, realizzando così un equilibrio tra esplorazione e sfruttamento dell'esperienza storica. NFSP integra inoltre due elementi tecnici cruciali per garantire la stabilità dell'apprendimento. In primo luogo, impiega il *reservoir sampling* [85], che consente di evitare alterazioni dovute al campionamento da memorie di dimensione limitata. In secondo luogo, utilizza dinamiche anticipatorie [77], le quali permettono agli agenti di campionare in misura controllata il proprio comportamento di risposta ottimale e di seguire in modo più efficace l'evoluzione nel tempo delle strategie avversarie.

Costo computazionale. Lo studio principale [36] evidenzia che i costi computazionali di NFSP derivano principalmente dall'uso delle due memorie di apprendimento di dimensione finita e dall'addestramento iterativo delle due reti neurali mediante aggiornamenti stocastici. Le memorie $\mathcal{M}^{\mathrm{RL}}$ e $\mathcal{M}^{\mathrm{SL}}$ sono implementate rispettivamente come *circular buffer* e *reservoir sampling*, soluzioni che mantengono i requisiti di memoria entro limiti controllati indipendentemente dal numero di iterazioni, contribuendo così alla scalabilità dell'algoritmo. Negli esperimenti, la dimensione di tali memorie raggiunge valori significativi: fino a fino a 600,000 e 30 milioni in Limit Texas Hold'em, indicando che la qualità dell'apprendimento dipende anche dalla disponibilità di ampi buffer di esperienza.

L'addestramento comporta inoltre numerosi aggiornamenti dei parametri delle reti neurali, eseguiti con frequenze calibrate rispetto al numero di passi di gioco — ad esempio due aggiornamenti per rete ogni 128 o 256 interazioni. Il costo computazionale cresce dunque linearmente con il numero di iterazioni e con la dimensione delle reti utilizzate, come mostrato dai risultati sperimentali in cui architetture neurali più grandi permettono prestazioni migliori, ma implicano naturalmente un aumento del carico computazionale.

Lo studio [36] sottolinea anche che NFSP trae un vantaggio computazionale significativo dal suo approccio basato sul campionamento, evitando l'esplorazione esplicita dell'intero spazio degli stati. Ciò gli consente di operare in domini di grandi dimensioni, senza la necessità di enumerare tutte le informazioni di gioco. Tuttavia, la stabilità e l'accuratezza dell'algoritmo richiedono l'uso di grandi memorie, frequenti aggiornamenti delle reti e procedure di campionamento ben calibrate, rendendo l'algoritmo più oneroso rispetto ai metodi tabellari, ma significativamente più scalabile rispetto alle procedure esaustive.

Risultati. NFSP è stato testato su Leduc Hold'em, in cui riduce progressivamente la sfruttabilità, soprattutto con architetture neurali più ampie e, grazie alla miscela controllata tra strategia media e risposta ottimale, produce dati di addestramento stabili e diversificati.

L'algoritmo è stato inoltre applicato a Limit Texas Hold'em, dove ha appreso una strategia competitiva rispetto ai migliori agenti della *Annual Computer Poker Competition* 2014 [36], ottenendo prestazioni comparabili alla metà superiore degli algoritmi pur senza utilizzare alcuna astrazione realizzata a mano.

Nel complesso, gli esperimenti dimostrano che NFSP è in grado di combinare stabilità, scalabilità e assenza di conoscenza di dominio, realizzando un apprendimento efficace.

2.3 Metodi basati su Programmazione Lineare e Risposte Ottimali

Gli algoritmi basati su oracoli e risposte ottimali rappresentano un'evoluzione dei metodi classici di risoluzione dei giochi in forma estesa, nati dall'esigenza di superare i limiti computazionali associati alla conversione in forma normale e alla successiva soluzione tramite Programmi Lineari. Sebbene l'approccio originario basato su LP abbia costituito il primo metodo generale per la risoluzione di giochi ad informazione imperfetta, la crescita esponenziale delle strategie deterministiche ha rapidamente reso tale tecnica impraticabile per domini anche moderatamente complessi. L'introduzione della rappresentazione in forma sequenza e del *Sequence-Form Linear Program* [47, 49, 69] ha permesso di affrontare giochi di grande scala in tempo polinomiale, aprendo la strada a metodi più avanzati capaci di integrare apprendimento, ottimizzazione e generazione incrementale di strategie. In questo contesto si collocano approcci moderni come PSRO, ODO, XDO, NXDO e RMDO, i quali si basano su procedure iterative che alternano la ricerca di risposte ottimali tramite oracoli e l'aggiornamento di un meta-gioco approssimato. Questi algoritmi consentono di esplorare in modo selettivo lo spazio strategico, ampliando progressivamente il supporto delle strategie rilevanti e mantenendo un compromesso favorevole tra accuratezza e complessità computazionale.

2.3.1 ODO

Contesto e descrizione algoritmo. *Online Double Oracle* (ODO) [26] è un algoritmo che si fonda sugli approcci *Double Oracle* (DO) [60], i quali costituiscono metodi particolarmente efficaci per l'individuazione di un Equilibrio di Nash (NE) approssimato in giochi caratterizzati da un supporto dell'equilibrio relativamente contenuto. Nella procedura del DO, ciascun giocatore viene inizializzato con un insieme limitato di strategie, che consente di operare su un sotto-gioco del gioco originario. A ogni iterazione, un *Oracle* fornisce una risposta ottimale rispetto all'NE del sotto-gioco corrente; tale strategia, considerata ottimale o approssimativamente ottimale, viene aggiunta all'insieme di strategie di ciascun agente. Il processo iterativo termina quando la strategia di risposta ottimale è già presente in tale insieme o quando l'incremento prestazionale risulta trascurabile (figura 2.3).

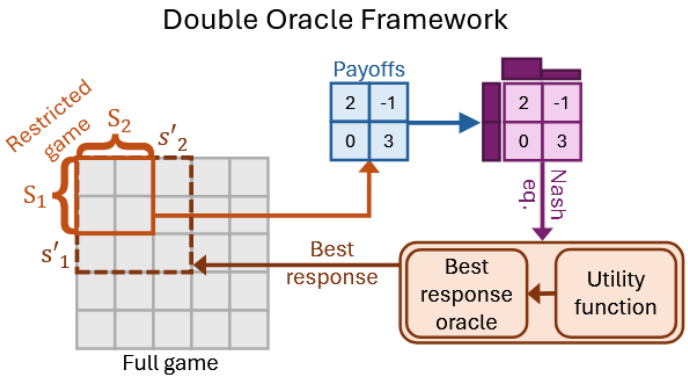

Figura 2.3: Rappresentazione del ciclo di funzionamento degli algoritmi basati su Double Oracle. [5].

Sebbene il metodo DO rappresenti un procedimento efficiente per approssimare l'NE in giochi a somma zero di ampia dimensione, esso presenta alcune limitazioni. In primo luogo, l'applicazione dei metodi DO richiede un coordinamento tra i giocatori nella risoluzione dell'NE dei sotto-giochi, rendendo necessario che entrambi seguano le medesime dinamiche di apprendimento. Tale requisito risulta poco realistico, in quanto in molti scenari concreti l'avversario può adottare strategie arbitrarie e potenzialmente non stazionarie nei sotto-giochi. In secondo luogo, e in maniera più rilevante, i metodi DO non risultano razionali [11], poiché non forniscono un meccanismo di apprendimento capace di sfruttare le debolezze dell'avversario.

In questo contesto, ODO risulta una soluzione scalabile per giochi a somma zero in forma normale a due giocatori, in cui lo spazio delle strategie è di dimensione proibitiva. Lo strumento teorico principale è fornito dall'analisi no-regret [20, 76] nell'apprendimento online [76]. Combinando tale analisi [30] con la struttura dei metodi DO, l'algoritmo ODO eredita i vantaggi di entrambi gli approcci.

L'elemento centrale del funzionamento di ODO è costituito da una controparte online dell'*Online Single Oracle* (OSO) [26] impiegato nel DO [60], opportunamente modificata per soddisfare la proprietà di no-regret. Un vantaggio determinante dell'OSO consiste nel fatto che il relativo limite di rimpianto non dipende dalla dimensione dell'intero insieme di strategie pure disponibili per un giocatore, bensì dalla dimensione del cosiddetto insieme di *strategie effettive*, il quale cresce linearmente con la dimensione del supporto dell'Equilibrio di Nash.

A differenza degli algoritmi no-regret tradizionali, come MWU [30], nei quali l'intero insieme di strategie pure deve essere considerato a ogni iterazione, l'OSO opera su un sottoinsieme dello spazio strategico complessivo. L'aspetto operativo essenziale consiste nel fatto che, a ogni round t, l'algoritmo valuta l'inserimento di una nuova strategia soltanto qualora essa rappresenti la risposta ottimale rispetto alla perdita media calcolata su una specifica finestra temporale (definita successivamente).

Costo computazionale. Nonostante la progettazione dell'insieme delle strategie effettive, il calcolo della risposta ottimale esatta nell'OSO richiede comunque di considerare l'intero insieme di strategie pure. Tale vincolo può essere rilassato mediante l'impiego di risposte ottimali approssimate, che consentono di preservare l'efficienza computazionale senza compromettere le garanzie teoriche essenziali.

Il costo computazionale dell'algoritmo risulta pertanto dominato dal numero di risposte ottimali calcolate e dalla gestione dell'insieme delle strategie effettive. Il tasso di convergenza di ODO non dipende dalla dimensione del gioco, ma piuttosto dalla dimensione dell'insieme di strategie effettive di entrambi i giocatori. Esiste infatti una relazione lineare tra la dimensione dell'insieme di strategie effettive e la dimensione del supporto dell'NE. Inoltre, in molti giochi reali, la dimensione del supporto dell'NE è effettivamente molto inferiore alla dimensione del gioco. Pertanto, ODO può essere utilizzato sia teoricamente che empiricamente come risolutore in giochi a somma zero di grandi dimensioni [26].

Risultati. La valutazione empirica dell'algoritmo evidenzia un miglioramento rispetto ai principali metodi concorrenti in una vasta gamma di scenari sperimentali. Nello studio di Yang et al. [26], gli autori conducono esperimenti su: giochi in forma normale generati casualmente, un insieme di 15 giochi reali forniti da Czarnecki et al., e due giochi di poker in forma estesa (Kuhn e Leduc), utilizzati nella loro formulazione tabulare.

In particolare, nei giochi di poker tabulari, ODO risulta competitivo con solver come CFR e XFP: in Leduc Poker raggiunge sfruttabilità prossime a quelle di CFR e superiori a quelle dei metodi PSRO, mentre in Kuhn Poker supera persino CFR in termini di sfruttabilità. Infine, quando opposto a un avversario imperfetto, OSO dimostra una rapida capacità di sfruttamento, ottenendo payoff positivi e superando le prestazioni della controparte PSRO, che invece mantiene un comportamento più conservativo [26].

2.3.2 PSRO

Contesto e descrizione algoritmo. L'approccio *Policy Space Response Oracles* (PSRO) [52] si è affermato come una sintesi naturale tra i metodi classici per il calcolo degli equilibri e le tecniche di apprendimento automatico. Tale metodo integra contributi provenienti da diverse comunità scientifiche, offrendo una prospettiva unificata sull'interazione strategica tra agenti.

In PSRO, a ciascun giocatore viene assegnato un insieme iniziale di strategie X_i. Le utilità associate ai profili nello spazio dei profili X vengono quindi stimate, producendo un gioco ristretto iniziale $\hat{G}^{S\downarrow X}$.

Ad ogni iterazione, un *meta-strategy solver* (MSS) seleziona un profilo strategico $\sigma \in \Delta X$ dal gioco ristretto corrente come obiettivo verso cui calcolare una nuova *best response*; qui Δ indica il simplesso delle distribuzioni di probabilità definite sull'insieme considerato. Successivamente, ogni giocatore $i \in N$ apprende in modo indipendente una nuova strategia di risposta $s'_i \in S_i$, valutata rispetto al proprio obiettivo di risposta (RO), rappresentato da una funzione che assegna un valore ai profili strategici [5].

Nel PSRO standard, l'obiettivo di risposta per il giocatore i è definito come $\mathrm{RO}_i(\sigma) = u_i(s'_i, \sigma_{-i})$, e la massimizzazione di tale valore rispetto a s'_i determina la risposta ottimale alle strategie σ_{-i} degli altri giocatori. Durante l'apprendimento, le strategie avversarie vengono mantenute fisse, rendendo l'ambiente stazionario e consentendo il calcolo della risposta ottimale.

Una volta individuata, la strategia s'_i viene aggiunta all'insieme X_i delle strategie disponibili nel gioco ristretto. Tale procedura viene ripetuta fino al soddisfacimento di un criterio di arresto, che può consistere in un numero prefissato di iterazioni oppure nel raggiungimento di un livello minimo di rimpianto per l'equilibrio di Nash (NE) stimato del gioco ristretto [5].

PSRO opera su giochi ristretti, che devono contenere un sottoinsieme sufficientemente rappresentativo delle strategie dell'intero gioco pur mantenendo una complessità computazionale gestibile [3]. L'individuazione di tali sottoinsiemi — con il minor numero possibile di strategie e senza perdita rilevante di informazione strategica — costituisce il cosiddetto problema dell'*esplorazione strategica* [45], che rappresenta un aspetto centrale nello sviluppo di metodi PSRO.

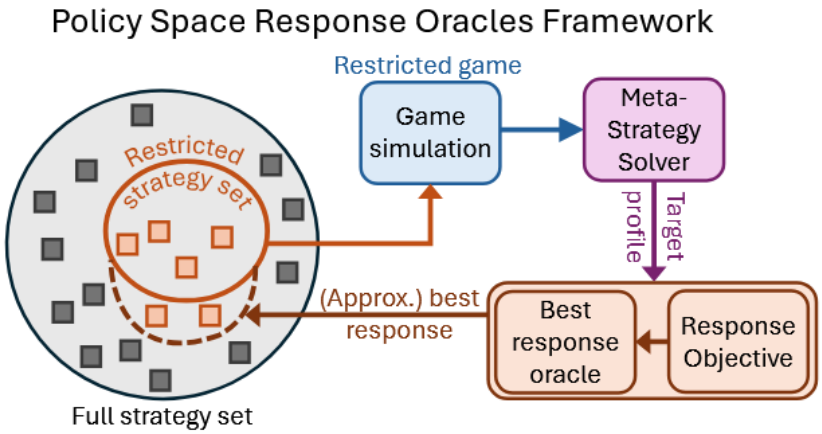

Figura 2.4: Immagine riassuntiva del funzionamento di PSRO. L'algoritmo generalizza DO introducendo MSS, e consentendo obiettivi di risposta ottimale diversi da NE. Inoltre, PSRO supporta vari RO e oracoli di risposta ottimale (approssimativi) [5].

Costo computazionale. Nonostante la sua garanzia di convergenza teorica, raggiungere una convergenza esatta in giochi di grandi dimensioni è spesso impossibile a causa di vincoli come le limitate risorse computazionali. Il costo computazionale dell'approccio, infatti, è determinato principalmente da due componenti: la computazione della risposta ottimale e la simulazione dei payoff nel gioco ristretto. Come evidenziato nello studio [5], entrambe le attività risultano intrinsecamente onerose dal punto di vista computazionale, soprattutto in giochi di grandi dimensioni.

Il calcolo della risposta ottimale richiede l'ottimizzazione di una strategia rispetto al profilo corrente individuato dal risolutore. In giochi complessi o con spazi di stato estesi, tale procedura può richiedere l'uso di metodi di *reinforcement learning* (RL) o di tecniche di ricerca avanzata, comportando un costo significativo in termini di tempo e campioni. La difficoltà cresce ulteriormente quando l'oracolo di risposta deve operare in ambienti non stazionari o parzialmente osservabili, nei quali la ricerca di una risposta ottimale può richiedere numerose iterazioni di apprendimento [5].

Parallelamente, la costruzione e l'aggiornamento del gioco ristretto implicano la stima dei payoff dei profili strategici tramite simulazione. Poiché ogni valutazione di un profilo richiede l'esecuzione di molteplici episodi simulati e la successiva media dei risultati, tale componente rappresenta una parte sostanziale del costo complessivo dell'algoritmo. In questo senso, la simulazione dei profili costituisce uno dei principali colli di bottiglia di

PSRO, tanto che vari lavori successivi hanno introdotto tecniche specificamente orientate a ridurre tale onere [5].

Risultati. PSRO ha dimostrato, nel corso degli anni, una notevole efficacia empirica in una vasta gamma di domini complessi, contribuendo al raggiungimento di risultati importante. In primo luogo, PSRO e i metodi da esso derivati hanno ottenuto prestazioni di rilievo in giochi strategici su larga scala, come dimostrato nel caso di Barrage Stratego [57], in cui sono state superate le migliori prestazioni precedenti e ottenuti risultati competitivi contro giocatori umani esperti. Un successo analogo è stato riportato nel dominio di StarCraft [84], dove approcci ispirati al PSRO hanno consentito di superare in modo convincente sia agenti precedenti sia giocatori umani professionisti.

Oltre ai giochi competitivi classici, PSRO ha trovato applicazione efficace in campi ad alta complessità strutturale. Nel contesto della sicurezza e dei giochi difensivi, questo metodo ha supportato lo sviluppo di strategie ottimali in giochi di sicurezza e scenari simili [5], mostrando capacità superiori nell'identificazione di politiche robuste.

2.3.3 XDO

Contesto e descrizione algoritmo. *Extensive-Form Double Oracle* (XDO) [58] nasce come risposta a una limitazione strutturale del PSRO [5], il quale, pur avendo ottenuto risultati empirici notevoli, presenta una complessità potenzialmente esponenziale quando applicato a giochi estesi di grandi dimensioni, poiché miscela le strategie esclusivamente alla radice dell'albero decisionale.

Alla luce di questa criticità, XDO viene introdotto con l'obiettivo esplicito di ricondurre il paradigma Double Oracle [60] nel dominio dei giochi in forma estesa, superando l'inefficienza rappresentazionale della forma normale. L'idea centrale è di mantenere la popolazione di strategie in forma estesa e di generare, a ogni iterazione, un gioco ristretto in cui le azioni ammissibili in ciascun insieme informativo siano solo quelle effettivamente selezionate da almeno una strategia della popolazione, garantendo una convergenza più efficiente verso equilibri di Nash approssimati [58].

Un aspetto distintivo di XDO è la modalità con cui viene costruito il gioco ristretto:

l'algoritmo opera direttamente sul gioco in forma estesa, in cui a ogni insieme informativo vengono ammesse esclusivamente le azioni selezionate da almeno una strategia della popolazione. Questa trasformazione consente di definire un sottoinsieme del gioco originale che conserva la struttura sequenziale, evitando l'esplosione combinatoria tipica della rappresentazione in forma normale.

Il gioco ristretto viene quindi risolto mediante un algoritmo tabellare per giochi estesi, ottenendo una politica che rappresenta un meta-NE approssimato. Tale politica viene poi estesa al gioco completo attribuendo scelte arbitrarie agli insiemi informativi non visitati nel gioco ristretto [58]. Una nuova *best response* rispetto al meta-NE viene quindi calcolata tramite un oracolo e aggiunta alla popolazione di strategie. Il processo prosegue fino all'iterazione in cui nessuno dei due giocatori è in grado di individuare una best response che migliori il meta-NE: in questa circostanza, le politiche del meta-NE risultano essere risposte ottimali approssimate tra loro anche nel gioco originale e costituiscono pertanto un equilibrio di Nash approssimato [58].

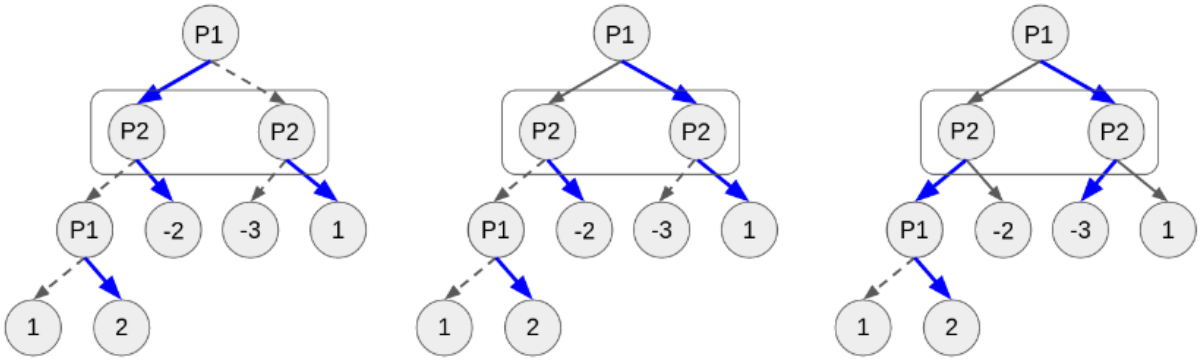

Figura 2.5: Tre iterazioni di XDO (da sinistra a destra). In questi diagrammi di gioco in forma estesa, il giocatore 1 (P1) gioca alla radice, poi P2 gioca senza conoscere l'azione di P1, e se entrambi hanno giocato a sinistra P1 gioca un'altra azione. La ricompensa di P1 è il numero sulla foglia raggiunta. Le azioni nel gioco ristretto sono continue, mentre quelle tratteggiate sono all'esterno del gioco ristretto. Le azioni Meta-NE sono nere, mentre quelle non presenti nel meta-NE sono blu. [58].

Formalmente (figura 2.5), XDO mantiene una popolazione di strategie pure Π_t al tempo t. A ogni iterazione, viene costruito un gioco ristretto definito limitando, per ogni insieme informativo s_i, il set delle azioni ammesse alle sole azioni selezionate da almeno una politica della popolazione. Una politica ε-NE $\pi^{(r*)}$ del gioco ristretto viene quindi calcolata tramite

CFR o un metodo equivalente e successivamente estesa al gioco completo. Infine, per ciascun giocatore viene determinata una *best response* (BR) rispetto al meta-NE $\pi^{(r*)}$, e tali BR vengono aggiunte alla popolazione, così da ottenere Π_{t+1} [58].

Costo computazionale. XDO è garantito convergere in un numero di iterazioni lineare rispetto al numero di insiemi di informazione, in netto contrasto con il comportamento potenzialmente esponenziale di altri metodi simili come PSRO. La motivazione teorica centrale è che in XDO ogni risposta ottimale deve introdurre almeno un'azione nuova in un insieme informativo attualmente non coperto, assicurando un ampliamento monotono del gioco ristretto sino alla copertura completa dello spazio informativo. Ciò consente a XDO di individuare rapidamente gli insiemi di azioni effettivamente rilevanti per l'equilibrio, evitando l'espansione inutile dell'intero spazio strategico [58].

Risultati. L'efficacia di XDO è stata valutata attraverso una serie di esperimenti [58]. Nei test tabellari su Leduc poker, XDO ha mostrato un netto vantaggio in termini di rapidità di convergenza verso strategie a bassa sfruttabilità rispetto ad altri algoritmi, come ad esempio PSRO [5].

Ulteriori esperimenti su 2-Clone Leduc poker hanno evidenziato che XDO necessita di un numero significativamente inferiore di stati visitati rispetto a metodi come CFR+ [12] e MCCFR [51], grazie alla capacità di evitare l'espansione delle azioni ridondanti introdotte dai "cloni".

2.3.4 NXDO

Contesto e descrizione algoritmo. *Neural XDO* (NXDO) [58] viene concepito per essere applicato a classi di giochi caratterizzate da un'elevata complessità, nei quali la generalizzazione sugli insiemi di informazione, resa possibile dall'impiego di strategie derivate da reti neurali, risulta particolarmente vantaggiosa.

A differenza delle versioni basate su un oracolo che fornisce una risposta ottimale esatta, NXDO ne impiega di approssimate, addestrate tramite algoritmi di *deep reinforcement learning* (DRL), quali PPO [74] o DDQN [83]. Inoltre, anziché rappresentare esplicitamente il gioco ristretto come l'insieme delle azioni consentite in ogni insieme di informazione, NXDO

costruisce tale gioco sostituendo lo spazio delle azioni originali con un insieme discreto di meta-azioni. Ognuna di queste meta-azioni corrisponde a una politica della popolazione, alla quale viene delegata la scelta dell'azione concreta.

Formalmente, NXDO mantiene una popolazione di politiche DRL Π_t al tempo t. A ogni iterazione viene generato un gioco ristretto in forma estesa e viene calcolato un meta-equilibrio di Nash (meta-NE). In questo gioco ristretto, in ciascun insieme di informazione sono disponibili meta-azioni che selezionano una politica dalla popolazione corrente. Pur modificando lo spazio delle azioni, NXDO preserva invariati gli stati, le osservazioni e le cronologie del gioco originale. Una volta che ciascun giocatore seleziona una meta-azione, l'azione da eseguire viene campionata dalla politica corrispondente e utilizzata per la transizione dello stato globale [58].

Sulla base di ciò, un meta-NE π_r^* viene calcolato tramite un metodo DRL per la risoluzione di giochi in forma estesa, come NFSP [36]. Successivamente, le risposte ottimali approssimate $\mathrm{BR}_1(\pi_{r,2}^*)$ e $\mathrm{BR}_2(\pi_{r,1}^*)$ vengono apprese tramite algoritmi DRL quali PPO o DDQN, e infine aggiunte alla popolazione. A condizione che tali risposte approssimate siano sufficientemente vicine alle risposte ottimali ideali e che il risolutore interno trovi un meta-NE approssimato con adeguata precisione, NXDO eredita le proprietà di convergenza di XDO.

Costo computazionale. L'espressività del gioco ristretto deriva da un aumento non trascurabile del carico computazionale. In particolare, l'utilizzo di meta-azioni che delegano la scelta all'interno della popolazione comporta una crescita lineare del numero di meta-azioni con il procedere delle iterazioni, rendendo progressivamente più onerosa la risoluzione del gioco ristretto.

Inoltre, la costruzione e la risoluzione del gioco ristretto in forma estesa introduce un costo aggiuntivo, poiché richiede l'addestramento progressivamente più lungo del risolutore del meta-NE (ad esempio NFSP) a ogni iterazione, come mostrato dalle analisi sugli episodi cumulativi impiegati nei cicli interni ed esterni di NXDO [58]. Tale problema risulta particolarmente significativo nei giochi di piccole dimensioni, dove il tempo necessario per calcolare strategie meta-NE accurate può superare quello richiesto per addestrare la maggior parte delle strategie pure, riducendo quindi la competitività di NXDO in tali contesti.

Nello studio di McAleer et al. [58] viene riportato come le prove neurali su giochi complessi, quali *20-Clone Leduc* e il *Loss Game*, richiedano tipicamente da due a quattro

giorni di calcolo con 8–16 core CPU e un consumo di memoria tra 10 e 40 GB, mentre gli esperimenti tabulari richiedono fino a un giorno su un singolo core con un utilizzo di memoria compreso tra 1 e 10 GB.

Risultati. Sebbene le garanzie formali di convergenza possano non risultare pienamente applicabili, NXDO presenta un'elevata compatibilità con spazi di azione continui. Nei giochi caratterizzati da un numero molto ampio di azioni, l'algoritmo consente una riduzione efficace della complessità dell'albero di gioco, le quali non risultano applicabili in presenza di azioni continue.

Al fine di validare empiricamente l'efficacia dell'approccio proposto, sono stati condotti esperimenti su due differenti classi di ambienti [58], tra cui l'*m-Clone Leduc*, è analogo al poker Leduc, ma prevede la duplicazione m delle azioni di *call*, *fold* e *bet*.

I risultati sperimentali evidenziano come NXDO mostri prestazioni superiori sia a PSRO [5] sia a NFSP [36], sia nel caso di *m-Clone Leduc* sia nel *Loss Game* a spazio di azione continuo.

2.3.5 RMDO

Contesto e descrizione algoritmo. Il *Regret Minimizing Double Oracle* (RMDO) [81] rappresenta un'estensione dell'approccio *Double Oracle* (DO) [60] che integra meccanismi di minimizzazione del rimpianto, consentendo di derivare il numero atteso di iterazioni e la complessità campionaria necessaria per raggiungere un ϵ-equilibrio di Nash (ϵ-NE).

RMDO mantiene la struttura generale dei metodi DO tradizionali. Il gioco ristretto viene costruito includendo esclusivamente un sottoinsieme delle strategie pure disponibili nel gioco originale, mentre la popolazione Π_t contiene le strategie attualmente impiegate nel gioco ristretto all'iterazione t. Un ruolo fondamentale è svolto dalle finestre temporali T_j, definite come partizioni dell'insieme delle iterazioni per cui la popolazione rimane invariata: per ogni $t_0, t_1 \in T_j$ vale $\Pi_{t_0} = \Pi_{t_1}$. Il numero di finestre, indicato con k, corrisponde quindi al numero di giochi ristretti generati dall'iterazione iniziale fino all'iterazione finale T. A differenza dei metodi DO preesistenti, RMDO consente di espandere il gioco ristretto in qualunque momento, rendendolo una metodologia più flessibile e generale [81].

La prima componente essenziale di RMDO è la funzione di frequenza $m(\cdot)$, la quale determina il ritmo con cui calcolare la risposta ottimale. Tale funzione è definita come una mappatura da $\mathbb{N} \cap [0, k-1]$ a $\mathbb{N}^+$ e $m(j)$ indica la frequenza di calcolo della risposta ottimale nella j-esima finestra temporale. Poiché l'algoritmo DO basato sulla minimizzazione del rimpianto alterna ciclicamente aggiornamenti CFR e calcolo della BR, risulta cruciale bilanciare opportunamente tali due componenti per favorire una rapida convergenza [81].

La seconda componente chiave è rappresentata dallo schema di media ponderata: nella finestra T_j viene introdotto un peso w_t per ogni iterazione $t \in T_j$, consentendo l'utilizzo di una versione scontata del minimizzatore del rimpianto e accelerando così la convergenza nel gioco ristretto [81].

La procedura operativa di RMDO nella j-esima finestra T_j è la seguente. A ogni iterazione t, dato che la finestra corrente è j, il gioco ristretto G_t viene costruito restringendo le strategie pure nella popolazione Π_t. In G_t la minimizzazione del rimpianto viene eseguita attraversando l'albero del gioco, calcolando il rimpianto a livello di ogni insieme informativo e aggiornando la strategia tramite un algoritmo di minimizzazione controfattuale del rimpianto.

All'inizio, quando $t = 0$, il gioco ristretto e l'aggiornamento strategico vengono omessi poiché Π_0 è vuoto; il valore atteso viene quindi calcolato seguendo una politica completamente casuale. Una volta superata la fase iniziale, quando $t > 0$ e la finestra corrente è T_j, la strategia media congiunta della finestra viene estesa al gioco originale ogni $m(j)$ iterazioni, assegnando probabilità nulle alle azioni non contenute nella popolazione. A questo punto viene calcolata la risposta ottimale del gioco originale rispetto alla strategia media estesa. Le due strategie nuove vengono quindi aggiunte alla popolazione Π_{t+1}. Infine, se l'aggiornamento produce una popolazione diversa da quella precedente ($\Pi_{t+1} \neq \Pi_t$), viene avviata una nuova finestra temporale e la strategia π_{t+1} viene reinizializzata come strategia casuale uniforme.

Costo computazionale. Nello studio condotto da Tang et al. [81] viene dimostrato che la complessità campionaria necessaria per raggiungere un ϵ-NE dipende sia dal numero di finestre temporali k, sia dalla funzione di frequenza $m(j)$ che regola la computazione della risposta ottimale. In particolare, il numero di iterazioni richiesto è dell'ordine $O\Big(k|A||S|^2/\epsilon^2 - k + \sum_j m(j)\Big)$, mentre la complessità campionaria complessiva risulta

$O\Big(k|A||S|^3/\epsilon^2 - k|S| + |S|\sum_j m(j)\Big)$, considerando che sia il minimizzatore del rimpianto sia il calcolo della risposta ottimale richiedono la traversata dell'intero albero del gioco.

Per PDO, che adotta una frequenza di aggiornamento costante $m(j) = c$, la complessità risulta significativamente ridotta: gli autori dimostrano che il numero di iterazioni necessarie è $O(k|A||S|^2/\epsilon^2 + (c-1)k)$ e la complessità campionaria complessiva si attesta a $O\Big(k|A||S|^3/\epsilon^2 + ck|S| + \frac{k|A||S|^3}{c\,\epsilon^2} - \frac{k|S|}{c}\Big)$, risultando dunque polinomiale.

Risultati. Lo studio [81] valuta empiricamente le prestazioni di RMDO su una serie ampia di giochi in forma estesa, tra cui Kuhn Poker, Leduc Poker e Leduc Poker Dummy. Gli esperimenti analizzano la convergenza in termini di sfruttabilità rispetto sia al numero di insiemi informativi visitati sia al tempo di esecuzione.

I risultati mostrano che, nel confronto diretto con XDO, RMDO con periodicità 50 risulta significativamente meno sfruttabile e converge più rapidamente in giochi complessi come *Large Kuhn Poker*. Inoltre, il confronto con algoritmi di riferimento basati sulla minimizzazione del rimpianto, quali CFR+, evidenzia che RMDO mantiene i vantaggi dei metodi Double Oracle nei giochi con equilibrio a supporto ridotto (come Leduc Dummy), risultando al contempo competitivo nei giochi più complessi.

2.4 Metodi di ricerca e approcci euristici

Gli algoritmi di ricerca ed euristici mirano a esplorare selettivamente lo spazio delle decisioni nei giochi ad informazione imperfetta, producendo strategie efficaci senza richiedere la completa risoluzione del gioco. Metodi come *Monte Carlo Tree Search* (MCTS) e la sua estensione ISMCTS utilizzano simulazioni guidate per stimare il valore delle azioni, operando rispettivamente sugli stati o sugli insiemi informativi, risultando utili quando l'albero di gioco è troppo vasto per una valutazione esaustiva. Euristiche come l'*Effective Hand Strength* (EHS) forniscono stime rapide della forza attuale e potenziale della mano, supportando le decisioni di ricerca. Approcci più avanzati, quali MCRNR e MCCFVFP, combinano tecniche Monte Carlo con risposte ottimali o principi di *fictitious play*, apprendendo strategie che bilanciano sfruttamento e robustezza. Nel complesso, questi metodi offrono soluzioni pratiche per gestire la complessità del poker Texas Hold'em Heads-Up Limit.

2.4.1 EHS

Contesto e descrizione algoritmo. L'*Effective Hand Strength* (EHS) è un algoritmo sviluppato nel 1998 con l'obiettivo di competere ai massimi livelli nel poker contro giocatori umani esperti. Esso costituisce il nucleo decisionale dell'agente artificiale *Loki* [7], uno dei primi sistemi in grado di partecipare a partite di poker mediante un modello di valutazione dell'avversario.

Loki, attraverso EHS, integra differenti componenti del gioco per supportare il processo decisionale. Un primo elemento considerato è la *forza della mano* (*Hand Strength*, HS), definita come la valutazione della qualità della propria mano rispetto alle possibili mani avversarie. Tale stima dipende almeno dalle carte private del giocatore e dalle carte comuni disponibili; tuttavia, valutazioni più raffinate tengono conto anche del numero di avversari attivi, della posizione al tavolo e della sequenza di puntate osservata. In forma avanzata, la forza della mano viene calcolata considerando tutte le probabilità associate alle possibili combinazioni avversarie [7], nonché il potenziale di miglioramento della mano nelle fasi successive del gioco.

La probabilità di detenere la mano migliore in un dato momento può essere stimata tramite tecniche di enumerazione completa delle combinazioni possibili. Ad esempio, considerando la mano A♢–Q♣ con un flop 3♡–4♣–J♡, rimangono 47 carte sconosciute, da cui derivano 1081 possibili mani avversarie. L'algoritmo di enumerazione calcola il percentile della mano confrontando il numero di combinazioni migliori, peggiori o equivalenti. In tale scenario, 444 combinazioni risultano superiori, 9 equivalenti e 628 inferiori. Attribuendo ai casi di parità un peso pari a metà, si ottiene una forza della mano

$$\mathrm{HS} = \frac{628 + \frac{9}{2}}{1081} \approx 0.585,$$

ovvero una probabilità del 58.5% che tale mano sia superiore a una mano avversaria casuale. Per più avversari, la stima viene elevata a potenza: contro cinque avversari casuali, la probabilità scende a $0.585^5 \approx 0.069$, evidenziando l'impatto del numero di contendenti.

La forza della mano, tuttavia, non è sufficiente a caratterizzare compiutamente la qualità complessiva di una mano. Ad esempio, la mano 5♡–2♡ con flop 3♡–4♣–J♡ è debolissima nella configurazione attuale, ma presenta un potenziale di miglioramento elevato: l'uscita di un cuore, di un Asso o di un 6 può produrre un colore o una scala. Con due carte comuni ancora da rivelare, la probabilità di ottenere una mano vincente supera il 50%, attribuendo notevole valore a una mano inizialmente sfavorevole.

Il potenziale positivo (*Positive Potential*, Ppot) quantifica la probabilità di migliorare una mano sfavorita fino a renderla vincente, mentre il potenziale negativo (*Negative Potential*, Npot) stima la probabilità di perdere il vantaggio quando si parte in posizione favorevole. Tali valori sono determinati mediante enumerazione: per ciascuna delle 1081 possibili mani avversarie si considerano le 990 combinazioni delle due carte comuni rimanenti, classificando ogni esito come vantaggioso, svantaggioso o in parità.

L'*Effective Hand Strength* [7] combina la forza immediata della mano e il suo potenziale positivo secondo la formula:

$$\mathrm{EHS} = \mathrm{HS}^n + (1 - \mathrm{HS}^n)\ \mathrm{Ppot},$$

dove n rappresenta il numero di avversari ancora attivi. Il valore ottenuto costituisce una stima globale che integra la probabilità attuale di essere in vantaggio e quella di acquisire un vantaggio nelle fasi successive del gioco.

Costo computazionale. EHS presenta un costo computazionale significativo, poiché si basa su tecniche di enumerazione esaustiva di tutte le possibili combinazioni di carte avversarie e delle carte comuni rimanenti. Se il calcolo della sola *Hand Strength* (HS) richiede la valutazione di tutte le $\binom{47}{2} = 1081$ possibili mani dell'avversario dopo il flop, con un confronto completo tra la mano corrente e ciascun esito possibile [7], il calcolo dell'*Hand Potential* comporta un'espansione combinatoria ancora più ampia: per ogni mano avversaria vengono considerate le $\binom{45}{2} = 990$ possibili combinazioni di turn e river, per un totale di oltre un milione di scenari analizzati per singola situazione.

Questa complessità rende l'EHS accurato ma computazionalmente oneroso, richiedendo implementazioni ottimizzate per essere utilizzato in tempo reale.

Risultati. Dal punto di vista sperimentale, i risultati [7] mostrano che tale investimento computazionale produce un miglioramento sostanziale delle prestazioni del sistema. Le versioni del programma *Loki* che integrano EHS con tecniche di modellazione dell'avversario superano in modo netto le versioni prive di modellazione, convergendo più rapidamente verso un profitto medio superiore lungo 100,000 mani.

2.4.2 MCTS

Contesto e descrizione algoritmo. I metodi Monte Carlo rappresentano una delle tecniche più consolidate nell'ambito degli algoritmi numerici e hanno dimostrato un'efficacia significativa anche nei sistemi di gioco basati sull'intelligenza artificiale [2].

Una delle varianti principali di questa famiglia, il *Monte Carlo Tree Search* (MCTS), si fonda su due idee centrali: da un lato, il presupposto che il valore reale di un'azione possa essere approssimato attraverso simulazioni casuali; dall'altro, la possibilità di utilizzare tali stime per orientare progressivamente la politica decisionale verso una strategia di tipo *best-first*. L'algoritmo costruisce gradualmente un albero di ricerca parziale, la cui espansione è continuamente guidata dai risultati ottenuti nelle fasi di esplorazione precedenti. In questo modo, l'albero stesso diventa il supporto principale per stimare il valore delle mosse disponibili, con stime che tendono a migliorare man mano che il numero di simulazioni aumenta e la struttura si arricchisce di nuovi nodi [18].

Il funzionamento di MCTS è quindi di natura iterativa: l'albero viene ampliato fino a quando non viene raggiunto un determinato limite computazionale, solitamente espresso in termini di tempo, memoria o numero massimo di iterazioni. Una volta esaurito il budget a disposizione, la procedura si arresta e viene selezionata l'azione iniziale che, secondo i criteri interni dell'algoritmo, risulta essere la più promettente. Ogni nodo dell'albero rappresenta uno stato del dominio considerato, mentre gli archi che lo collegano ai nodi successivi corrispondono alle azioni che consentono la transizione da uno stato a quello successivo [18]. All'interno di ciascun ciclo di ricerca si passa attraverso quattro fasi fondamentali [22] (figura 2.6).

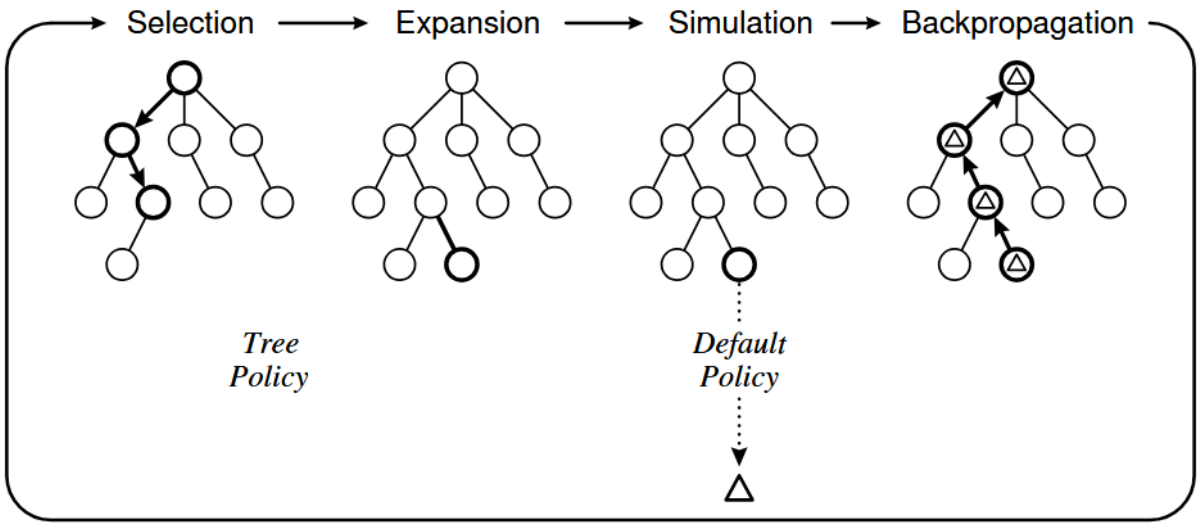

Figura 2.6: Rappresentazione grafica del funzionamento di MCTS [18].

La prima è la *selezione*, durante la quale, a partire dal nodo radice, viene applicata ricorsivamente una politica di scelta che conduce verso i nodi ritenuti più urgenti da esplorare, fino al raggiungimento di uno stato non terminale che presenti ancora possibili espansioni.

Segue l'*espansione*, in cui uno o più nodi figli vengono aggiunti all'albero in base alle azioni disponibili in quello stato.

Successivamente si procede con la *simulazione*, che consiste nell'esecuzione di una partita virtuale a partire dai nuovi nodi, utilizzando una politica predefinita il cui scopo è fornire una stima del valore atteso dello stato.

Infine, durante la fase di *backpropagation*, il risultato ottenuto dalla simulazione viene retropropagato lungo il percorso dell'albero, aggiornando le statistiche dei nodi precedentemente selezionati. Queste informazioni contribuiscono a orientare le future scelte dell'algoritmo, modificando la valutazione degli stati già esplorati.

Al termine dell'elaborazione, l'algoritmo restituisce l'azione radice associata al nodo figlio che risulta "migliore" secondo i criteri stabiliti dall'implementazione. Tra quelli più frequentemente discussi [21, 71] emergono:

- figlio massimo, che privilegia il nodo con la ricompensa empirica più alta;
- figlio robusto, che seleziona il nodo visitato più frequentemente;
- figlio massimo robusto, che ricerca una combinazione favorevole di valore e numero di visite, eventualmente prolungando l'esplorazione se nessun nodo soddisfa requisiti minimi di affidabilità;
- figlio sicuro, che seleziona il nodo in grado di massimizzare un limite inferiore di confidenza, cercando un compromesso tra valore atteso e affidabilità statistica.

Queste varianti riflettono le differenti interpretazioni e finalità operative adottate nelle implementazioni di MCTS, confermando la flessibilità dell'algoritmo rispetto ai criteri decisionali.

Risultati. Il *Monte Carlo Tree Search* ha esercitato un impatto significativo nel campo dell'intelligenza artificiale sin da quando la simulazione Monte Carlo è stata impiegata, per la prima volta, come strumento euristico per individuare in modo selettivo le parti più promettenti dell'albero di ricerca di un gioco. Questo approccio ha dimostrato la propria efficacia in tutti quei contesti decisionali che possono essere formalizzati come problemi di ricerca in grafi o alberi di grandi dimensioni, in particolare quando le decisioni possono essere campionate mediante simulazioni casuali eseguibili con grande rapidità rispetto al tempo reale [18].

Il successo ottenuto in una vasta gamma di giochi, tra cui il *Computer Go* [53, 54], conferma il potenziale dell'approccio per l'intero spettro dei problemi decisionali. La sua efficacia in applicazioni non strettamente ludiche ne evidenzia ulteriormente la versatilità, mostrando come il metodo possa essere esteso con buoni risultati a scenari di pianificazione, ottimizzazione e controllo [18].

Nonostante tali punti di forza, MCTS presenta anche diverse limitazioni. Nei domini caratterizzati da un elevato fattore di ramificazione o da una profondità di ricerca molto estesa, l'applicazione dell'algoritmo puro risulta spesso impraticabile [18], così come avviene per la maggior parte degli approcci di ricerca non informati. Difficoltà ulteriori emergono quando le simulazioni sono computazionalmente costose e il numero di campioni ottenibili è ridotto: in questi casi, l'algoritmo rischia di apprendere da un insieme di dati insufficiente, compromettendo la qualità delle decisioni.

2.4.3 ISMCTS

Contesto e descrizione algoritmo. Gran parte della letteratura su *Monte Carlo Tree Search* si è concentrata su giochi a informazione perfetta. Nonostante ciò, numerosi ricercatori hanno esplorato metodi atti ad adattare MCTS anche a giochi caratterizzati da informazioni parzialmente nascoste. Tra questi, uno degli approcci più diffusi è la *determinizzazione* delle informazioni mancanti, tecnica che ha ottenuto risultati rilevanti in giochi quali *Bridge* [33] e *Klondike Solitaire* [9]. Tale metodologia consiste nel campionare ripetutamente uno stato completo del gioco a partire dall'insieme informativo del giocatore e nell'applicare a questi stati algoritmi sviluppati per giochi a informazione perfetta.

Sebbene efficace in diversi scenari, la determinizzazione presenta alcune debolezze strutturali ampiamente riconosciute, riguardanti la suddivisione del budget computazionale, che duplica di fatto lo sforzo di esplorazione, e la *strategy fusion* [29], per cui differenti stati compatibili con lo stesso insieme informativo sono trattati come nodi distinti dell'albero, finendo implicitamente per assumere che sia possibile selezionare decisioni diverse a partire da essi, violando l'assunzione fondamentale secondo cui un giocatore non può distinguere tra stati appartenenti allo stesso insieme informativo.

Per superare parte di queste criticità viene introdotta la famiglia di algoritmi nota come *Information Set Monte Carlo Tree Search* (ISMCTS) [24].

Il funzionamento di ISMCTS si articola attraverso iterazioni guidate da *determinizzazioni*, ovvero istanziazioni complete dello stato coerenti con l'informazione disponibile al giocatore. A ogni iterazione l'algoritmo seleziona una determinizzazione casuale dall'insieme informativo corrente e limita l'esplorazione dell'albero ai nodi e alle azioni compatibili con essa. Tale meccanismo permette di gestire correttamente la variabilità delle azioni

disponibili nei diversi stati appartenenti allo stesso insieme informativo: nei nodi in cui l'avversario è il giocatore attivo, ad esempio, le azioni disponibili possono variare tra determinizzazioni diverse e la selezione viene trattata come un *subset-armed bandit problem*, adattando opportunamente una formula UCB per bilanciare esplorazione e sfruttamento rispetto alla frequenza con cui un'azione risulta effettivamente disponibile [24].

Ogni iterazione segue la struttura caratteristica di MCTS: si procede con una fase di *selezione* che discende l'albero utilizzando una variante modificata di UCB, limitandosi ai rami compatibili con la determinizzazione corrente; si effettua poi l'*espansione* del nodo quando si incontra un'informazione non ancora rappresentata nell'albero; quindi si esegue una *simulazione* completa a partire dalla determinizzazione selezionata; infine, il risultato ottenuto viene *retropropagato* lungo il percorso seguito, aggiornando sia le statistiche dei nodi visitati sia il conteggio di disponibilità delle azioni nei nodi concorrenti [24].

In questo modo, le statistiche accumulate riflettono l'effettiva probabilità che un'azione sia utile e disponibile negli stati compatibili con l'informazione del giocatore.

Costo computazionale. La sostituzione della ricerca su stati pienamente determinati con una ricerca in cui ogni nodo dell'albero rappresenta un insieme di informazioni anziché uno specifico stato del gioco, consente di unificare in un unico albero le statistiche associate alle mosse, rendendo quindi più efficiente l'utilizzo del budget computazionale: invece di costruire più alberi indipendenti ISMCTS aggrega conoscenza proveniente da una pluralità di stati compatibili all'interno della medesima struttura di ricerca [24].

Oltre a questa efficienza computazionale, l'approccio offre anche un modello decisionale più coerente con la natura informativa del dominio considerato. Dal momento che la selezione delle mosse avviene direttamente sugli insiemi informativi, la ricerca tiene automaticamente conto della validità di un'azione su molteplici stati possibili, attenuando o eliminando i problemi causati dalla fusione di strategie. In questo modo, ISMCTS produce stime più realistiche delle decisioni ottimali in giochi con informazione imperfetta e si configura come un passo significativo verso una gestione più rigorosa dell'incertezza nei metodi Monte Carlo [24].

Risultati. I risultati sperimentali [24] mostrano in modo chiaro che gli algoritmi della famiglia ISMCTS offrono un miglioramento significativo rispetto agli approcci basati sulla

determinizzazione nei giochi a informazione imperfetta. Nei test condotti su domini eterogenei — *Lord of the Rings: The Confrontation*, il gioco *Phantom (4,4,4)* e il gioco di carte *Dou Di Zhu* — ISMCTS si distingue soprattutto nei contesti in cui la determinizzazione soffre maggiormente gli effetti della *strategy fusion* o dell'inefficiente utilizzo del budget computazionale.

2.4.4 MCRNR

Contesto e descrizione algoritmo. L'algoritmo Monte-Carlo Restricted Nash Response (MCRNR) [65] è sviluppato per bilanciare diverse esigenze. Innanzitutto, poiché una strategia perfettamente razionale in equilibrio di Nash (NES) non coincide necessariamente con la migliore controstrategia rispetto a comportamenti avversari non ottimali [64], è possibile ottenere risultati significativamente migliori sfruttando tali debolezze attraverso risposte ottimali mirate. Tuttavia, un'eccessiva focalizzazione sullo sfruttamento dell'avversario comporta il rischio di diventare a propria volta altamente vulnerabili.

MCRNR affronta questo compromesso: il metodo, basato sul campionamento, consente il calcolo offline di strategie di Nash ristrette in giochi complessi rappresentati in forma estesa, tramite la combinazione di MCCFR [51], che consente di approssimare strategie di equilibrio mediante controfattuali campionati, e la tecnica della *Restricted Nash Response* (RNR) [42, 43], concepita per integrare in modo controllato la conoscenza del comportamento avversario all'interno del processo di ricerca dell'equilibrio.

L'idea alla base di RNR è infatti quella di introdurre un modello dell'avversario all'interno del processo di ricerca dell'equilibrio, limitando la strategia dell'agente a un insieme ristretto di possibili risposte che bilancino sfruttamento e sicurezza. In pratica, RNR costruisce un gioco modificato nel quale l'avversario segue la strategia modellata con una certa probabilità, mentre con la restante probabilità si comporta come un giocatore razionale qualsiasi. In questo modo, il giocatore ottiene una strategia che tende a sfruttare l'avversario modellato, ma senza diventare eccessivamente vulnerabile a strategie alternative. Tuttavia, l'applicazione diretta di RNR ai giochi in forma estesa rimane costosa, poiché richiede l'analisi sistematica di porzioni molto ampie dell'albero di gioco.

Per superare questo limite, viene utilizzato MCRNR [65], che rappresenta una versione campionata di RNR integrata con la procedura di riduzione del rimpianto propria di

MCCFR. MCRNR utilizza campionamenti Monte Carlo per attraversare solo le parti rilevanti dell'albero di gioco e aggiornare progressivamente le strategie attraverso i rimpianti controfattuali, mantenendo inalterate le proprietà fondamentali di RNR. Il risultato è un algoritmo capace di apprendere strategie che sfruttano in misura controllata gli avversari non ottimali, ma che al tempo stesso mantengono un livello di sfruttabilità contenuto, avvicinandosi alle garanzie tipiche degli equilibri di Nash.

Costo computazionale. Dal punto di vista computazionale, MCRNR rappresenta un miglioramento sostanziale rispetto alla versione classica di RNR, grazie all'uso del campionamento Monte Carlo: mentre un'iterazione di RNR (così come di CFR) richiede l'aggiornamento di tutti gli insiemi informativi dell'albero di gioco, un'iterazione di MCRNR aggiorna soltanto gli insiemi informativi attraversati dalla singola storia terminale campionata. In questo modo, ogni iterazione tocca solo una piccola frazione del grafo, riducendo drasticamente il tempo di calcolo necessario per ogni passo dell'algoritmo [65].

Lo studio di Ponsen et al. [65] evidenzia infatti che milioni di iterazioni campionate con MCRNR corrispondono, in termini di porzioni realmente esplorate, a un numero molto inferiore di iterazioni complete di RNR, con un risparmio di tempo significativo. Questo vantaggio consente a MCRNR di convergere più rapidamente verso strategie robuste, anche in giochi complessi come il poker, pur mantenendo la capacità di incorporare un modello dell'avversario tramite il parametro di confidenza p, che modula il compromesso tra sfruttamento e robustezza strategica.

L'efficienza computazionale dell'algoritmo lo rende quindi particolarmente adatto a scenari nei quali il tempo di apprendimento risulta limitato o in cui l'albero di gioco è troppo ampio per essere esplorato esaustivamente [65].

Risultati. Gli esperimenti [65] mostrano chiaramente che MCRNR offre prestazioni superiori rispetto a RNR sia in termini di velocità di convergenza sia in termini di qualità delle strategie apprese. Nei giochi di dimensioni ridotte, come *OCP*, *Goofspiel*, *Bluff* e *PAM*, MCRNR produce un'approssimazione dell'equilibrio molto più rapidamente, soprattutto nelle prime iterazioni, dimostrando che il campionamento favorisce un apprendimento iniziale più efficiente. I risultati indicano anche che il parametro p incide fortemente sul bilanciamento tra sfruttamento dell'avversario e sfruttabilità, con una gamma efficace tipicamente compresa tra 0.5 e 0.8 o, in alcuni giochi, tra 0.97 e 1.

Nei test condotti sul poker, dominio di gran lunga più complesso, MCRNR si dimostra capace di sfruttare bot deboli come *POKI* in misura significativamente maggiore rispetto a MCCFR. Anche contro avversari più solidi, l'algoritmo mantiene prestazioni competitive, analoghe a quelle di MCCFR, dimostrando che le strategie generate rimangono robuste e non eccessivamente sfruttabili.

2.4.5 MCCFVFP

Contesto e descrizione algoritmo. L'algoritmo *Monte Carlo Counterfactual Value-Based Fictitious Play* (MCCFVFP) [66] nasce dall'esigenza di combinare i vantaggi del campionamento Monte Carlo, che riduce drasticamente il numero di nodi visitati a ogni iterazione, con la struttura del *Fictitious Play*, introducendo i valori controfattuali nel calcolo delle risposte ottimali. L'obiettivo è duplice: da un lato ottenere un metodo capace di convergere teoricamente verso un equilibrio di Nash in giochi estesi, dall'altro sfruttare in modo efficace la presenza, tipica dei giochi reali su larga scala, di un'elevata proporzione di strategie dominate.

MCCFVFP combina quindi l'efficienza di MCCFR con la logica strategica del FP, introducendo un meccanismo di aggiornamento basato sui valori controfattuali anziché sui tradizionali valori di rimpianto. In MCCFVFP, per ogni insieme informativo vengono calcolati e aggiornati i valori controfattuali $Q_i(I, a)$, ottenuti accumulando le ricompense associate alla scelta sistematica dell'azione a in I. A differenza di CFR, l'algoritmo non richiede la computazione del rimpianto né l'utilizzo della procedura di *regret-matching*: la strategia per l'iterazione successiva viene determinata selezionando semplicemente l'azione con valore controfattuale massimo, producendo quindi una risposta ottimale pura in stile Fictitious Play. Questa scelta riduce in modo drastico il carico computazionale, perché elimina molte operazioni necessarie ai metodi RM, come la normalizzazione del rimpianto positivo, e sfrutta il fatto che, nelle iterazioni, la probabilità di raggiungere ciascun insieme informativo è determinata da strategie pure [66].

Il funzionamento in modalità Monte Carlo permette inoltre di aggiornare solo gli insiemi informativi effettivamente visitati nella traiettoria campionata, limitando così la computazione a una piccola parte dell'albero di gioco. L'adozione di risposte ottimali pure, insieme alla natura selettiva del campionamento, incrementa significativamente le opportunità di

potatura, poiché molti nodi del gioco risultano automaticamente irrilevanti quando la probabilità congiunta di raggiungerli è nulla. Ciò riduce il numero effettivo di nodi visitati da $O(|S|)$ in MCCFR a circa $O(|N|\sqrt{|S|})$ [66].

Costo computazionale. L'algoritmo MCCFVFP è altamente efficiente nel risparmio di risorse di calcolo. Ad esempio, quando un insieme informativo I dispone di $|A(I)| = x$ azioni, MCCFVFP necessita soltanto di $2x + 1$ operazioni di somma per aggiornare i valori controfattuali, mentre MCCFR richiede $6x - 2$ addizioni e ulteriori $3x$ moltiplicazioni per eseguire lo stesso passaggio. Questa differenza implica che, a parità di condizioni, MCCFVFP utilizza circa 2/9 del tempo di calcolo richiesto da MCCFR per ogni insieme informativo. Considerando che la fase di addestramento in giochi reali attraversa tipicamente oltre un miliardo di nodi, la riduzione del costo per nodo si traduce in un miglioramento ingegneristico di grande rilievo, sia in termini di tempo sia di consumo di risorse [66].

A questa efficienza si aggiunge un ulteriore vantaggio legato alla potatura dell'albero di gioco, una tecnica fondamentale nei metodi di ricerca su alberi decisionali. Nei metodi basati su CFR, la forma più semplice è applicabile quando nessun giocatore ha una probabilità positiva di raggiungere uno stato s. In tali casi, l'intero sottoalbero radicato in s può essere ignorato senza compromettere il corretto aggiornamento dei rimpianti.

Risultati. Nello studio condotto da Qi et al. [66], l'algoritmo viene testato su una varietà di giochi estesi con informazione imperfetta, tra cui *Kuhn-extension poker*, *Leduc-extension poker* e diverse configurazioni di *Texas Hold'em*. I Risultati mostrano una tendenza chiara: mentre algoritmi a esplorazione completa come CFR+ o DCFR possono inizialmente ottenere prestazioni migliori, all'aumentare della complessità la natura campionata di MCCFVFP permette di superare rapidamente sia MCCFR sia le altre varianti. Nei giochi di dimensioni medio-grandi, MCCFVFP converge in modo stabile più velocemente, mostrando una capacità superiore di sfruttare strutture del gioco, come l'alta presenza di strategie dominate.

In particolare, MCCFVFP ottiene sistematicamente una convergenza più rapida rispetto a MCCFR, sia quando si considerano i nodi toccati durante l'iterazione sia quando si misura il tempo reale necessario per raggiungere un dato livello di sfruttabilità. Poiché l'algoritmo richiede solo circa 2/9 del costo computazionale di MCCFR per processare lo stesso insieme informativo, a parità di nodi elaborati MCCFVFP risulta più efficiente . Nei

test su *Texas Hold'em* a due giocatori, MCCFVFP mostra un vantaggio del 20–30% in termini di rapidità di convergenza della sfruttabilità, mentre negli scenari multigiocatore — valutati tramite competizioni dirette tra agenti — l'algoritmo supera nettamente MCCFR, ottenendo ricompense significativamente migliori nelle stesse finestre di addestramento [66].

Capitolo 3

Opponent Modeling Expectimax

3.1 Expectimax

L'algoritmo *expectimax* costituisce un approccio fondato sulla ricerca euristica per la selezione delle azioni, finalizzato alla pianificazione di una strategia di scommessa a partire da uno specifico stato di gioco. Tale metodo esegue un'induzione a ritroso sull'albero delle decisioni radicato nello stato considerato, analizzando in modo ricorsivo tutte le possibili azioni intraprese e i corrispondenti stati successivi, fino al raggiungimento della conclusione della partita. Al termine della sequenza decisionale, l'esito del gioco (ad esempio vittoria, sconfitta o pareggio) risulta noto. Poiché è nota anche l'azione che ha condotto a tale esito, il valore associato a quell'azione può essere identificato direttamente con il valore dello stato terminale raggiunto [73].

Una volta determinati i valori delle azioni disponibili per un dato giocatore, vengono formulate ipotesi circa le scelte che tale giocatore adotterà. Le ipotesi così definite rappresentano la sua strategia di scommessa e consentono di calcolare il valore complessivo del punto decisionale considerato, permettendo di iterare il processo per le decisioni antecedenti.

Un possibile metodo per integrare la ricerca euristica nella selezione delle azioni nel contesto del poker consiste nel rinunciare alla determinazione esplicita delle azioni ottimali per l'avversario durante la ricerca, assumendo invece che tali scelte siano prese autonomamente dall'avversario stesso. Questa diversa impostazione determina un calcolo finalizzato a individuare la strategia ottimale contro un avversario specifico, anziché contro un avversario ipotetico nel caso peggiore. In termini di teoria dei giochi, il procedimento individua una

strategia di risposta ottimale rispetto a un particolare avversario, e non una coppia di strategie di equilibrio che garantisca a entrambi i giocatori un valore minimo indipendentemente dal comportamento dell'altro [73].

Questa procedura di ricerca euristica è comunemente denominata algoritmo *expectimax* [61, 70], il cui impiego nei processi decisionali relativi al poker è stato introdotto per la prima volta da Davidson [25]. Le ipotesi concernenti il comportamento di ciascun giocatore giustificano la denominazione dell'algoritmo: si assume che il programma che invoca la procedura desideri selezionare l'azione con il valore più elevato tra quelle disponibili, contribuendo così alla componente "max" del nome. Si assume inoltre che l'avversario scelga le proprie azioni secondo una strategia predeterminata, trasformando la sua decisione in un evento di natura probabilistica. Ne deriva che il valore di un nodo decisionale appartenente all'avversario venga calcolato come valore atteso delle sue possibili azioni, ottenuto come somma dei valori delle azioni ponderati in base alla probabilità che esse vengano effettivamente intraprese. Tale componente di valore atteso dà origine alla parte "expecti" della denominazione dell'algoritmo.

Tra i contributi più rilevanti in relazione all'integrazione della modellazione dell'avversario con procedure di ricerca basate su *expectimax*, si collocano tre linee di ricerca. Reibman e Ballard [67] propongono una variante del minimax in cui alcuni nodi avversari sono trattati come nodi casuali tramite un valore atteso basato sulla probabilità che l'avversario scelga azioni subottimali. Jansen [40] introduce la procedura *probi-max*, che sostituisce i backup minimi con backup attesi nei nodi avversari, con probabilità derivate da semplici regole euristiche sulla fallibilità dell'avversario. Sen e Aurora [75], infine, sviluppano un giocatore a massima utilità attesa che apprende un modello probabilistico dell'avversario tramite osservazione e lo utilizza nel processo decisionale, in un framework riconducibile alla ricerca expectimax.

Sebbene gli approcci di modellazione dell'avversario descritti costituiscano un primo insieme di esempi significativi, essi sono stati concepiti prevalentemente per giochi a informazione perfetta. Nel contesto del poker, e in particolare nell'ambito dell'Heads-Up Limit Texas Hold'em (HULHE), la letteratura offre un numero ridotto di applicazioni di tali tecniche, spesso caratterizzate da limitazioni non trascurabili.

Nel lavoro di Billings et al. [8], il comportamento dell'avversario viene modellato principalmente sulla base delle osservazioni storiche, come le frequenze con cui quest'ultimo seleziona specifiche azioni. Altre varianti considerano ulteriori informazioni contestuali,

quali la dimensione del piatto, le puntate correnti o la fase della mano. Sebbene tali metodi consentano di catturare alcuni aspetti statistici del comportamento avversario, essi risultano comunque approssimativi: il fatto che un giocatore tenda, ad esempio, a chiamare frequentemente con mani di valore medio non implica che mantenga la stessa politica decisionale nell'intero arco della partita. In modo analogo, l'adozione di una strategia di base o di un insieme predefinito di strategie da cui campionare le possibili scelte dell'avversario introduce assunzioni e generalizzazioni eccessivamente forti, che riducono l'affidabilità del modello.

Alla luce di tali limitazioni, in questo capitolo viene presentata una variante dell'algoritmo Expectimax che integra la componente di massimizzazione e ricerca propria dell'approccio originale con una modellazione probabilistica dell'avversario fondata sulle possibili mani private e sulle carte comuni che possono essere distribuite. Tale integrazione consente di ottenere una rappresentazione più accurata del comportamento avversario rispetto alle soluzioni presenti in letteratura.

Questo contributo rappresenta un elemento di novità, soprattutto nel contesto di HULHE. Nel lavoro di Schauenberg [73], una forma di expectimax con modellazione dell'avversario viene applicata al Texas Hold'em heads-up, ma con un modello concettualmente più semplice rispetto a quello proposto in questa tesi. Qui viene infatti introdotta una variante specificamente adattata a HULHE, che combina e raffina idee preesistenti dando origine a un approccio originale. Tale metodo prende il nome di *Opponent Modeling Expectimax*.

3.2 OM-Expectimax

L'idea alla base di *Opponent Modeling Expectimax* (OME) consiste nel costruire un unico albero decisionale dal punto di vista del giocatore che intende ottimizzare la propria strategia, modellando l'avversario non come un agente perfettamente razionale (come avviene nel minimax tradizionale), ma come un decisore fallibile descritto tramite una distribuzione di probabilità sulle sue possibili azioni. Questo consente di sostituire i nodi minimi tipici del minimax con nodi di aspettativa, nei quali il valore di ciascun punto decisionale dell'avversario viene calcolato come valore atteso rispetto alla sua probabilità di compiere ciascuna azione.

La struttura dell'albero decisionale prevede tre categorie fondamentali di nodi: nodi di decisione del giocatore che sta massimizzando (nodi *max*), nodi che rappresentano le

decisioni dell'avversario (nodi *opponent*), e nodi di aleatorietà (nodi *chance*) legati alla distribuzione delle carte comunitarie. In corrispondenza dei nodi di tipo *max*, il giocatore seleziona l'azione che massimizza il valore dei possibili stati successivi, secondo la regola

$$V(s) = \max_{a \in A(s)} V(s_a),$$

dove $A(s)$ rappresenta l'insieme delle azioni disponibili e s_a lo stato risultante dall'azione a.

Nei nodi dell'avversario, invece, si assume che l'altro giocatore adotti una politica di comportamento non ottimale, ma descrivibile tramite una distribuzione di probabilità $\pi(a \mid s)$ sulle azioni disponibili. Come si osserva nel pseudocodice dell'algoritmo (figura 3.1), tale distribuzione non è arbitraria: viene costruita sulla base di una stima della forza della mano avversaria, rappresentata da una misura di *hand strength* o *effective hand strength* calcolata per tutte le possibili combinazioni compatibili con le informazioni osservabili. Una volta calcolata l'EHS per ciascuna ipotesi di mano avversaria, essa viene mappata in una probabilità sulle azioni disponibili. L'idea è che un avversario con mano presumibilmente forte tenderà ad assumere comportamenti più aggressivi (puntare o rilanciare), mentre un avversario con mano debole mostrerà una maggiore propensione ad abbandonare. In termini qualitativi, si possono modellare tali dipendenze con funzioni del tipo: $p(\text{raise} \mid s, h) \propto \alpha_r \cdot \text{EHS}(h)$, $p(\text{call} \mid s, h) \propto \alpha_c \cdot \big(1 - |\text{EHS}(h) - 0.5|\big)$, e $p(\text{fold} \mid s, h) \propto \alpha_f \cdot (1 - \text{EHS}(h))$, dove h denota una possibile mano avversaria e i coefficienti $\alpha_r, \alpha_c, \alpha_f$ calibrano la sensibilità del modello. Dopo normalizzazione, tali probabilità vengono aggregate su tutte le possibili mani avversarie:

$$\pi(a \mid s) = \sum_{h \in H_s} P(h \mid s)\, p(a \mid s, h),$$

dove H_s è l'insieme delle possibili mani avversarie e $P(h \mid s)$ è la probabilità che l'avversario detenga la mano h, tipicamente uniforme sulle combinazioni compatibili.

Il valore dei nodi avversari viene quindi calcolato tramite la regola stocastica:

$$V(s) = \sum_{a \in A(s)} \pi(a \mid s)\, V(s_a),$$

con s_a stato generato dall'azione dell'avversario. Ciò consente di modellare l'avversario come un decisore probabilistico informato, anziché come un avversario perfetto ostile.

Input: s stato corrente, A insieme di azioni disponibili
Output: Distribuzione $\pi(a \mid s)$ sulle azioni $a \in A$
$D \leftarrow \text{MazzoResiduo}(s)$
$H \leftarrow \text{ManiAvversariePossibili}(D, s)$
if $|H|$ *grande* **then**
  $H \leftarrow \text{Campiona}(H)$
end
foreach $a \in A$ **do**
  $\pi[a] \leftarrow 0$
end
foreach $h \in H$ **do**
  $ehs \leftarrow EHS(h, \text{community}(s))$
  foreach $a \in A$ **do**
    if $a =$ *raise* **then**
      $p_h[a] \propto f_{\text{raise}}(ehs)$
    else if $a =$ *call/check* **then**
      $p_h[a] \propto f_{\text{call}}(ehs)$
    else
      $p_h[a] \propto f_{\text{fold}}(1 - ehs)$
    end
  end
  normalizza p_h
  foreach $a \in A$ **do**
    $\pi[a] \leftarrow \pi[a] + p_h[a]$
  end
end
foreach $a \in A$ **do**
  $\pi[a] \leftarrow \pi[a]/|H|$
end
normalizza π
return π

Figura 3.1: Stima della politica avversaria $\pi(a \mid s)$ a partire dall'Effective Hand Strength. In questo caso, il modello mostra un campionamento nel caso in cui le mani avversarie da valutare siano troppo elevate, ma ciò dipende dalle risorse computazionali disponibili e limiti dello specifico contesto applicativo.

I nodi di aleatorietà (nodi *chance*) trattano la distribuzione delle carte future (*flop*, *turn*, *river*) come un processo stocastico puro: tutte le combinazioni possibili di carte residue vengono enumerate e ponderate uniformemente. Il valore del nodo è dunque la media dei valori dei nodi successivi:

$$V(s) = \frac{1}{|\Omega|} \sum_{c \in \Omega} V(s_c),$$

dove Ω rappresenta l'insieme delle possibili distribuzioni di carte compatibili con la situazione corrente.

Il processo termina nei nodi finali, nei quali la mano è conclusa tramite *fold* o *showdown*. Tali nodi vengono valutati calcolando il valore atteso del risultato, che dipende dal contenuto del piatto, dalla forza relativa delle mani e dalle possibili carte restanti, qualora vi siano ancora elementi ignoti. La valutazione terminale rappresenta quindi una stima del guadagno atteso in quello stato. Una panoramica della logica di funzionamento dell'algoritmo viene proposta nella figura 3.2

Input: s stato, $tipo \in \{\text{MAX}, \text{OPP}, \text{CHANCE}\}$, profondità d
Output: $V(s)$
if s *terminale* ***or*** $d = 0$ **then**
  return Valutazione(s)
end
if $tipo = CHANCE$ **then**
  return $\sum_{c \in C} \Pr(c \mid s)\, V(s_c, tipo_c, d-1)$
else
  $A \leftarrow$ Azioni(s)
  if $A = \emptyset$ **then**
    $s' \leftarrow$ CambiaGiocatore(s)
    return $V(s', \text{MAX}, d-1)$
  end
end
if $tipo = MAX$ **then**
  return $\max_{a \in A} V(s_a, tipo_a, d-1)$
else
  $\pi \leftarrow$ StimaPoliticaAvversaria(s, A)
  return $\sum_{a \in A} \pi(a)\, V(s_a, tipo_a, d-1)$
end

Figura 3.2: Pseudocodice dell'algoritmo OM-Expectimax. Da notare che non sono presenti, per semplicità, le gestioni esplicite relative ai nodi di transizione (*skip*). Si può però considerare il loro funzionamento come simile alla casistica senza azioni disponibili presente nelle righe centrali.

Una componente meno evidente ma concettualmente importante del modello è costituita dai nodi di transizione, o nodi *skip*. Questi nodi non rappresentano una decisione autonoma: il loro scopo è gestire correttamente l'alternanza dei turni tra giocatore ed avversario, permettendo di mantenere una coerenza simile a quella degli altri algoritmi di ricerca, essendo che nel poker HULHE il giocatore ad agire all'inizio del round non è obbligatoriamente l'ultimo ad aver agito nel round precedente.

Dal punto di vista dell'albero, i nodi *skip* funzionano come nodi di sola transizione: essi non alterano il valore del processo decisionale, ma mantengono la coerenza strutturale dell'albero garantendo che ogni decisione sia attribuita al giocatore corretto. Questo accorgimento permette di evitare distorsioni sia nella propagazione dei valori sia nella generazione dei nodi figli, mantenendo la separazione concettuale tra le decisioni effettive (che aggiornano le probabilità o selezionano azioni ottimali) e i passaggi tecnici necessari per gestire l'ordine dei turni.

3.3 Costo computazionale

Opponent modeling expectimax, offrendo una modellazione molto approfondita dell'interazione tra comportamento dell'avversario, incertezza sulle carte future e processo decisionale ottimizzante, presenta un costo computazionale intrinsecamente elevato. Il motivo principale risiede nella natura combinatoria del problema: a ogni nodo dell'albero decisionale si aprono rami multipli dovuti alle azioni del giocatore, alle possibili reazioni probabilistiche dell'avversario e alle molteplici combinazioni di carte che possono essere distribuite nei nodi di aleatorietà. In particolare, i nodi dell'avversario rappresentano la componente più onerosa in termini di calcolo, poiché richiedono di considerare tutte le combinazioni di carte che l'avversario potrebbe detenere. Se il mazzo residuo contiene n carte, il numero di mani private compatibili è pari a $\binom{n-2}{2}$, e per ciascuna di esse è necessario calcolare una stima di *hand strength* o *effective hand strength*. Tali stime richiedono a loro volta la valutazione della forza relativa della mano su tutte le possibili completazioni delle carte comunitarie, generando un costo di ordine combinatorio che cresce rapidamente all'aumentare delle carte non ancora distribuite (figura 3.3).

Anche i nodi di aleatorietà introducono una complessità significativa: quando il modello deve considerare, ad esempio, la distribuzione del *turn* o del *river*, il numero di combinazioni da analizzare può raggiungere valori dell'ordine di decine o centinaia, ciascuna delle quali genera un nuovo ramo dell'albero da esplorare. Pertanto, nei livelli intermedi dell'albero — dove le carte comunitarie non sono ancora tutte rivelate — la complessità esplode in modo combinatorio, sia per le carte residue sia per le possibili mani dell'avversario.

Il costo computazionale dei nodi *max* è più contenuto, poiché il numero di azioni possibili in un punto decisionale di poker è limitato, ma il loro contributo alla crescita dell'albero

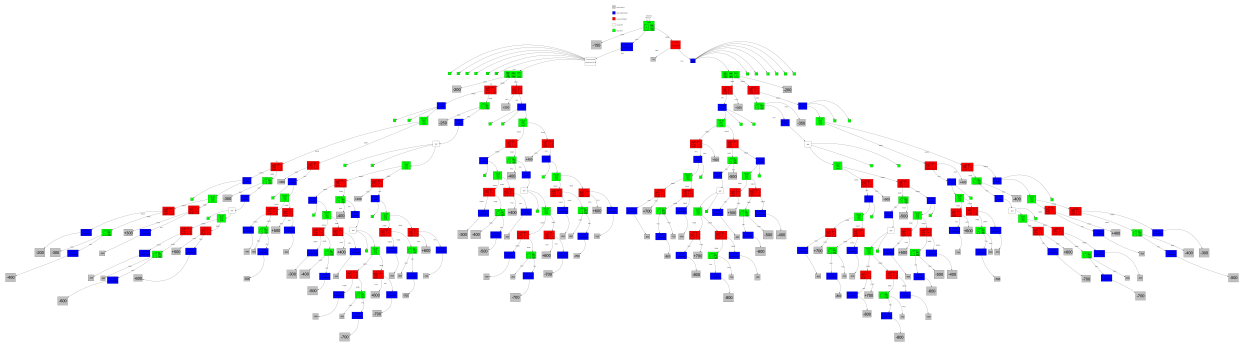

Figura 3.3: Rappresentazione di una piccola porzione dell'albero di gioco di Expectimax. La raffigurazione riguarda un test condotto su un mazzo di 10 carte e prende in considerazione solamente una delle possibili distribuzioni di carte, con la limitazione a un singolo rilancio per turno. Ciò mostra in modo evidente la crescita enorme dell'albero decisionale anche in condizioni estremamente semplificate. I nodi blu rappresentano i nodi *chance*, i verdi sono nodi *max*, i rossi rappresentano nodi *opponent*, mentre i nodi *skip* sono di colore bianco. Le foglie dell'albero, di colore grigio, rappresentano infine i nodi terminali

resta comunque non trascurabile. Il ricorso ai nodi di transizione (*skip*) può mitigare parzialmente la crescita dell'albero, eliminando la necessità di generare rami non significativi e mantenendo l'albero coerente senza creare nodi decisionali fittizi, ma esso non ne riduce la complessità intrinseca.

Combinando tutte queste componenti, la complessità teorica dell'algoritmo può essere approssimata da un modello del tipo:

$$T(d) \approx b_{\max}^{d} \cdot \binom{n-2}{2} \cdot |\Omega(d)|,$$

dove $b_{\max}$ è il fattore di diramazione determinato dalle azioni disponibili, $\binom{n-2}{2}$ rappresenta la dimensione dello spazio delle mani avversarie possibili e $|\Omega(d)|$ denota la quantità di combinazioni di carte future da simulare al livello di profondità d dell'albero. Tale espressione, pur semplificata, mette in evidenza come il costo cresca rapidamente con la profondità della ricerca e con il numero di elementi nascosti nel gioco.

Ne consegue che, senza tecniche di ottimizzazione, caching o riduzione della profondità, l'esplorazione esaustiva dell'albero sarebbe impraticabile. Per questo motivo, l'algoritmo viene tipicamente limitato da una profondità massima di ricerca e fa largo uso di memorizzazione dei risultati intermedi per evitare ricalcoli ridondanti. Inoltre, tramite parallelizzazione, l'algoritmo riesce a esplorare contemporaneamente diverse parti dell'albero di gioco,

diminuendo il tempo di esecuzione. Per ridurre ulteriormente il carico computazionale, si potrebbe considerare di sostituire HS o EHS con un calcolo euristico più veloce o guidare l'esplorazione tramite simulazioni Monte Carlo, campionando quindi le possibili mani avversarie e le carte future anziché enumerarle tutte.

Capitolo 4

Risultati sperimentali su HULHE

4.1 Configurazione degli esperimenti

Implementazione e addestramento algoritmi. Gli algoritmi descritti sono stati implementati sulla base della letteratura scientifica ad essi dedicata, modificandone alcuni aspetti in caso di necessità di adattemnto alla variante HULHE. Per mantenere una coerenza sperimentale, il tempo di addestramento è stato fissato a circa una settimana in totale, sulla base delle risorse disponibili e con la possibilità di tracciarne la convergenza e prestazioni allo stesso livello di addestramento.

Essendo HULHE un gioco molto ampio, le implementazioni di algoritmi tabellari (come CFR) sono state realizzate con un grado di astrazione: la chiave dell'insieme informativo teneva conto di grandezza del piatto, puntate dei giocatori, turno, e *bucket* (ovvero la categoria) della mano. In particolare, la decisione è stata di utilizzare 169 bcukets per il pre-flop, 1000 per il flop, 500 per il turn e 200 per il river, ottenuti tramite il calcolo di EHS. La conseguenza è una perdita di informazione, ma la riduzione della dimensione dello spazio degli insiemi informativi ha permesso di eseguire l'addestramento in tempi ragionevoli, pur mantenendo una buona qualità delle strategie apprese. I metodi CFR hanno raggiunto circa tra i 90 e 100 mila insiemi informativi, con una politica di ripiego che implicasse l'utilizzo del più simile insieme informativo a quello attuale, nel caso in cui non fosse stato incontrato durante l'addestramento.

Similmente si è adottata un'astrazione per gli approcci Double Oracle: le politiche apprese sono state memorizzate con insiemi informativi basati sulla stessa chiave di quelli di CFR, ma con un bucketing molto meno granulare. Questa decisione si fonda sul fatto che tale tipologia di metodi risulta molto più lenta nella convergenza in giochi ad informazione imperfetta, soprattutto giochi della grandezza di HULHE. Astrazioni basate su EHS sarebbero risultate troppo costose computazionalmente, rendendo quasi obbligatorio ridurre il numero di bucket a 20, selezionati attraverso un'euristica basata sulla forza della mano. Inoltre, il calcolo di una risposta ottimale esatta è molto oneroso in tali metodi, in quanto prevede di traversare l'intero albero di gioco ad ogni strategia aggiunta: per questo, l'oracolo fornisce, nelle implementazioni realizzate, una risposta ottimale approssimata.

Gli algoritmi basati su ricerca, fatta eccezione per MCRNR e MCCFVFP, sono stati implementati senza astrazione, essendo che la ricerca avviene a partire dallo stato attuale e non richiede la memorizzazione di strategie per tutti gli insiemi informativi. Tuttavia, per mantenere tempi di esecuzione ragionevoli, sono state adottate alcune limitazioni di profondità nella ricerca, oltre a tecniche di caching per evitare ricalcoli ridondanti e parallelizzazione per una visita più efficiente dell'albero di gioco. Allo stesso modo, negli approcci basati su FP si è deciso di non usare astrazione sulla mano del giocatore, in modo da seguire più fedelmente possibile la logica dell'algoritmo originale, in cui è essenziale avere il massimo grado di precisione a granularità, e la letteratura originale, che, a differenza di quella presente per CFR, non menziona l'utilizzo di astrazione.

Nonostante le differenze, gli algoritmi sono stati sottoposti a test preliminari e controlli per poter essere utilizzati. Innanzitutto, sono stati eseguiti test in partite contro avversari semplici con politiche fisse, consistenti in una sola azione (come solo *raise* o *call*) oppure semi-casuali, guidate da euristiche semplici di forza della mano. Inoltre, per controllare la convergenza, il calcolo della sfruttabilità sarebbe stato molto oneroso, essendo un calcolo della risposta ottimale, per ciascun algoritmo, sull'intero albero di gioco. Si è scelto quindi di controllare altri parametri: nel caso di approcci basati su rimpianti controfattuali, il numero di insiemi informativi nuovi trovati ad ogni iterazione cala progressivamente, fino a rimanere quasi fisso, e la distribuzione di probabilità sulle azioni tende a stabilizzarsi, con variazioni minime tra un'iterazione e l'altra. Per gli approcci basati su FP, si è monitorata la variazione delle prestazioni nel tempo, considerando le versioni precedenti come avversari, fino ad un momento nel quale il miglioramento è diventato sempre meno significativo. Infine, nei metodi basati su oracoli la convergenza è stata controllata in funzione del cambiamento

delle politiche e dell'aggiunta di nuove risposte ottimali calcolate, anche se, nell'arco di tempo fisso dell'addestramento, tali approcci sono rimasti lontani dall'equilibrio, essendo HULHE una variante molto grande.

Parametri dei test. La fase sperimentale è stata articolata su diversi livelli di complessità, con l'obiettivo di analizzare in modo sistematico il comportamento degli algoritmi al variare del contesto. In particolare, lo scopo principale degli esperimenti è stato quello di osservare le prestazioni degli agenti in confronti diretti (*head-to-head*), misurando per ciascuno di essi il guadagno o la perdita media contro ogni altro algoritmo all'interno di un dominio caratterizzato da elevata variabilità e forte componente stocastica.

Per raggiungere tale obiettivo, l'insieme di test è stato organizzato in modo graduale, aumentando progressivamente la quantità di mani giocate e, di conseguenza, riducendo l'impatto della varianza sui risultati. È stata quindi adottata una struttura a tre livelli:

1. Match con bankroll iniziale di 1500 chips, con struttura di puntate pari a 50 per la *small bet* e 100 per la *big bet*. Questa configurazione fornisce una prima indicazione sulle differenze di performance, pur mantenendo elevata la variabilità dovuta al numero ridotto di mani.

2. Match con bankroll iniziale di 10 000 chips, mantenendo invariata la struttura delle puntate. L'aumento significativo del capitale iniziale consente di ottenere una maggiore profondità di gioco e una riduzione della componente aleatoria, permettendo una valutazione più stabile e affidabile delle prestazioni.

3. Esperimenti su 1000 mani singole giocate, con l'obiettivo di isolare il comportamento degli algoritmi sul lungo periodo, concentrandosi sull'effettivo guadagno medio per mano, indipendentemente dall'esito complessivo del match.

Questo approccio ha permesso di costruire una stima progressivamente più accurata del guadagno medio generale di ciascun agente, evidenziando con maggiore chiarezza quali strategie tendano a prevalere quando la varianza si riduce.

Per la valutazione delle prestazioni sono state considerate diverse metriche, ciascuna specifica per un aspetto preciso del comportamento degli algoritmi. Innanzitutto, è stato analizzato il numero di mani necessarie per ottenere una vittoria e, più in generale, l'efficienza dell'algoritmo nel convertire situazioni favorevoli in profitto. Tuttavia, la metrica

ritenuta più informativa è stata il guadagno medio per mano, preferita alla semplice percentuale di mani vinte. Un algoritmo può infatti vincere più mani del proprio avversario senza però riuscire a capitalizzare adeguatamente le situazioni vantaggiose, mostrando un rendimento economico inferiore.

Un ulteriore aspetto cruciale è stato la robustezza nei confronti di strategie eterogenee. Alcuni algoritmi di tipo euristico, ad esempio, possono risultare particolarmente efficaci contro determinate tipologie di approcci, pur diventando vulnerabili in contesti differenti. Altri, come quelli basati su principi di ottimalità teorica, assumono che l'avversario selezioni le azioni secondo una strategia razionale, ma possono essere più in difficoltà contro agenti progettati per deviare intenzionalmente da tali ipotesi. Pertanto, un algoritmo che mantiene buone prestazioni contro una vasta gamma di avversari, pur non essendo il migliore in termini di guadagno medio, può comunque essere considerato solido e competitivo nel complesso.

L'insieme di queste metriche ha permesso di delineare un quadro complessivo equilibrato, integrando sia aspetti quantitativi sia considerazioni qualitative sul comportamento degli agenti, e fornendo così una valutazione approfondita delle loro capacità nel dominio del poker HULHE.

4.2 Risultati empirici

Partite con capitale iniziale ridotto. Nel primo insieme di test, corrispondente alle partite con somma iniziale pari a quindici volte il big blind, è emerso un quadro complessivamente equilibrato. In questa configurazione la maggior parte degli algoritmi ha mostrato prestazioni comparabili, con differenze limitate. È stato tuttavia possibile individuare alcune tendenze significative.

Anzitutto, i metodi basati su oracoli, quali XDO, ODO e RMDO, hanno mostrato una chiara fragilità strutturale: la loro capacità di sfruttare l'avversario nel breve periodo non è risultata sufficiente a compensare l'assenza di una componente più solida di ottimizzazione strategica, conducendo a prestazioni mediamente inferiori rispetto agli altri approcci, anche se non troppo distanti.

Parallelamente, alcuni algoritmi maggiormente orientati allo sfruttamento, in particolare quelli fondati su ricerca come OM-Expectimax e ISMCTS, hanno occasionalmente ottenuto

risultati superiori alla media. Pur non perseguendo esplicitamente strategie vicine all'equilibrio, tali metodi hanno beneficiato della possibilità di operare senza astrazione, riuscendo in certe circostanze a capitalizzare errori locali degli avversari.

Nel complesso, tuttavia, anche gli algoritmi basati sulla minimizzazione del rimpianto hanno confermato una buona stabilità. In questo scenario ridotto, il bilanciamento tra varianza e profondità strategica non ha evidenziato differenze marcate, ma ha già suggerito quali metodologie avrebbero mostrato maggiore solidità negli esperimenti successivi.

Partite con capitale iniziale aumentato. L'incremento del capitale iniziale ha reso più evidente la distinzione tra le diverse famiglie di algoritmi. Pur mantenendo un numero medio di mani relativamente contenuto (nell'ordine di poche centinaia), questa configurazione ha ridotto parzialmente l'impatto della varianza, permettendo l'emergere di pattern più chiari.

In questo contesto, gli algoritmi basati su programmazione lineare — in particolare XDO, ODO e RMDO — si sono collocati stabilmente in fondo alla classifica, registrando una perdita media superiore a 0.5 grandi bui per mano. Prestazioni comparabili sono state osservate anche per FSP, nonostante la sua struttura concettualmente più flessibile.

Di contro, gli approcci di ricerca più semplici, come EHS e MCTS, hanno mostrato prestazioni migliori grazie alla maggiore immediatezza con cui riescono a sfruttare pattern locali nel comportamento avversario. Risultati leggermente superiori sono emersi dagli algoritmi neurali quali DeepCFR, NXDO e NFSP, che sono riusciti a ottenere guadagni medi di circa 0.3 grandi bui contro metodi puramente euristici. A un livello simile si sono collocati PSRO e alcuni algoritmi basati su rimpianto controfattuale, tra cui ECFR, RCFR e CFR, che hanno confermato la loro solidità generale.

La parte alta della classifica ha invece visto una sostanziale parità tra i rimanenti algoritmi di ricerca e le varianti più avanzate di CFR, indicando come l'incremento del capitale giochi a favore degli approcci dotati di una componente strategica più matura.

Partite a numero fisso di mani. Il terzo gruppo di esperimenti, basato su incontri composti da un numero fisso di mani (1000), ha permesso di valutare gli algoritmi in condizioni in cui la varianza risulta sensibilmente attenuata. In questa configurazione le differenze prestazionali tra le diverse categorie sono emerse in maniera particolarmente netta, come si evince dalla tabella 4.1.

Gli approcci double oracle, in particolare XDO e ODO, hanno nuovamente mostrato

le peggiori prestazioni, con perdite medie prossime a un piccolo buio per mano. Risultati altrettanto negativi sono stati rilevati per FSP, che non è riuscito a competere efficacemente in questo scenario. Leggermente superiori, ma comunque con valori medi negativi, si sono collocati alcuni algoritmi neurali come NXDO e NFSP, insieme all'approccio di ricerca ibrida MCCFVFP ed agli algoritmi di programmazione lineare PSRO e RMDO. In questo caso, hanno saputo sfruttare il comportamento di approcci più deboli, ma non hanno mostrato particolare solidità contro avversari equilibrati. In una zona analoga si sono posizionati anche gli algoritmi basati su rimpianto come TCFR ed ECFR, che non hanno prodotto prestazioni di rilievo in questo particolare scenario.

Risultati migliori sono stati ottenuti dagli approcci neurali DeepCFR e SDCFR, oltre che da MCRNR, che riescono a capitalizzare in modo consistente con algoritmi sfruttabili, sebbene superati da metodi euristici quali MCTS ed EHS, i quali hanno mostrato una sorprendente solidità, raggiungendo valori medi confrontabili con quelli prodotti da CFR.

All'interno del gruppo di algoritmi più performanti non sono state riscontrate differenze particolarmente marcate nei guadagni medi. CFR-BR, DDCFR ed OM-Expectimax si sono distinti come un insieme di metodi altamente sfruttanti, capaci di ottenere risultati molto elevati contro avversari non bilanciati, pur mostrando una certa vulnerabilità contro le strategie più solide.

	CFR	CFR+	CFR-BR	ECFR	TCFR	DeepCFR	SD-CFR	RCFR	DDCFR	MCCFR	FSP	NFSP	ODO	PSRO	XDO	NXDO	RMDO	EHS	MCTS	ISMCTS	MCRNR	MCCFVFP	OME	media
CFR	–	-0.22	-0.09	+0.35	-0.47	+0.08	+0.09	-0.15	-0.36	-0.35	+0.87	+0.56	+0.29	+0.08	+0.39	+0.11	-0.12	+0.02	+0.13	+0.03	-0.04	+0.12	-0.10	+0.05
CFR+	+0.26	–	+0.12	+0.38	+0.16	+0.18	+0.16	+0.17	+0.10	+0.13	+0.98	+0.25	+0.41	+0.19	+0.42	+0.18	+0.10	+0.11	+0.10	+0.17	+0.33	+0.22	+0.09	+0.23
CFR-BR	+0.04	+0.14	–	+0.12	-0.03	+0.05	+0.04	-0.01	+0.40	+0.20	+1.20	-0.11	+0.46	+0.33	+1.06	+0.27	+0.24	+0.30	+0.25	-0.01	+0.18	+0.30	-0.16	+0.24
ECFR	-0.23	-0.31	-0.10	–	-0.34	-0.30	-0.41	+0.04	-0.19	-0.50	+0.41	+0.04	+0.31	+0.15	+0.39	+0.02	-0.14	-0.36	-0.17	-0.21	-0.20	+0.03	-0.22	-0.10
TCFR	+0.48	-0.11	+0.07	+0.44	–	-0.20	-0.16	-0.07	-0.01	-0.80	+0.46	-0.20	+0.48	+0.09	+1.02	-0.03	+0.46	-0.08	-0.40	-0.66	-0.18	+0.06	-0.25	+0.01
DeepCFR	-0.02	-0.13	-0.01	+0.39	+0.22	–	-0.04	+0.10	-0.49	-0.28	+1.00	-0.18	+0.51	+0.08	+1.98	+0.05	+0.60	-0.19	-0.10	-0.09	-0.11	+0.05	+0.02	+0.15
SD-CFR	+0.05	-0.08	-0.03	+0.47	+0.17	+0.09	–	+0.14	-0.21	-0.22	+0.91	+0.10	+0.40	+0.12	+1.21	+0.13	+0.67	-0.02	-0.11	-0.01	-0.15	+0.14	-0.09	+0.17
RCFR	+0.19	-0.14	+0.10	-0.04	+0.11	-0.07	-0.13	–	-0.04	-0.25	+0.24	+0.03	+0.29	+0.15	+0.97	+0.08	+0.19	-0.20	+0.05	-0.18	+0.13	+0.24	-0.10	+0.07
DDCFR	+0.38	-0.07	-0.35	+0.23	+0.07	+0.52	+0.26	+0.11	–	+0.05	+0.88	+0.34	+0.25	+0.03	+0.31	+0.09	+0.19	+0.22	+0.14	+0.01	+0.15	+0.16	+0.06	+0.18
MCCFR	+0.42	-0.11	-0.18	+0.60	+0.89	+0.26	+0.32	+0.27	-0.05	–	+0.95	-0.10	+0.35	+0.09	+0.88	+0.23	+1.14	+0.17	+0.10	+0.13	+0.05	+1.25	+0.04	+0.35
FSP	-0.77	-0.85	-1.11	-0.32	-0.42	-0.91	-0.81	-0.21	-0.73	-0.71	–	-0.27	+0.05	-0.49	-0.08	-0.27	-0.44	-0.87	-0.80	-1.96	-0.93	-0.51	-1.23	-0.67
NFSP	-0.45	-0.21	+0.16	-0.24	+0.28	+0.20	-0.10	-0.02	-0.31	+0.15	+0.32	–	+0.26	+0.16	+0.57	-0.05	+0.08	-0.15	+0.36	-0.05	-0.21	-0.15	-0.27	+0.01
ODO	-0.28	-0.40	-0.44	-0.29	-0.45	-0.52	-0.30	-0.27	-0.22	-0.30	-0.01	-0.20	–	-0.12	+0.08	-0.22	-0.31	-0.40	-0.50	-0.47	-0.39	-0.17	-0.59	-0.31
PSRO	-0.03	-0.13	-0.30	-0.12	-0.07	-0.11	-0.09	-0.14	-0.01	-0.08	+0.61	-0.12	+0.16	–	+0.22	-0.06	+0.11	-0.15	-0.05	-0.19	-0.06	-0.04	-0.22	-0.04
XDO	-0.32	-0.41	-1.02	-0.31	-1.00	-1.89	-1.14	-0.87	-0.30	-0.81	+0.13	-0.51	-0.07	-0.19	–	-0.47	-1.10	-0.31	-0.42	-0.50	-0.83	-0.58	-1.04	-0.63
NXDO	-0.10	-0.18	-0.17	+0.02	+0.05	-0.04	-0.07	-0.03	-0.09	-0.19	+0.29	+0.12	+0.25	+0.07	+0.52	–	+0.13	-0.19	-0.10	-0.17	-0.21	-0.19	-0.29	-0.03
RMDO	+0.09	-0.08	-0.24	+0.18	-0.40	-0.54	-0.64	-0.15	-0.18	-1.12	+0.49	-0.06	+0.36	-0.09	+1.17	-0.10	–	-0.30	-0.21	-0.28	+0.14	+0.16	-0.15	-0.09
EHS	+0.03	-0.07	-0.29	+0.39	+0.10	+0.21	+0.15	+0.22	-0.19	-0.13	+0.87	+0.15	+0.37	+0.16	+0.32	+0.29	+0.26	–	+0.13	-0.14	+0.02	+0.12	-0.10	+0.13
MCTS	-0.12	-0.11	-0.23	+0.20	+0.41	+0.14	+0.05	-0.01	-0.10	-0.06	+0.90	-0.36	+0.48	+0.04	+0.49	+0.10	+0.17	-0.12	–	-0.03	+0.28	+0.30	-0.18	+0.10
ISMCTS	+0.02	-0.10	+0.10	+0.25	+0.70	+0.09	+0.16	+0.21	+0.01	-0.04	+1.90	+0.15	+0.57	+0.17	+0.54	+0.22	+0.32	+0.12	+0.06	–	+0.25	+0.40	+0.08	+0.28
MCRNR	+0.05	-0.32	-0.17	+0.24	+0.16	+0.11	+0.19	-0.13	-0.12	-0.01	+0.95	+0.27	+0.44	+0.16	+0.86	+0.19	-0.17	+0.02	-0.23	-0.26	–	+0.12	-0.22	+0.10
MCCFVFP	-0.06	-0.20	-0.29	+0.01	-0.07	-0.05	-0.08	-0.21	-0.15	-1.15	+0.53	+0.18	+0.19	+0.04	+0.50	+0.23	-0.12	-0.09	-0.24	-0.41	-0.12	–	-0.23	-0.08
OME	+0.12	-0.05	-0.10	+0.29	+0.26	+0.02	+0.17	+0.11	-0.08	-0.09	+1.26	+0.31	+0.62	+0.23	+1.00	+0.36	+0.16	+0.10	+0.28	-0.04	+0.21	+0.20	–	+0.24
media	+0.01	+0.19	+0.21	-0.15	-0.02	+0.12	+0.11	+0.04	+0.15	+0.30	-0.73	-0.02	-0.34	-0.07	-0.67	-0.06	-0.11	+0.11	+0.08	+0.19	+0.12	-0.03	+0.15	–

Tabella 4.1: Confronto tra tutti gli algoritmi analizzati. Il valore numerico all'interno delle singole celle va considerato come il guadagno medio per mano (in grandi bui, quindi bb/h) dell'algoritmo indicato nella riga, come giocatore iniziale, contro quello indicato nella colonna, calcolato su 1000 mani giocate. Valori positivi indicano un guadagno, mentre valori negativi indicano una perdita. Dai risultati si comprende come il giocatore iniziale, nell'HULHE, abbia un leggero vantaggio strategico.

Un comportamento più equilibrato è stato osservato per RCFR, caratterizzato da una

buona robustezza contro una vasta gamma di avversari. In queste condizioni, i guadagni medi contro le fasce inferiori di algoritmi si sono attestati tra 0.2 e 0.4 grandi bui per mano, mentre i confronti diretti con algoritmi di pari livello hanno evidenziato risultati tendenzialmente neutri o oscillanti tra -0.15 e +0.1 bb per mano. Nonostante non sia tra i migliori in termini di guadagno medio per mani, RCFR tende a mantenere le perdite contenute anche contro avversari molto forti.

Gli algoritmi complessivamente migliori sono risultati ISMCTS, CFR+ e MCCFR. Pur non essendo sempre i più efficaci in termini di puro sfruttamento, essi hanno mostrato la capacità più elevata di minimizzare le perdite, mantenendo una strategia stabile e difficilmente sfruttabile nel lungo periodo. Tali risultati confermano, ancora una volta, l'efficacia degli approcci basati sulla minimizzazione del rimpianto nel contesto del poker a informazione imperfetta.

4.3 Discussioni ed analisi

Motivazioni delle prestazioni. L'andamento complessivo dei risultati può essere interpretato alla luce delle caratteristiche strutturali che contraddistinguono le diverse famiglie di algoritmi impiegate. Il comportamento osservato non è sorprendente: in larga misura riflette i limiti teorici, le assunzioni computazionali e le condizioni operative per cui ciascun approccio è stato originariamente sviluppato.

Un primo gruppo particolarmente problematico è costituito dagli algoritmi fondati sulla programmazione lineare e, più in generale, sui metodi di tipo *double oracle*, quali XDO, ODO, RMDO e PSRO. Questi algoritmi nascono per giochi con alberi decisionali sensibilmente più piccoli e più regolari di HULHE; nel dominio considerato, la loro struttura risulta intrinsecamente inadeguata. La loro efficienza dipende infatti dalla capacità di calcolare risposte ottimali o quasi ottimali all'interno di uno spazio altamente ridotto. Tuttavia, in un gioco della complessità di HULHE, il calcolo di una risposta ottimale esatta non è praticabile con tempi ragionevoli, e ciò obbliga all'utilizzo di approssimazioni severe che degradano drasticamente la qualità delle strategie generate. A parità di tempo di addestramento, questi algoritmi convergono molto più lentamente rispetto a metodi più moderni e risultano altamente sfruttabili. Anche le varianti più sofisticate, come PSRO e RMDO, pur introducendo

miglioramenti nella generazione delle risposte, non riescono a compensare completamente il costo computazionale crescente e l'imprecisione inevitabile delle approssimazioni adottate.

FSP rappresenta un caso peculiare. Pur avendo una convergenza teorica garantita, tale risultato è valido solo in condizioni di budget computazionale elevato o illimitato. In uno scenario con tempo di addestramento fisso e senza possibilità di utilizzare astrazioni particolarmente raffinate, FSP non riesce a raggiungere la profondità strategica necessaria per risultare competitivo. Il suo affidamento su risposte ottimali accurate rende il calcolo ancora più oneroso, penalizzandolo rispetto ad algoritmi che, pur meno teoricamente eleganti, sono in grado di produrre valutazioni e aggiornamenti strategici molto più rapidamente.

Un secondo gruppo è quello degli algoritmi basati su reti neurali. Metodi come DeepCFR, NFSP, SDCFR e NXDO sono stati espressamente progettati per operare in giochi talmente vasti da rendere impraticabile l'esplorazione completa dell'albero. La capacità di generalizzazione delle reti neurali rappresenta, in questi contesti, un compromesso efficace tra precisione e scalabilità. Tuttavia, nel contesto specifico di HULHE e in presenza di astrazione, tali algoritmi soffrono di una duplice debolezza: da un lato, la qualità dell'apprendimento è limitata dal ridotto tempo di addestramento disponibile; dall'altro, la generalizzazione introduce distorsioni che solo modelli particolarmente espressivi e ben addestrati riuscirebbero a correggere. Di conseguenza, pur mostrando prestazioni ragionevoli, questi metodi risultano sfruttabili da algoritmi di ricerca o da varianti di CFR più rapide e stabili.

La terza categoria è costituita dagli algoritmi di ricerca, che hanno mostrato una robustezza sorprendente nel contesto sperimentale adottato. Approcci come MCTS, ISMCTS ed Expectimax, pur non mirando all'equilibrio e privi di garanzie formali nel lungo periodo, operano direttamente sul gioco reale senza alcun tipo di astrazione. Questa caratteristica, unita al fatto che non richiedono addestramento, li rende estremamente competitivi quando gli algoritmi avversari sono penalizzati dal tempo di apprendimento o dalla perdita di informazione. In particolare, ISMCTS e Expectimax si sono distinti nella maggior parte degli scenari, mostrando come l'immediatezza valutativa e la capacità di sfruttare pattern locali possano rappresentare un vantaggio significativo. È tuttavia importante sottolineare che questa forza è situazionale: qualora si disponesse di un tempo di addestramento molto più esteso o di astrazioni estremamente precise, gli algoritmi di equilibrio e quelli neurali tenderebbero a superarli nel lungo periodo.

Infine, i metodi basati sulla minimizzazione del rimpianto rappresentano il gruppo più

coerente e prevedibile nelle loro prestazioni. Regret Matching, CFR, MCCFR, RCFR e le relative varianti si distinguono per l'equilibrio tra stabilità, robustezza e adattabilità. Il loro comportamento rispecchia ampiamente le aspettative teoriche: in media, convergono verso strategie solide e difficilmente sfruttabili. Le differenze interne alla famiglia dipendono principalmente dalla velocità con cui riescono a ridurre il rumore e a stabilizzare le stime. Varianti come TCFR ed ECFR si sono rivelate meno efficaci poiché particolarmente sensibili agli errori di astrazione, che ne compromettono la capacità di aggiornare correttamente il rimpianto. Al contrario, CFR+ e DDCFR, progettati per controllare più efficacemente la variabilità degli aggiornamenti e per accelerare la convergenza, hanno mostrato prestazioni eccellenti. CFR+, in particolare, si conferma ancora una volta come uno degli algoritmi più affidabili, e la sua capacità di risolvere versioni astratte del gioco trova piena corrispondenza nei risultati sperimentali.

In sintesi, le prestazioni osservate riflettono il delicato equilibrio tra complessità computazionale, stabilità delle stime, sensibilità all'astrazione e capacità di sfruttamento dell'avversario. Gli algoritmi più efficaci sono quelli che riescono a coniugare rapidità di adattamento, tolleranza agli errori introdotti dalla modellizzazione e robustezza nel medio-lungo periodo. Le differenze emerse non costituiscono anomalie, ma confermano la stretta relazione tra le assunzioni teoriche alla base degli algoritmi e la loro effettiva capacità di competere all'interno di un gioco complesso e ad alta varianza come l'Heads-Up Limit Hold'em.

Il ruolo della varianza. I risultati sperimentali devono essere interpretati anche alla luce dell'elevata varianza che caratterizza il gioco di HULHE. Come riportato nella letteratura di riferimento [12], la deviazione standard di una singola mano in questo dominio è pari a circa 5 big blind per gioco (bb/g). Tale valore, che rappresenta la fluttuazione tipica del risultato di una singola mano indipendente, ha un impatto diretto sulla significatività statistica delle prestazioni osservate, rendendo difficile distinguere differenze reali da oscillazioni casuali su orizzonti temporali brevi.

Per comprendere meglio l'importanza della varianza, si consideri che la deviazione standard dopo n mani indipendenti decresce come $\sigma_n = \frac{5}{\sqrt{n}}$ bb/g. Nei contesti sperimentali considerati, le configurazioni con capitale relativamente ridotto e numero medio di mani dell'ordine di qualche centinaio risultano dunque intrinsecamente soggette a oscillazioni ampie. Ad esempio, con 300 mani, la deviazione standard rimane intorno a $\sigma_{300} \approx 0.29$ bb/g, un valore comparabile alle differenze prestazionali osservate tra diversi algoritmi nella

fascia media della classifica. È quindi naturale che, in tali configurazioni, metodi exploitativi o privi di garanzie teoriche, quali MCTS, ISMCTS o Expectimax, possano occasionalmente ottenere risultati superiori contro avversari più solidi, semplicemente grazie alle oscillazioni casuali del gioco.

Lo studio [12] offre inoltre un'analisi quantitativa della difficoltà di stabilire differenze statisticamente significative anche su orizzonti molto lunghi. Persino un match di 100,000 mani presenta un intervallo di confidenza al 95% pari a circa 31 milli-big blind per gioco (mbb/g), una soglia superiore alle differenze di prestazione tra molte strategie storicamente sviluppate. Questo dato evidenzia quanto sia complesso discriminare con certezza tra algoritmi vicini in termini di qualità strategica, anche quando essi giocano un numero elevato di incontri.

Applicando questi ragionamenti alle configurazioni sperimentali del presente lavoro, risulta evidente come le diverse configurazioni presentino livelli molto differenti di affidabilità statistica. Le partite con capitale iniziale ridotto, pur essendo utili per individuare tendenze qualitative, soffrono di una componente aleatoria predominante. Le configurazioni con capitale aumentato riducono parzialmente la varianza, ma anch'esse non permettono una distinzione netta tra strategie di qualità vicine. Solo il match a 1000 mani consente una valutazione più stabile, poiché la deviazione standard scende a circa 0.158 bb/g, abbastanza piccola da permettere il riconoscimento di differenze prestazionali sostanziali tra algoritmi con comportamento profondamente diverso.

In sintesi, la struttura intrinseca di HULHE rende il gioco altamente sensibile alla varianza, e di conseguenza impone una notevole cautela nell'interpretazione dei risultati sperimentali. A seconda della configurazione, possono essere necessarie decine di migliaia di mani per ridurre in maniera significativa l'incertezza statistica, e persino orizzonti di gioco molto ampi potrebbero non essere sufficienti per distinguere strategie simili. Questo aspetto spiega, almeno in parte, perché algoritmi estremamente solidi nel lungo termine possano occasionalmente essere superati da tecniche più exploitative nei contesti sperimentali più brevi, e conferma la centralità di una corretta gestione della varianza nelle valutazioni empiriche di algoritmi per giochi a informazione imperfetta.

Capitolo 5

Conclusioni

Nel suo insieme, questa tesi ha avuto l'obiettivo di esplorare il poker *Heads-Up Limit Hold'em* come caso di studio emblematico per l'analisi dei giochi strategici a informazione imperfetta, mettendo in relazione il comportamento degli algoritmi sviluppati con i fondamenti teorici che ne regolano la struttura. Attraverso una panoramica iniziale della teoria dei giochi, sono stati introdotti i concetti che tradizionalmente guidano lo studio delle interazioni competitive tra agenti razionali, tra cui l'equilibrio di Nash, la sfruttabilità e i metodi per approssimare strategie ottimali in domini complessi. Questi strumenti teorici hanno costituito la base per comprendere sia le difficoltà intrinseche nella risoluzione di giochi di grande scala sia le principali famiglie di algoritmi che permettono oggi di affrontarli in modo efficiente.

Il lavoro ha quindi approfondito le tecniche algoritmiche più rilevanti per la risoluzione di giochi sequenziali a informazione incompleta, in particolare quelle basate sulla minimizzazione del rimpianto e sul calcolo iterativo di strategie approssimate. Questi metodi, sviluppati originariamente per superare le limitazioni dei tradizionali approcci basati sulla programmazione lineare, hanno rivoluzionato il modo in cui la comunità scientifica affronta giochi come il poker, rendendo possibile la gestione di spazi di stati estremamente ampi e la ricerca di strategie quasi ottimali in tempi praticabili. Alla luce di tali progressi, il poker HULHE rappresenta dunque non soltanto un dominio applicativo di interesse autonomo, ma anche un contesto fondamentale per valutare le prestazioni, l'adattività e la robustezza degli algoritmi progettati per competere in ambienti incerti e parzialmente osservabili.

Inoltre, HULHE rappresenta il primo gioco competitivo non banale a informazione imperfetta giocato da esseri umani ad essere stato essenzialmente debolmente risolto. Ciononostante, ci si potrebbe chiedere quale sia il significato ultimo della risoluzione di questa categoria di giochi, di cui il poker fa parte.

Le innovazioni alla base di questo risultato rappresentano progressi algoritmici generali che rendono il ragionamento basato sulla teoria dei giochi, in modelli su larga scala di qualsiasi tipo, più trattabile. E, sebbene il contesto possa apparire ludico, la teoria dei giochi è sempre stata concepita come uno strumento dotato di implicazioni profonde, come dimostrano i suoi effetti precoci sulla politica della Guerra Fredda [63]. Più recentemente, si è assistito inoltre a un aumento significativo di applicazioni della teoria dei giochi nel settore della sicurezza, includendo l'implementazione di sistemi per i controlli aeroportuali, la programmazione dei voli dei marshal e la pianificazione di attività di pattugliamento della guardia costiera [80]. Gli algoritmi basati sulla minimizzazione del rimpianto controfattuale (CFR), della stessa famiglia di quelli discussi nel presente lavoro, sono stati persino oggetto di studio per poter essere utilizzati in processi decisionali, in cui non vi è alcun avversario apparente, con potenziali ricadute nell'ambito della diagnostica e dell'assistenza medica [23].

Poiché i contesti decisionali della vita reale sono fortemente caratterizzati da incertezza e informazioni mancanti, sono necessari progressi algoritmici, analoghi a quelli utilizzati per risolvere il poker, per guidare le applicazioni future. Tuttavia, risulta importante riportare una celebre osservazione attribuita ad Alan Turing a proposito delle proprie ricerche sui giochi: "Non sarebbe onesto, da parte nostra, nascondere il fatto che il motivo principale che ha spinto il lavoro è stato il puro divertimento della cosa" [62, 82].

Bibliografia

[1] L. V. Allis. Searching for solutions in games and artificial intelligence. *PhD thesis, Vrije Universiteit Amsterdam, The Netherlands*, 1994.

[2] Holger Baier and Paul D. Drake. The power of forgetting: Improving the last-good-reply policy in monte carlo go. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(4):303–309, 2010. doi: 10.1109/TCIAIG.2010.2097630.

[3] David Balduzzi, Karl Tuyls, Julien Perolat, and Thore Graepel. Re-evaluating evaluation, 2018. URL `https://arxiv.org/abs/1806.02643`.

[4] A. Banerjee, S. Merugu, I. S. Dhillon, and J. Ghosh. Clustering with bregman divergences. *Journal of Machine Learning Research*, 6(Oct):1705–1749, 2005.

[5] Ariyan Bighashdel, Yongzhao Wang, Stephen McAleer, Rahul Savani, and Frans A. Oliehoek. Policy space response oracles: A survey, 2024. URL `https://arxiv.org/abs/2403.02227`.

[6] D. Billings, N. Burch, A. Davidson, R.C. Holte, J. Schaeffer, T. Schauenberg, and D. Szafron. Approximating game-theoretic optimal strategies for full-scale poker. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 661–668, 2003.

[7] Darse Billings, Denis Papp, Jonathan Schaeffer, and Duane Szafron. Opponent modeling in poker. pages 493–499, 01 1998.

[8] Darse Billings, Aaron Davidson, Terence Schauenberg, Neil Burch, Michael Bowling, Robert Holte, Jonathan Schaeffer, and Duane Szafron. Game-tree search with adaptation in stochastic imperfect-information games. In H. Jaap van den Herik, Yngvi Björnsson, and Nathan S. Netanyahu, editors, *Computers and Games*, pages 21–34, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-32489-8.

[9] R. Bjarnason, A. Fern, and P. Tadepalli. Lower bounding klondike solitaire with monte-carlo planning. In *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS)*, pages 26–33, Thessaloniki, Greece, 2009.

[10] E. Borel and J. Ville. *Applications de la théorie des probabilités aux jeux de hasard*. Gauthier-Villars, 1938.

[11] M. Bowling and M. Veloso. Rational and convergent learning in stochastic games. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1021–1026, 2001.

[12] M. Bowling, N. Burch, M. Johanson, and

O. Tammelin. Heads-up limit hold'em poker is solved. *Science*, 347(6218):145–149, 2015.

[13] M. Bowling, N. Burch, M. Johanson, and O. Tammelin. Heads-up limit hold'em poker is solved: Supplementary online material. Online, January 2015.

[14] J. Bronowski. The ascent of man. Documentary, Episode 13, 1973.

[15] G. W. Brown. Iterative solution of games by fictitious play. 13(1):374–376, 1951.

[16] N. Brown and T. Sandholm. Solving imperfect-information games via discounted regret minimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1829–1836, 2019.

[17] Noam Brown, Adam Lerer, Sam Gross, and Tuomas Sandholm. Deep counterfactual regret minimization. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 793–802. PMLR, 09–15 Jun 2019. URL `https://proceedings.mlr.press/v97/brown19b.html`.

[18] Cameron B. Browne, Edward Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43, 2012. doi: 10.1109/TCIAIG.2012.2186810.

[19] M. Campbell, A. J. Jr. Hoane, and F. Hsu. Deep blue. *Artificial Intelligence*, 134: 57–83, 2002.

[20] Nicolo Cesa-Bianchi and Gábor Lugosi. *Prediction, learning, and games*. Cambridge University Press, 2006.

[21] G. M. J.-B. Chaslot, M. H. M. Winands, H. J. van den Herik, J. W. H. M. Uiterwijk, and B. Bouzy. Progressive strategies for monte-carlo tree search. *New Mathematics and Natural Computation*, 4(3): 343–357, 2008.

[22] Guillaume Chaslot, Sander Bakkes, Istvan Szita, and Pieter Spronck. Monte-carlo tree search: A new framework for game ai. 01 2008.

[23] K. Chen and M. Bowling. Tractable objectives for robust policy optimization. In *Advances in Neural Information Processing Systems 25 (NIPS)*, pages 2078–2086, 2012.

[24] Peter Cowling, Edward Powley, and Daniel Whitehouse. Information set monte carlo tree search. *IEEE Transactions on Computational Intelligence and Ai in Games*, 4:120–143, 06 2012. doi: 10.1109/TCIAIG.2012.2200894.

[25] A. Davidson. Opponent modeling in poker: Learning and acting in a hostile and uncertain environment. Master's thesis, University of Alberta, 2002.

[26] Le Cong Dinh, Yaodong Yang, Stephen

McAleer, Zheng Tian, Nicolas Perez Nieves, Oliver Slumbers, David Henry Mguni, Haitham Bou Ammar, and Jun Wang. Online double oracle, 2023. URL https://arxiv.org/abs/2103.07780.

[27] Ryan D'Orazio, Dustin Morrill, James R. Wright, and Michael Bowling. Alternative function approximation parameterizations for solving games: An analysis of f-regression counterfactual regret minimization, 2020. URL https://arxiv.org/abs/1912.02967.

[28] D. Ferrucci. Introduction to "this is watson". *IBM Journal of Research and Development*, 56(3.4):1:1–1:15, 2012.

[29] I. Frank and D. Basin. Search in games with incomplete information: A case study using bridge card play. *Artificial Intelligence*, 100(1-2):87–123, 1998.

[30] Yoav Freund and Robert E. Schapire. Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, 29(1-2):79–103, 1999.

[31] D. Fudenberg and D. K. Levine. *The theory of learning in games*, volume 2. MIT press, 1998.

[32] Richard G. Gibson, Neil Burch, Marc Lanctot, and Duane Szafron. Efficient monte carlo counterfactual regret minimization in games with many player actions. In *Neural Information Processing Systems*, 2012. URL https://api.semanticscholar.org/CorpusID:1583651.

[33] M. L. Ginsberg. Gib: Imperfect information in a computationally challenging game. *Journal of Artificial Intelligence Research*, 14:303–358, 2001.

[34] Amy Greenwald, Jiacui Li, Eric Sodomka, and Michael L. Littman. Solving for best responses in extensive-form games using reinforcement learning methods. In *The 1st Multidisciplinary Conference on Reinforcement Learning and Decision Making (RLDM)*, 2013.

[35] Sergiu Hart and Andreu Mas-Colell. A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, 68 (5):1127–1150, 2000. ISSN 00129682, 14680262.

[36] J. Heinrich and D. Silver. Deep reinforcement learning from self-play in imperfect-information games. *arXiv preprint arXiv:1603.01121*, 2016.

[37] Johannes Heinrich, Marc Lanctot, and David Silver. Fictitious self-play in extensive-form games. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 805–813. JMLR.org, 2015.

[38] Josef Hofbauer and William H. Sandholm. On the global convergence of stochastic fictitious play. *Econometrica*, 70(6): 2265–2294, 2002.

[39] Eric Jackson. Targeted counterfactual regret minimization. In *The AAAI-17 Workshop on Computer Poker and Imperfect*

Information Games, page 6, 2017.

[40] P. C. Jansen. *Using Knowledge About the Opponent in Game-Tree Search*. PhD thesis, Carnegie-Mellon University, 1992.

[41] M. Johanson. Robust strategies and counter-strategies: Building a champion level computer poker player. Master's thesis, University of Alberta, 2007.

[42] M. Johanson and M. Bowling. Data-biased robust counter strategies. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 264–271, 2009.

[43] M. Johanson, M. Zinkevich, and M. Bowling. Computing robust counter-strategies. In *Advances in Neural Information Processing Systems 20 (NIPS)*, pages 721–728, 2008.

[44] M. Johanson, N. Bard, N. Burch, and M. Bowling. Finding optimal abstract strategies in extensive-form games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 12345–12352, 2021.

[45] P. R. Jordan, L. J. Schvartzman, and M. P. Wellman. Strategy exploration in empirical games. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1297–1304, 2010.

[46] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[47] D. Koller and N. Megiddo. The complexity of two-person zero-sum games in extensive form. *Games and Economic Behavior*, 4 (4):528–552, 1992.

[48] D. Koller and A. Pfeffer. Representations and solutions for game-theoretic problems. *Artificial Intelligence*, 94:167–215, 1997.

[49] D. Koller, N. Megiddo, and B. von Stengel. Efficient computation of equilibria for extensive two-person games. *Games and Economic Behavior*, 14(2):247–259, 1996.

[50] Theodore J. Lambert III, Marina A. Epelman, and Robert L. Smith. A fictitious play approach to large-scale optimization. *Operations Research*, 53(3):477–489, 2005.

[51] Marc Lanctot, Kevin Waugh, Martin Zinkevich, and Michael Bowling. Monte carlo sampling for regret minimization in extensive games. In *Advances in Neural Information Processing Systems 22 (NIPS)*, pages 1078–1086, 2008.

[52] Marc Lanctot, Vinicius Zambaldi, Audrunas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Perolat, David Silver, and Thore Graepel. A unified game-theoretic approach to multiagent reinforcement learning, 2017. URL `https://arxiv.org/abs/1711.00832`.

[53] C.-S. Lee, M.-H. Wang, G. M. J.-B. Chaslot, J.-B. Hoock, A. Rimmel, O. Teytaud, S.-R. Tsai, S.-C. Hsu, and T.-P. Hong. The computational intelligence of mogo revealed in taiwan's computer go tournaments. *IEEE Transactions on Computational Intelligence and AI in Games*, 1(1):73–89, 2009.

[54] C.-S. Lee, M. Müller, and O. Teytaud. Guest editorial: Special issue on monte carlo techniques and computer go. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(4):225–228, Dec 2010.

[55] David S. Leslie and Edmund J. Collins. Generalised weakened fictitious play. *Games and Economic Behavior*, 56(2):285–298, 2006.

[56] Huale Li, Xuan Wang, Shuhan Qi, Jiajia Zhang, Yang Liu, Yulin Wu, and Fengwei Jia. Solving imperfect-information games via exponential counterfactual regret minimization, 2020. URL `https://arxiv.org/abs/2008.02679`.

[57] Stephen McAleer, John Lanier, Roy Fox, and Pierre Baldi. Pipeline psro: A scalable approach for finding approximate nash equilibria in large games, 2021. URL `https://arxiv.org/abs/2006.08555`.

[58] Stephen McAleer, John Lanier, Kevin Wang, Pierre Baldi, and Roy Fox. Xdo: A double oracle algorithm for extensive-form games, 2022. URL `https://arxiv.org/abs/2103.06426`.

[59] H. Brendan McMahan and Geoffrey J. Gordon. A fast bundle-based anytime algorithm for poker and other convex games. In *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 323–330, 2007.

[60] H. Brendan McMahan, Geoffrey J. Gordon, and Avrim Blum. Planning in the presence of cost functions controlled by an adversary. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 536–543, 2003.

[61] D. Michie and R. A. Chambers. Boxes: An experiment in adaptive control. In E. Dale and D. Michie, editors, *Machine Intelligence 2*, pages 137–152. Oliver and Boyd, 1968.

[62] P. Mirowski. What were von neumann and morgenstern trying to accomplish? In Weintraub, editor, *Toward a History of Game Theory*, pages 113–147. Duke University Press, 1992. Mirowski cita Turing come autore del paragrafo contenente questa frase. Il paragrafo appare in [85], all'interno di un capitolo avente Turing come uno dei tre contributori. Quali parti del capitolo siano opera di quale contributore, in particolare il materiale introduttivo contenente questa citazione, non è reso esplicito.

[63] O. Morgenstern. The cold war is cold poker. *N. Y. Times Mag.*, pages 21–22, 1961.

[64] M. J. Osborne and A. Rubinstein. *A Course in Game Theory*. The MIT Press, 1994.

[65] Marc Ponsen, Steven Jong, and Marc Lanctot. Computing approximate nash equilibria and robust best-responses using sampling. *J. Artif. Intell. Res. (JAIR)*, 42:

575–605, 09 2011. doi: 10.1613/jair.3402.

[66] Ju Qi, Falin Hei, Ting Feng, Dengbing Yi, Zhemei Fang, and Yunfeng Luo. Accelerating nash equilibrium convergence in monte carlo settings through counterfactual value based fictitious play, 2024. URL https://arxiv.org/abs/2309.03084.

[67] A. L. Reibman and B. W. Ballard. Non-minimax search strategies for use against fallible opponents. In *Proceedings of the AAAI National Conference*, pages 338–342, 1983.

[68] Nick Abou Risk and Duane Szafron. Using counterfactual regret minimization to create competitive multiplayer poker agents. Technical report, University of Alberta, Department of Computing Science, 2010.

[69] I. V. Romanovskii. Reduction of a game with complete memory to a matrix game. *Soviet Mathematics*, 3:678–681, 1962.

[70] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2nd edition, 2003.

[71] Frank C. Schadd. Monte-carlo search techniques in the modern board game thurn and taxis. Master's thesis, Maastricht University, 2009.

[72] J. Schaeffer, R. Lake, P. Lu, and M. Bryant. Chinook the world man-machine checkers champion. *AI Magazine*, 17(1):21–29, 1996.

[73] Terence Conrad Schauenberg. Opponent modelling and search in poker. Master's thesis, University of Alberta, 2006.

[74] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

[75] S. Sen and N. Arora. Learning to take risks. In *Proceedings of the AAAI-97 Workshop on Multiagent Learning*, pages 59–64, 1997.

[76] Shai Shalev-Shwartz. Online learning and online convex optimization. *Foundations and Trends in Machine Learning*, 4(2): 107–194, 2011.

[77] J. S. Shamma and G. Arslan. Dynamic fictitious play, dynamic gradient play, and distributed convergence to nash equilibria. *IEEE Transactions on Automatic Control*, 50(3):312–327, 2005.

[78] David Silver and Joel Veness. Monte-carlo planning in large pomdps. In *Advances in Neural Information Processing Systems*, pages 2164–2172, 2010.

[79] Eric Steinberger. Single deep counterfactual regret minimization, 2019. URL https://arxiv.org/abs/1901.07621.

[80] Milind Tambe. Security and game theory: Algorithms, deployed systems, lessons learned. 01 2011. doi: 10.1017/CBO9780511973031.

[81] Xiaohang Tang, Le Cong Dinh, Stephen Marcus McAleer, and Yaodong Yang. Regret-minimizing double oracle for extensive-form games, 2023. URL https://arxiv.org/abs/2304.10498.

[82] A. Turing. Digital computers applied to games. In B.V. Bowden, editor, *Faster Than Thought*, chapter 25. Pitman, 1976.

[83] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 30 (1), Mar. 2016. doi: 10.1609/aaai.v30i1.10295. URL `https://ojs.aaai.org/index.php/AAAI/article/view/10295`.

[84] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, J. Oh, C. Hesse, D. Silver, and K. Kavukcuoglu. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575:350–354, 2019.

[85] J. S. Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)*, 11(1):37–57, 1985.

[86] J. von Neumann. Zur theorie der gesellschaftsspiele. *Mathematische Annalen*, 100 (1):295–320, 1928.

[87] J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, Princeton, 2 edition, 1947.

[88] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4): 279–292, 1992.

[89] K. Waugh, D. Schnizlein, M. Bowling, and D. Szafron. Abstraction pathology in extensive games. In *Proceedings of the 8th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2009.

[90] K. Waugh, M. Zinkevich, M. Johanson, M. Kan, D. Schnizlein, and M. Bowling. A practical use of imperfect recall. In *Proceedings of the Eighth Symposium on Abstraction, Reformulation and Approximation (SARA)*, 2009.

[91] Kevin Waugh, Dustin Morrill, J. Andrew Bagnell, and Michael Bowling. Solving games with functional regret estimation, 2014. URL `https://arxiv.org/abs/1411.7974`.

[92] Hang Xu, Kai Li, Haobo Fu, QIANG FU, Junliang Xing, and Jian Cheng. Dynamic discounted counterfactual regret minimization. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=6PbvbLyqT6`.

[93] M. Zinkevich, M. Johanson, M. Bowling, and C. Piccione. Regret minimization in games with incomplete information. Technical Report TR 2007-17, University of Alberta, Department of Computing Science, 2007.