Alma Mater Studiorum
Università degli Studi di Bologna

Facoltà di Scienze Matematiche, Fisiche e Naturali
Corso di Laurea in Informatica
Materia di Tesi: Ingegneria del Software

Analisi e progettazione di un programma di gioco ad informazione imperfetta

Tesi di Laurea di:
Andrea Bolognesi

Relatore:
Chiar.mo Prof. Paolo Ciancarini

II Sessione
Anno Accademico 2002/2003

Alma Mater Studiorum
Università degli Studi di Bologna

Facoltà di Scienze Matematiche, Fisiche e Naturali
Corso di Laurea in Informatica
Materia di Tesi: Ingegneria del Software

Analisi e progettazione di un programma di gioco ad informazione imperfetta

Tesi di Laurea di:
Andrea Bolognesi

Relatore:
Chiar.mo Prof. Paolo Ciancarini

Parole chiave: Kriegspiel; Teoria dei Giochi; Giochi a somma zero; Informazione imperfetta

II Sessione
Anno Accademico 2002/2003

alla mia famiglia
e a Elena

Indice

Capitolo 1

Introduzione

Nell'Intelligenza Artificiale lo studio dell'interazione strategica tra giocatori ha solitamente ignorato quelle tipologie di giochi dette ad *informazione imperfetta*, come il Poker o il Kriegspiel.

L'idea di far giocare il computer nacque agli albori dell'informatica. Il principio fondamentale prevedeva che il computer, al suo turno, creasse una qualche parte dell'albero di gioco a partire dalla posizione corrente, ne valutasse le foglie tramite euristiche e quindi eseguisse una ricerca minimax sull'albero per determinare la mossa ottima alla radice. Questa semplice idea compone tuttora il nucleo centrale di molti programmi che giocano e viene applicata con successo in un'ampia varietà di giochi, quali gli Scacchi, la Dama, Othello, Backgammon e altri [32].

In letteratura sono riportati ben pochi programmi in grado di giocare in modo discreto a Poker o a Bridge, né si ricordano programmi in grado di giocare a Kriegspiel. Ciò accade non a caso, poiché le due categorie di giochi elencate, Scacchi e Dama da un lato, Poker e Kriegspiel dall'altro, appartengono a due classi fondamentalmente differenti: perciò le tecniche adottate per una tipologia di gioco non funzionano per l'altra [16].

La differenza principale risiede nell'informazione disponibile a chi gioca. In giochi come gli Scacchi, lo stato corrente è osservabile nel suo complesso dai giocatori; l'unica incertezza riguarda le mosse future. In giochi come il Poker invece, i giocatori hanno informazione imperfetta: hanno solo una conoscenza parziale sullo stato corrente del gioco. Ciò può creare complessi ragionamenti. Ad esempio, per il gioco del Poker un giocatore potrebbe fare la seguente considerazione: "Visto che ho due assi sul tavolo, ma l'avversario ha rilanciato, allora o sta bluffando o ha una buona mano; se continuo a rilanciare, l'avversario potrà realizzare che ho almeno un terzo asso in mano, così abbandonerà il piatto; quindi forse dovrei puntare meno, ma...". È pressoché ovvio che le tecniche classiche sono inadeguate per la soluzione di giochi

di questo tipo: nessuna variante dell'algoritmo minimax può ripercorrere il ragionamento appena descritto.

Questo vale per il Poker in cui grande importanza assume la componente aleatoria, non presente invece nel gioco del Kriegspiel. Vedremo in seguito che, anche se basato sulle stesse regole degli Scacchi, il Kriegspiel presenta una teoria completamente diversa ([9]). La natura numerica molto complessa e la mancanza di una componente casuale amplificano la necessità di giocare eseguendo ragionamenti strategici e probabilistici, rendendo così il gioco di difficile trattazione. È interessante osservare che, allo stato dell'arte, non esistono programmi in grado di giocare a Kriegspiel.

Nella *Teoria dei Giochi* di von Neumann e Morgenstern, d'altra parte, quasi l'intero studio è concentrato sui giochi ad informazione imperfetta. La teoria dei giochi trova interesse a lavorare con giochi derivati dalla "vita reale", e, più precisamente, da applicazioni economiche ed è chiaro che nella vita reale difficilmente si ha a disposizione un'informazione perfetta.

In giochi ad informazione perfetta, è facile definire la mossa ottima per ogni giocatore, in quanto in ogni stato c'è una mossa che è almeno tanto corretta quanto lo sono le altre. Nei giochi ad informazione imperfetta la situazione non è altrettanto favorevole. Nel semplice gioco della "morra cinese", ogni strategia deterministica è perdente se viene rivelata all'avversario. Intuitivamente, nei giochi in cui manca parte dell'informazione, può essere vantaggioso lasciare l'avversario all'oscuro delle proprie scelte e l'unico modo per farlo consiste nell'utilizzare *strategie probabilistiche*. Con tali strategie, l'esistenza di *strategie ottime* può essere provata anche in giochi ad informazione imperfetta. Ciò significa che teoricamente esiste una strategia probabilistica ottima per il Poker, come esiste una strategia ottima deterministica per gli Scacchi. Nel 1950, Kuhn ha infatti mostrato che, per una variante semplificata del Poker, la strategia ottima è in effetti probabilistica.

Poiché algoritmi basati sulla ricerca minimax non possono produrre strategie probabilistiche, il metodo adottato nella Teoria dei Giochi consiste nel trasformare l'albero di gioco in una matrice. Tale rappresentazione prende il nome di *forma normale* o *forma strategica* del gioco. Differenti tecniche, ad esempio la programmazione lineare, possono essere quindi applicate alla matrice per ottenere strategie ottime. Sfortunatamente, la matrice è esponenziale sulla dimensione dell'albero di gioco e questo rende il metodo impraticabile per molti giochi complessi.

Un recente lavoro di Koller, Megiddo e von Stengel, propone un approccio alternativo per lavorare con giochi ad informazione imperfetta, in cui si definisce una nuova rappresentazione, chiamata *sequence form*, lineare sulla dimensione dell'albero di gioco e applicabile ad algoritmi classici che posso-

no essere adattati alla nuova rappresentazione per trovare le strategie ottime ([12]).

Nella sezione 3.2, analizzerò una soluzione alternativa nel caso particolare del Kriegspiel, che sfrutta alcune euristiche e utili accorgimenti per semplificare l'albero delle mosse di gioco e rende quindi possibile l'utilizzo di un algoritmo di ricerca. Proporrò quindi l'utilizzo di una funzione di valutazione, adottando, per la prima volta in letteratura, un tale metodo nella soluzione di un gioco ad informazione imperfetta.

1.1 Teoria dei Giochi

1.1.1 Storia

La Teoria dei Giochi è emersa recentemente offrendo una efficace proposta alternativa ai metodi convenzionali nell'analisi dell'economia. Sebbene ci siano stati numerosi predecessori illustri, la formulazione della teoria dei giochi intesa come parte della teoria economica fu proposta da John von Neumann e Oskar Morgenstern nel famoso libro *Theory of Games and Economic Behavior* (Princeton University Press, 1944)[38].

Se confrontiamo questa data con quella della nascita della geometria o della fisica, ne concludiamo che stiamo parlando di una *scienza bambina*, così come la chiama Roberto Lucchetti in [22]. Ha dunque tutti i pregi ed anche, forse, i pochi difetti di una giovane creatura. In particolare, dice Lucchetti, essa ha in sè interessi e potenzialità straordinari, che risiedono in un suo aspetto bifronte: da una parte ha applicazioni notevoli, interessanti, nuove: è una disciplina pratica. Dall'altra è invece capace di proporre, in maniera originale, questioni profonde, che si legano ad altre discipline e non solo, ma anche agli aspetti più importanti del pensiero umano.

Lo scopo principale della Teoria dei Giochi è quello di considerare situazioni dove agenti siano in grado di prendere decisioni, non come reazioni a cause esogene (*variabili morte*), ma come reazioni strategiche ad azioni di altri agenti (*variabili vive*). Ad un agente viene sottoposto un insieme di mosse che può giocare e che formeranno una strategia, la migliore risposta all'ambiente che lo circonda.

Le strategie possono essere pure (ossia viene giocata una particolare mossa) o miste (il gioco segue una distribuzione di probabilità). Si dice che viene raggiunto un Equilibrio di Nash qualora ogni azione di un agente generi una reazione in ogni altro agente che, a sua volta, origina la stessa azione iniziale. In altri termini, ogni giocatore concorda sulle mosse migliori per ciascun giocatore.

Ciò che gli economisti chiamano teoria dei giochi, gli psicologi chiamano teoria delle situazioni sociali, il che descrive accuratamente su cosa verte l'applicazione matematica di tale teoria. Ci sono due filoni fondamentali che distinguono tra giochi non cooperativi e giochi cooperativi [20]. La teoria cooperativa studia il formarsi di coalizioni, che possono essere di vantaggio ai singoli componenti; d'altra parte la teoria non cooperativa cerca invece di spiegare i meccanismi delle decisioni dei singoli, sulla base di ragionamenti individuali, senza alleanze fra individui. Si deve soprattutto a von Neumann l'idea di analizzare i giochi studiando il nascere delle coalizioni fra individui, mentre è Nash che ha dato impulso alla teoria non cooperativa [22].

John Nash sostenne che tutti i giochi cooperativi potevano essere ritradotti in una forma non cooperativa, tale posizione è oggi conosciuta come "Nash Programme".

La letteratura dei giochi non cooperativi suggerisce un'altra divisione fra giochi, che riguarda la loro descrizione in forma matematica. Si parla allora di giochi in forma estesa (dinamica) e di giochi in forma normale o strategica (statica). Nella prima categoria si ha la descrizione del gioco sotto forma di albero: si tratta di costruire un grafo che, partendo dalla radice, descrive il gioco mossa per mossa, fino ad arrivare a presentare tutte le situazioni finali, conseguenti ad una data serie di mosse. Il tutto fornisce un modo estremamente efficace ed esauriente per analizzare un gioco, ma ha il difetto di essere molto complicato. La forma strategica invece di solito precisa il numero dei giocatori, lo spazio delle loro strategie e la funzione di utilità di ciascuno di essi: una funzione a valori reali, definita sul prodotto cartesiano degli spazi di strategie. Il punto fondamentale è forse nel fatto che le strategie in questa descrizione sono un dato del problema, mentre nella forma estesa abbiamo serie di mosse ed un compito delicato di chi analizza il gioco è proprio quello di dedurne da queste le strategie di ogni giocatore [22].

John von Neumann e Oskar Morgenstern (1944) introdussero il gioco in forma normale e in forma estesa, il concetto di strategie pure e miste, giochi con coalizioni e l'assiomatizzazione dell'utilità attesa, che divenne indispensabile per l'economia in presenza di incertezza. Impiegarono il concetto di soluzione minimax, proposto già da John von Neumann (1928) per risolvere semplici giochi strategici a somma zero. Nel 1950, John Nash introdusse il concetto di Equilibrio di Nash, che divenne il fulcro della Teoria dei Giochi, sebbene si consideri oggi che tale concetto fosse stato affrontato già ai tempi di Cournot (1838). Nash introdusse quindi nel 1951 il concetto di "Nash Bargaining Solution" per giochi con coalizioni.

Si aprirono così le porte per ulteriori raffinamenti dell'Equilibrio di Nash. Nel campo dei giochi non cooperativi, R. Duncan Luce e Howard Raiffa (1957) scrissero il primo libro di testo sulla Teoria dei Giochi e, in esso,

formalizzarono l'idea dell'eliminazione di strategie dominate e introdussero il concetto di Gioco Ricorsivo. H.W.Kuhn (1953) introdusse i giochi in forma estesa con informazione imperfetta, ossia giochi in cui un giocatore non conosce quali mosse sono già state giocate da altri giocatori. William Vickrey (1961) formalizzò per la prima volta il concetto di "asta". Reinhard Selten (1965) sviluppò il concetto di "Subgame Perfect Equilibrium", come un raffinamento per la soluzione di giochi in forma estesa. John C. Harsanyi (1967-8) sviluppò il concetto di "Bayesian Equilibrium" per giochi Bayesiani, ossia giochi con informazione incompleta, dove c'è incertezza sulle mosse.

Un'importante distinzione è, infatti, quella fra giochi ad informazione completa e quelli ad informazione imperfetta. Potremmo anzi dire che i secondi sono un primo passo verso una teoria più sofisticata ma anche più soddisfacente, perché propongono modelli più aderenti alla realtà. In effetti, quando studiamo un fenomeno, ad esempio economico o biologico, quasi mai tutte le informazioni sono parimenti note a tutti i protagonisti, pertanto un modello che tenga conto di questo aspetto appare più verosimile.

Anche nei giochi con coalizione si ottennero miglioramenti. Lloyd Shapley (1953) introdusse il concetto di "Valore di Shapley" e "Core", che era stato inizialmente concepito da F.Y. Edgeworth (1881) come soluzione di giochi con coalizione. Durante i primi anni '60, Robert J. Aumann e Martin Shubik incominciarono ad applicare la teoria dei giochi cooperativi all'economia e giunsero alla formulazione di numerosi concetti utili alla soluzione di giochi con coalizione. David Schmeidler (1969) sviluppò la soluzione per giochi con coalizione chiamata "Nucleolus".

Ulteriori sviluppi si ebbero negli anni '70. John C. Harsanyi (1973) propose una perspicace nuova interpretazione del concetto di strategia mista. Robert J. Aumann (1974) definì l'Equilibrio Correlato per i giochi Bayesiani, mentre Reinhard Selten (1975) introdusse il concetto di "Trembling Hand Equilibrium" per giochi Bayesiani.

Emersero nuove definizioni: Robert J. Aumann (1976) definì formalmente il concetto di "Common Knowledge", mentre B.D. Bernheim e D.G. Pearce (1984) formalizzarono il concetto di "rationalizability".

Si ebbero veloci miglioramenti: David M. Kreps e Robert Wilson (1982) introdussero il concetto di "Sequential Equilibrium" per giochi in forma estesa con informazione imperfetta. Ariel Rubinstein (1982), seguendo l'idea perspicace di Frederik Zeuthen (1930), trasformò la "Nash Bargaining Solution" per i giochi cooperativi in una trasposizione non cooperativa in forma estesa con una sequenza di contrattazioni. Elon Kohlberg e Jean-François Mertens (1986) svilupparono il concetto di "Forward Induction" per i giochi in forma estesa. Drew Fudenberg e E.S. Maskin (1986) svilupparono un famoso teorema per i giochi ricorsivi. Infine J.C. Harsanyi e R.Selten (1988) svilupparono

l'idea di "Equilibrium Selection" per ogni tipo di gioco, mentre D. Fudenberg e Jean Tirole (1991) espressero il "Bayesian Perfect Equilibrium" per i giochi Bayesiani in forma estesa.

Diversi Premi Nobel sono stati assegnati ai maggiori esponenti della Teoria dei Giochi: il Nobel fu condiviso da John Nash, J.C. Harsanyi e R. Selten nel 1994, da William Vickrey e James Mirrlees nel 1996. Herbert Simon, che introdusse concetti di razionalità sostanziale e procedurale, vinse il Premio Nobel nel 1978.

1.1.2 Breve cronologia

Riporto di seguito un'utile cronologia dei documenti principali della Teoria dei Giochi, proposta in [17].

- Teorema di Zermelo (1912). Una partita a scacchi finisce sicuramente in uno dei seguenti tre modi: vince il bianco, vince il nero, oppure stallo. Questo teorema inaugura l'approccio assiomatico ai giochi, come sottoinsieme del cosiddetto programma di Hilbert per una assiomatizzazione completa dei sistemi formali. Esso è stato accuratamente descritto in [35].

- Teorema Minimax (von Neumann, 1928). Basato sul teorema del punto fisso nella sua prima formulazione (Brouwer, 1910), il minimax è il primo concetto di soluzione di un gioco non cooperativo, ma la sua applicabilità è limitata ai giochi a somma costante.

- In *The Theory of Games and Economic Behavior* (1944), von Neumann e Morgenstern introducono il concetto di *stable set* e separano i giochi cooperativi da quelli non cooperativi. Tuttavia, posti di fronte al problema di come risolvere giochi a somma variabile, von Neumann e Morgenstern adottano la linea secondo cui tali giochi si offrono essenzialmente ad una soluzione negoziale e quindi possono essere risolti come giochi di coalizione tramite il concetto di *stable set*. Tutti i giochi considerati sono ad informazione completa.

- Equilibrio di Nash (Nash, 1950, 1951). Basato sul teorema del punto fisso di Kakutani (1941), è un concetto di soluzione valido per qualsiasi gioco non cooperativo. Di fatto, si può considerare come la generalizzazione del Minimax ai giochi a somma variabile.

- Selezione degli equilibri, *backward induction* e perfezione nei sottogiochi (Selten, 1965, 1975). Il criterio della perfezione nei sottogiochi

("subgame perfectness") è il principale raffinamento dell'equilibrio di Nash.

- Giochi ad informazione incompleta, risolti come se fossero ad informazione completa, ma imperfetta (Harsanyi, 1967). Il lavoro di Harsanyi propone il concetto di equilibrio perfetto di Bayes-Nash [29].

1.2 Simulazione strategica

1.2.1 Breve storia

La prima applicazione dei principi moderni del gioco di simulazione è avvenuta in campo militare, nel 1824 in Prussia: sono queste la data e il luogo di nascita del Kriegspiel (letteralmente: 'gioco di guerra') che lo Stato Maggiore prussiano utilizzò per addestrare i propri ufficiali.

Il sistema era semplice: i due allievi si sistemavano in due stanze separate tra loro da una terza, nella quale prendeva posto l'istruttore. Al centro di ogni stanza era dispiegata su un tavolo una grande mappa in scala di una qualunque zona geografica. Venivano indicati gli obiettivi strategici da raggiungere, i giocatori preparavano dei piani di battaglia trasmessi all'istruttore e ponevano i loro pezzi sulla mappa. A sua volta l'istruttore disponeva sulla sua mappa i pezzi di entrambi i giocatori; all'inizio del gioco solo lui sapeva quale fosse lo spiegamento dell'una e dell'altra parte.

Dato il via, i giocatori muovevano a turno i loro pezzi e quando un pezzo di un giocatore diventava visibile a distanza, l'istruttore lo piazzava sulla mappa dell'altro, simulando con precisione la possibilità di avvistamento del nemico su un autentico campo di battaglia. Quando le truppe degli opposti schieramenti giungevano a distanza di tiro, l'istruttore decideva le entità delle perdite e le conseguenze tattiche degli scontri. Il gioco continuava sino alla sconfitta di una delle due parti.

Nel 1837, il generale von Moltke dispose che il Kriegspiel fosse distribuito a ciascun reggimento dell'esercito prussiano; Francia e Turchia si interessarono al gioco. Dopo la vittoria prussiana nella guerra con la Francia del 1870-71, che secondo diversi osservatori si doveva anche all'addestramento degli ufficiali ottenuto con il Kriegspiel, anche Inghilterra e Stati Uniti adottarono analoghi giochi di simulazione.

1.2.2 Risiko al buio

Muovono dal precedente illustre descritto nel paragrafo 1.2.1 numerosi giochi strategici di guerra o varianti di famosi giochi da tavolo o d'ambientazione.

Affronto ora una breve analisi dell'evoluzione, dovuta all'introduzione di incertezza, del famoso gioco di società Risiko ([11]), che vede un interessante conseguenza in comune con la variante degli scacchi, chiamata anch'essa Kriegspiel, che sarà la principale fonte di studio nel seguito.

Il principale attrezzo motorio di Risiko è composto dalle carte Territorio, che rappresentano univocamente le caselle del planisfero. La variante, chiamata "Risiko al buio" prevede che la distribuzione ai giocatori di tali carte rimanga segreta in omaggio al fatto che ci si trova in un gioco militare astratto. Proprio perché il primo attrezzo motorio è costituito da "carte" esse devono rimanere incognite, con il corredo di pedine schierate sui Territori che esse rappresentano.

A sostegno del "buio" vi è il fatto che Risiko, oltre che un gioco d'ambientazione, è un gioco di carte: esse non sono accessorie, ma sono un elemento costitutivo perché rappresentano le singole parti del planisfero. Così come i mazzi di carte rappresentano la totalità dei rispettivi giochi a cui si riferiscono.

Effettuata la distribuzione delle 42 carte dei territori tra i giocatori, questi dispongono in segreto le armate su una scheda strategica. In tal modo nessuno conosce la dislocazione degli altri né in termini di territori né in termini di armate. L'interessante conseguenza risiede nel fatto che nessuno può trarre un vantaggio immediato dalla casualità con cui le carte sono state distribuite.

Si tratta di un'innovazione fondamentale perché in tal modo viene notevolmente ridimensionata la fortuna del primo di mano di essere tale e quindi la sfortuna dell'ultimo di mano: la partita nasce in modo più equilibrato. L'esito dipenderà dall'abilità dei giocatori nello sfruttare le proprie risorse.

Questo è quanto accadde quando si scelse di introdurre nel gioco degli scacchi l'incertezza sulla posizione dei pezzi dell'avversario. Negli scacchi il giocatore bianco comincia la partita avvantaggiato dall'avere per primo il tratto. Ciò che accade invece negli scacchi invisibili, che affronteremo nella sezione 3.2, è una perfetta parità tra i giocatori. Sembra quindi che la "cecità" iniziale elimini il vantaggio dato dal tratto.

L'aspetto che rende interessante la variante del "Risiko al buio" sta nel fatto che la strategia assume un ruolo principe ed è su tale versante che si misura l'abilità dei giocatori, consentendo a tutti di razionalizzare la difesa dei propri confini in attesa degli attacchi degli avversari.

1.3 Organizzazione della tesi

Nel capitolo 2 viene introdotta e approfondita la teoria dei giochi a somma zero, di cui fanno parte, oltre gli Scacchi, il Poker ed il Kriegspiel.

Nel capitolo 3 vengono analizzati gli aspetti dei giochi ad informazione imperfetta. In particolare nella sezione 3.1 si esamina la soluzione adottata nel gioco del Poker, mentre nella sezione 3.2 si descrivono le regole del gioco del Kriegspiel e si analizzano le problematiche tecniche circa l'implementazione di un algoritmo in grado di affrontare questo gioco.

Il capitolo 4 introduce il finale di partita che vede il re e la torre del Bianco contro il re del Nero in Kriegspiel. Le sezioni 4.3 e 4.4 mostrano le soluzioni suggerite dai matematici Magari e Boyce e descrivono l'implementazione procedurale estrapolata da tali soluzioni. La sezione 4.5 propone una soluzione basata su algoritmo di ricerca, ricavata dalle discussioni teoriche affrontate in 3.2.

Nel capitolo 5 si estende l'implementazione vista per il finale di Torre ad altri finali classici del Kriegspiel, quali il finale di Donna (5.1), il finale con due alfieri (5.2) e quello di Pedone (5.3).

Infine, il capitolo 6 riassume il lavoro svolto e presenta i possibili sviluppi futuri.

Capitolo 2

Giochi a somma zero

La Teoria dei Giochi è la disciplina che si occupa di interazione strategica fra decisori, assunti essere intelligenti e "razionali". L'aggettivo intelligenti indica che tali decisori sono in grado di analizzare la situazione in cui si trovano e di conseguenza sono in grado di compiere ragionamenti logici complessi; "razionali" indica che essi hanno preferenze coerenti, ovvero transitive, sugli esiti finali del processo decisionale e che hanno l'obiettivo di massimizzare queste preferenze ([30]). La Teoria dei Giochi proposta da von Neumann e Morgenstern esegue un'analisi completa per la classe di giochi chiamati a *somma zero* con due giocatori ([13]), ovvero giochi in cui un giocatore vince ciò che l'altro giocatore perde. Ne esaminerò ora alcuni importanti aspetti, riportando gli approfondimenti e le formulazioni definite in [13].

2.1 Forma strategica

La descrizione matematica più semplice di un gioco è la cosiddetta *forma strategica*. Poiché per un gioco a somma zero con due giocatori la funzione d'utilità per il giocatore II ritorna un payoff che corrisponde al payoff negato del giocatore I, è possibile in questo caso soffermarsi alla sola funzione di utilità del giocatore I, che chiamerò nel seguito L.

Definizione 1 (Forma strategica) *La forma strategica o forma normale per un gioco a somma zero con due giocatori è data da una tripla* (X, Y, L) *dove*
1) X *è un insieme non vuoto che indica le strategie per il giocatore I*
2) Y *è un insieme non vuoto che indica le strategie per il giocatore II*
3) L *è una funzione di utilità definita su* $X \times Y$ *che ritorna la vincita (o la perdita) per il giocatore I. Quindi* $L : X \times Y \longrightarrow \mathbb{R}$ *ove* $\forall x \in X$ *e* $y \in$

Y, $L(x, y) = v$ e v è la vincita (se v è positivo) o la perdita (se v è negativo) per il giocatore I.

Un gioco in forma strategica (X, Y, L) è spesso chiamato gioco matriciale, poiché la funzione di utilità L può essere rappresentata come matrice. Siano $X = \{x_1, ..., x_m\}$, $Y = \{y_1, ..., y_n\}$, allora si chiama **matrice di gioco** o **matrice dei payoff** la matrice:

$$\mathbf{A}= \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix} \quad \text{dove } a_{ij} = L(x_i, y_j)$$

In questa forma, il giocatore I sceglie una riga ed il giocatore II sceglie una colonna, e II paga a I l'ammontare presente in matrice alle specifiche righe e colonne.

Gli elementi di X e Y vengono chiamati **strategie pure**; scelte più complesse che prendono più strategie pure secondo una qualche distribuzione di probabilità vengono chiamate **strategie miste**. Il giocatore che segue una strategia pura sceglierà sempre una sola riga (o colonna) nella matrice di gioco, quello che segue una strategia mista sceglierà differenti righe (o colonne) con una proporzione.

Quindi una strategia mista per il giocatore I può essere rappresentata da una m-upla $\mathbf{p} = (p_1, p_2, ..., p_m)$ ove p_i indica la probabilità che il giocatore I scelga l'i-esima strategia pura. $\mathbf{p}$ è quindi una partizione di eventi e $\sum_{i=1}^{m} p_i = 1$. Allo stesso modo la strategia mista per il giocatore II è n-upla $\mathbf{q} = (q_1, q_2, ..., q_n)$ ove q_j indica la probabilità che il giocatore II scelga la j-esima strategia pura.

È bene notare che nell'analisi di strategia mista viene fatta un sottile assunzione, ovvero si assume che *quando un giocatore segue una strategia mista, egli è interessato esclusivamente al suo guadagno medio.* Questo significa che il giocatore non è interessato al massimo valore possibile di vittoria, ma solo alla media. Questa assunzione è molto drastica, perché è evidente che di fronte alla scelta di vincere 5 milioni di euro subito e 10 milioni di euro con probabilità di 1/2, tutti sceglierebbero la prima possibilità. L'assunzione fatta è comunque ragionevole per valori monetari piccoli e ci permette di considerare unica la funzione di utilità dei giocatori e quindi di lavorare con giochi a somma zero.

Si consideri ora il semplice esempio proposto in [13], che descrive un caso del gioco 'pari o dispari'[1]. La matrice seguente indica la funzione di utilità L per il giocatore I:

$$\begin{array}{cc} & \text{II (pari)} \\ & \begin{array}{cc} \textit{dispari} & \textit{pari} \end{array} \\ \text{I (dispari)} \quad \begin{array}{r} \textit{dispari} \\ \textit{pari} \end{array} & \begin{pmatrix} -2 & +3 \\ +3 & -4 \end{pmatrix} \end{array}$$

Indichiamo con p il numero di volte in cui il giocatore I chiama il numero dispari e scegliamo p in modo che il giocatore I vinca la stessa cifra in media, sia che il giocatore II giochi un numero pari, sia che egli giochi un numero dispari. Poiché la vincita media per I è $-2p + 3(1-p)$ quando II chiama un numero dispari, ed è $+3p - 4(1-p)$ quando II chiama un numero pari, il giocatore I deve scegliere p in modo da rispettare l'uguaglianza $-2p + 3(1-p) = +3p - 4(1-p) \Rightarrow p = 7/12$. Quindi il giocatore I deve seguire una strategia mista, scegliendo un numero dispari con probabilità $7/12$ e un numero pari con probabilità $5/12$. In media egli vincerà $1/12$, ossia $8,\overline{3}\%$ ogni volta che giocherà, senza considerare quali scelte avrà fatto l'avversario.

Definizione 2 (Equalizing strategy) *La strategia che produce la stessa vincita media a prescindere dalle strategie dell'avversario è chiamata equalizing strategy.*

Ritorniamo all'esempio del gioco 'pari o dispari'. Se, da un lato, il giocatore I può garantirsi una vittoria media del $8,\overline{3}\%$, il giocatore II può usare la stessa strategia del giocatore I e quindi chiamare un numero dispari con probabilità di $7/12$ e un numero pari con probabilità di $5/12$. In questo modo, se I chiama un numero dispari, la perdita media per II è di $-2(7/12) + 3(5/12) = 1/12$, se I chiama un numero pari la perdita per II è di $+3(7/12) - 4(5/12) = 1/12$.

Questo significa che il giocatore I non può garantirsi una vittoria media più alta di $8,\overline{3}\%$ se II gioca al meglio le sue possibilità. Quindi I ha una procedura che gli garantisce un vincita almeno di $1/12$ e II ha una procedura che gli garantisce una perdita al massimo di $1/12$. In questo caso $1/12$ viene chiamato il **valore del gioco** e la procedura che entrambi i giocatori usano per assicurarsi questo risultato è chiamata **strategia ottima** o **strategia minimax**.

[1] entrambi i giocatori tirano a sorte due numeri, che vengono sommati; se la somma è pari vince il giocatore pari, altrimenti vince quello dispari

2.1.1 Soluzioni particolari

È facile risolvere giochi che presentano nella propria matrice elementi "saddle point".

Definizione 3 (Saddle point) *Se un elemento a_{ij} della matrice* **A** *ha le proprietà:*
(1) a_{ij} è il minimo della i-esima riga
(2) a_{ij} è il massimo della j-esima colonna
si dice che a_{ij} è **saddle point**. *Se a_{ij} è saddle point, allora esso corrisponde al valore del gioco e il giocatore I può vincere almeno a_{ij} scegliendo la riga i-esima, e il giocatore II può limitare la perdita a a_{ij} scegliendo la j-esima colonna.*

Per larghe matrici $m \times n$ un metodo semplice per individuare elementi saddle point consiste nel calcolare il minimo di ogni riga e il massimo di ogni colonna, quindi osservare dove i valori corrispondono.

Si consideri il gioco generale con matrice 2×2

$$\mathbf{A}= \begin{pmatrix} a & b \\ d & c \end{pmatrix}$$

Per trovare il valore del gioco e almeno una strategia ottima si eseguono i seguenti passi:
1. si cercano saddle point.
2. se non ci sono saddle point, si risolve trovando una equalizing strategy.
Si ha che, se $a \geq b$ allora $b < c$, altrimenti b sarebbe saddle point. Poiché $b < c$, si deve avere $c > d$, altrimenti c sarebbe saddle point. Continuando si vede che $d < a$ e $a > b$. In altri termini se $a \geq b$ allora $a > b < c > d < a$. Per simmetria, se $a \leq b$, allora $a < b > c < d > a$. Si è quindi ottenuto che se non ci sono saddle point, allora si ha $a > b < c > d < a$ oppure $a < b > c < d > a$.

Per cercare la equalizing strategy supponiamo che il giocatore I usi la strategia mista $(p, 1-p)$, ottenendo l'equazione $ap + d(1-p) = bp + c(1-p)$ e risolvendo su p

$$p = \frac{c-d}{(a-b)+(c-d)}$$

Poiché non ci sono saddle point, $(a-b)$ e $(c-d)$ sono o entrambi positivi o entrambi negativi, quindi $0 < p < 1$.

Il guadagno medio del giocatore I è quindi

$$v = ap + d(1-p) = \frac{ac-bd}{a-b+c-d}$$

Quando il giocatore II sceglie la prima colonna con probabilità q, usando quindi una strategia mista $(q, 1-q)$, si calcola la sua perdita media quando il giocatore I usa le righe 1 e 2 con, $aq + b(1-q) = dq + c(1-q)$ ottenendo

$$q = \frac{c-b}{a-b+c-d}$$

Ancora, poiché non ci sono saddle point si ha $0 < q < 1$. La perdita media per II usando questa strategia è

$$aq + b(1-q) = \frac{ac-bd}{a-b+c-d} = v$$

lo stesso valore ottenuto per il giocatore I. Questo prova che il gioco ha un valore e che i giocatori hanno strategie ottime.

Larghe matrici di gioco a volte possono essere ridotte eliminando righe o colonne che risultano sicuramente sbagliate per il giocatore che le sceglie.

Definizione 4 *Si dice che la i-esima riga di una matrice $\mathbf{A} = (a_{ij})$* **domina (strettamente)** *la k-esima riga se $\forall j \;\; a_{ij} \geq (>) a_{kj}$. La j-esima colonna di* **A domina (strettamente)** *la k-esima colonna se $\forall i \;\; a_{ij} \leq (<) a_{ik}$.*

Ciò che il giocatore I può raggiungere usando una riga dominata può essere raggiunto altrettanto bene usando la riga che la domina e lo stesso vale per le colonne dominate. Quindi una riga o una colonna dominate possono essere eliminate dalla matrice senza modificare il valore del gioco. Da notare che, durante la rimozione di una riga o colonna strettamente dominate, l'insieme delle strategie non cambia; esso può cambiare invece, se si tratta di una dominanza non stretta, nel qual caso l'insieme delle strategie si riduce. Può quindi accadere che strategie ottime vengano cancellate, ma sicuramente ne resteranno di utilizzabili.

Una riga (o colonna) può inoltre essere rimossa se è dominata da una combinazione di probabilità di altre righe (o colonne). Se per qualche $0 < p < 1$, $pa_{i_1 j} + (1-p)a_{i_2} j \geq a_{kj} \;\; \forall j$, allora la k-esima riga è dominata dalla strategia mista che sceglie la riga i_1 con probabilità p e la riga i_2 con probabilità $1-p$. Il giocatore I può fare una scelta buona, tramite questa strategia mista, almeno quanto se avesse scelto la riga k. Inoltre, ogni strategia mista che sceglieva la riga k con probabilità p_k può essere rimpiazzata da quella in cui la probabilità di k è divisa tra i_1 e i_2, cioé, la probabilità di i_1 diviene pp_k, mentre quella per i_2 diviene $(1-p)p_k$. Una simile argomentazione può essere condotta per le colonne.

Affrontiamo ora il caso di un gioco con matrice $2 \times n$ (o $m \times 2$), considerando il seguente esempio:

$$\begin{array}{cc} p & \\ 1-p & \end{array} \begin{pmatrix} 2 & 3 & 1 & 5 \\ 4 & 1 & 6 & 0 \end{pmatrix}$$

Si supponga che il giocatore I scelga la prima riga con probabilità p e la seconda riga con probabilità $1-p$. Se II sceglie la colonna 1, il payoff medio per I è di $2p + 4(1-p)$. Allo stesso modo, le scelte da parte del giocatore II delle colonne 2, 3 e 4 portano alle equazioni $3p+1(1-p)$, $p+6(1-p)$ e $5p$. Un semplice metodo grafico consiste nel tracciare queste quattro funzioni lineari di p, ricordando che $0 \leq p \leq 1$. Per un valore fissato di p, il giocatore I può assicurarsi che il suo guadagno medio sia almeno il minimo di queste quattro funzioni valutate su p. Quanto detto è conosciuto come lower envelope di queste funzioni. Poiché I vuole massimizzare il guadagno medio, egli vuole trovare p tale che indichi il massimo della lower envelope.

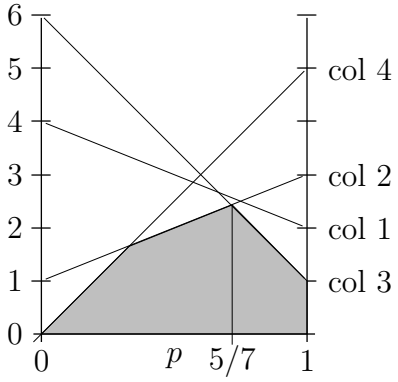


Figura 2.1: Soluzione con matrice $2 \times n$

Come mostra il grafico in figura 2.1 il massimo della lower envelope si ottiene nell'intersezione della colonna 2 con la colonna 3 e questo significa risolvere il gioco ristretto alla matrice $\begin{pmatrix} 3 & 1 \\ 1 & 6 \end{pmatrix}$ che ha valore $v = 17/7$. La strategia ottima per il giocatore I è $(5/7, 2/7)$ e per il giocatore II, nel gioco originale, è $(0, 5/7, 2/7, 0)$.

Si noti che, in figura, la colonna 1 non ha alcun ruolo, ossia la lower envelope resta invariata se viene eliminata la linea relativa alla colonna 1. Questo è pertanto un buon test per la dominazione di colonne (o righe), la colonna 1 è infatti dominata dalle colonne 2 e 3 prese con probabilità di 1/2 entrambe.

2.1.2 Il principio di indifferenza

Data la matrice $\mathbf{A}$ $m \times n$ di un gioco, se il giocatore I usa una strategia mista $\mathbf{p} = (p_1, p_2, ..., p_m)$ e il giocatore II sceglie la colonna j, il payoff medio del

giocatore I è pari a $\sum_{i=1}^{m} p_i a_{ij}$. Se V è il valore del gioco, una strategia ottima per I è caratterizzata dalla proprietà che il payoff medio per I è almeno V a prescindere dalla colonna j scelta da II, ossia

$$\sum_{i=1}^{m} p_i a_{ij} \geq V \quad \forall j \tag{2.1}$$

Allo stesso modo $\mathbf{q} = (q_1, q_2, ..., q_n)$ è una strategia ottima per II se e solo se

$$\sum_{j=1}^{n} a_{ij} q_j \leq V \quad \forall i \tag{2.2}$$

Quando entrambi i giocatori seguono le loro strategie ottime, il payoff medio $\sum_i \sum_j p_i a_{ij} q_j$ è esattamente V, in quanto

$$\begin{aligned} V = \sum_{j=1}^{n} V q_j \quad &\leq \textstyle\sum_{j=1}^{n} \left(\sum_{i=1}^{m} p_i a_{ij}\right) q_j = \sum_{i=1}^{m} \sum_{j=1}^{n} p_i a_{ij} q_j \\ &= \textstyle\sum_{i=1}^{m} p_i \left(\sum_{j=1}^{n} a_{ij} q_j\right) \leq \sum_{i=1}^{m} p_i V = V \end{aligned} \tag{2.3}$$

Il seguente teorema enuncia le condizioni in cui si ha l'uguaglianza in (2.1) per certi valori di j, e in (2.2) per certi valori di i.

Teorema 1 (The Equilibrium Theorem) *Si consideri un gioco con matrice* $\mathbf{A}$ $m \times n$ *e valore* V. *Sia* $\mathbf{p} = (p_1, p_2, ..., p_m)$ *una strategia ottima per il giocatore I e* $\mathbf{q} = (q_1, q_2, ..., q_n)$ *ottima per il giocatore II, allora*

$$\sum_{i=1}^{m} p_i a_{ij} = V \quad \textit{per ogni } j \textit{ in cui } q_j > 0$$

$$\sum_{j=1}^{n} a_{ij} q_j = V \quad \textit{per ogni } i \textit{ in cui } p_i > 0$$

dimostrazione. Si supponga che esista un k tale che $p_k > 0$ e $\sum_{j=1}^{n} a_{kj} q_j \neq V$. Allora dalla (2.2), $\sum_{j=1}^{n} a_{kj} q_j < V$, ma per la (2.3) si ha

$$V = \sum_{i=1}^{m} p_i \left(\sum_{j=1}^{n} a_{ij} q_j\right) < \sum_{i=1}^{m} p_i V = V$$

ove la diseguaglianza è stretta per via del k-esimo termine della somma. Tale contraddizione dimostra la prima conclusione del teorema, per la seconda si procede in modo analogo. $\diamond$

Se esiste una strategia ottima per il giocatore I che dà probabilità positiva alla riga i, allora ogni strategia ottima del giocatore II permette a I di

calcolare il valore del gioco se egli usa la riga i. La procedura suggerita al giocatore I è di provare a trovare una soluzione alle equazioni $\sum_{i=1}^{m} p_i a_{ij} = V$ formate dagli indici j in cui si considera che $q_j > 0$. Il giocatore I cerca quindi una strategia che rende il giocatore II indifferente su quale strategia ottima utilizzare. Chiaramente il giocatore II può fare lo stesso. Quanto enunciato prende il nome di **principio di indifferenza**.

2.1.3 Giochi con matrici non singolari

Si consideri una matrice di gioco $\mathbf{A}$ $m \times m$, e si supponga che $\mathbf{A}$ sia non singolare. Se assumiamo che *il giocatore I abbia una strategia ottima che dà probabilità positiva ad ogni riga*, caso che viene chiamato "all-strategies-active", allora, per il principio di indifferenza, ogni strategia ottima $\mathbf{q}$ di II soddisfa

$$\sum_{j=1}^{n} a_{ij} q_j = V \quad \forall i = 1, ..., m$$

Questo è un sistema di m equazioni in m incognite e, poiché $\mathbf{A}$ è non singolare, è possibile risolverlo su q_j. Sia $\mathbf{q}$ il vettore colonna delle strategie per il giocatore II e $\mathbf{1} = (1, 1, ..., 1)^T$ il vettore colonna composto da elementi 1, il sistema può essere scritto come $\mathbf{Aq} = V\mathbf{1}$.

Si noti che V non può essere zero, altrimenti $\mathbf{A}$ sarebbe singolare e, poiché $\mathbf{A}$ è non singolare, $\mathbf{A}^{-1}$ esiste. Moltiplicando entrambi i membri sulla sinistra per $\mathbf{A}^{-1}$, otteniamo $\mathbf{q} = V\mathbf{A}^{-1}\mathbf{1}$.

Per trovare V, si ricorre all'equazione $\sum_{j=1}^{m} q_j = 1$, che in notazione vettoriale è $\mathbf{1^T q} = 1$. Moltiplicando quindi per $\mathbf{1^T}$ la parte sinistra dell'equazione precedente si ottiene $1 = V\mathbf{1^T A^{-1} 1}$. La strategia ottima per il giocatore II è quindi data da

$$\mathbf{q} = \frac{\mathbf{A^{-1}1}}{\mathbf{1^T A^{-1} 1}} \tag{2.4}$$

Se qualche componente q_j risulta essere negativo, allora l'assunzione che I abbia una strategia ottima che dà peso positivo ad ogni riga è falsa. Se invece $q_j \geq 0 \quad \forall j$, si può cercare una strategia ottima per I con lo stesso metodo, ottenendo

$$\mathbf{p^T} = \frac{\mathbf{1^T A^{-1}}}{\mathbf{1^T A^{-1} 1}} \tag{2.5}$$

Ciò può essere sintetizzato nel teorema seguente.

Teorema 2 *Si assuma che la matrice* $\mathbf{A}$ *sia non singolare e che* $\mathbf{1^T A^{-1} 1} \neq 0$. *Allora il gioco di matrice* $\mathbf{A}$ *ha un valore* $V = 1/\mathbf{1^T A^{-1} 1}$ *e strategie ottime* $\mathbf{p^T} = V\mathbf{1^T A^{-1}}$ *e* $\mathbf{q} = V\mathbf{A^{-1}1}$, *a patto che siano* $\mathbf{p} \geq \mathbf{0}$ *e* $\mathbf{q} \geq \mathbf{0}$.

Se il valore del gioco è zero, questo metodo non può essere applicato direttamente. Nel caso di una matrice singolare comunque, è possibile aggiungere una costante positiva ad ogni elemento della matrice per ottenere un valore del gioco positivo, e questo può rendere la matrice di gioco non singolare.

2.1.4 Giochi simmetrici

Un gioco è detto simmetrico se i giocatori hanno stesse regole, quindi la matrice è quadrata, ed il payoff se I usa i e II usa j è il negato del payoff se I usa j e II usa i. Questo comporta che la matrice di gioco sia skew-symmetric, ovvero $\mathbf{A} = -\mathbf{A^T}$, o $a_{ij} = -a_{ji} \ \ \forall i, j$

Definizione 5 (Gioco Simmetrico) *Un gioco finito è detto simmetrico se la sua matrice di gioco è quadrata e skew-symmetric.*

Si noti che, seguendo la definizione di gioco simmetrico, il gioco "pari o dispari", con matrice

$$\begin{array}{cc} & \text{II} \\ & \begin{array}{cc} \text{pari} & \text{dispari} \end{array} \\ \text{I} \begin{array}{r} \text{pari} \\ \text{dispari} \end{array} & \begin{pmatrix} -1 & 1 \\ 1 & -1 \end{pmatrix} \end{array}$$

non viene considerato simmetrico, in quanto la matrice non è skew-symmetric.

Il gioco della " morra cinese", al contrario, ha una matrice skew-symmetric

$$\begin{array}{cc} & \text{II} \\ & \begin{array}{ccc} \text{carta} & \text{forbici} & \text{sasso} \end{array} \\ \text{I} \begin{array}{r} \text{carta} \\ \text{forbici} \\ \text{sasso} \end{array} & \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{pmatrix} \end{array}$$

ed è quindi un gioco simmetrico. Si noti che gli elementi della diagonale sono pari a zero e questo è vero per tutte le matrici skew-symmetric, in quanto $a_{ii} = -a_{ii}$ implica $a_{ii} = 0 \ \ \forall i$.

Teorema 3 *Un gioco simmetrico ha valore zero. Ogni strategia ottima per un giocatore è anche ottima per l'altro.*

dimostrazione. Sia $\mathbf{p}$ una qualsiasi strategia per I. Se II usa la stessa strategia il payoff medio è zero, poiché

$$\mathbf{p^T A p} = \sum\sum p_i a_{ij} p_j = \sum\sum p_i(-a_{ji})p_j = -\sum\sum p_j a_{ji} p_i = -\mathbf{p^T A p}$$

il che implica $\mathbf{p^T A p} = 0$. Poiché $\min_q \mathbf{p^T A q} \leq \mathbf{p^T A p} = 0 \ \ \forall \mathbf{p}$, segue che $\underline{V} \leq 0$. Allo stesso modo, $\max_p \mathbf{p^T A q} \geq 0 \ \ \forall \mathbf{q}$, quindi $\overline{V} \geq 0$. Ciò comporta l'esistenza del valore del gioco pari a zero. Si supponga che $\mathbf{p}$ sia ottima per I, allora $\sum_{i=1}^m p_i a_{ij} \geq 0 \ \ \forall j$. Quindi $\sum_{j=1}^m a_{ij} p_j = -\sum_{j=1}^m p_j a_{ji} \leq 0 \ \ \forall i$, perciò $\mathbf{p}$ è anche ottima per II. Per simmetria, se $\mathbf{q}$ è ottima per II, è ottima anche per I. $\diamond$

2.1.5 Invarianza

Definizione 6 *Sia $G = (X, Y, L)$ un gioco finito, e sia g una trasformazione biettiva di Y in Y. Il gioco G è detto invariante sotto g se per ogni $x \in X$ esiste un unico $x' \in X$ tale che*

$$L(x, y) = L(x', g(y)) \quad \forall y \in Y$$

Osserviamo che, se ci fosse un altro punto $x'' \in X$ tale che

$$L(x, y) = L(x'', g(y)) \quad \forall y \in Y$$

allora, avremmo $L(x', g(y)) = L(x'', g(y)) \ \ \forall y \in Y$, e, poiché g è suriettiva

$$L(x', y) = L(x'', y) \quad \forall y \in Y.$$

Quindi le strategie x' e x'' hanno payoff identici e possiamo rimuoverne una da X senza cambiare il problema.

Assumeremo quindi, senza perdere di generalità, che tutte le strategie duplicate sono state eliminate, ossia

$L(x', y) = L(x'', y) \ \ \forall y \in Y$ implica che $x' = x''$, e $L(x, y') = L(x, y'') \ \ \forall x \in X$ implica che $y' = y''$.

L'unicità di x' dipende quindi da questa assunzione. Poiché dipende solo da g e da x, chiameremo $x' = \overline{g}(x)$. Otteniamo

$$L(x, y) = L(\overline{g}(x), g(y)) \quad \forall x \in X \ \ e \ \ y \in Y$$

$\overline{g}$ è una trasformazione iniettiva di X, poiché se $\overline{g}(x_1) = \overline{g}(x_2)$, allora

$$L(x_1, y) = L(\overline{g}(x_1), g(y)) = L(\overline{g}(x_2), g(y)) = L(x_2, y) \quad \forall y \in Y$$

che implica $x_1 = x_2$, ricordando l'assunzione fatta in precedenza. Quindi esiste l'inversa g^{-1} di g, definita come $g^{-1}(g(x)) = g(g^{-1}(x)) = x$ e, poiché ogni trasformazione iniettiva di un insieme finito è anche suriettiva, $\overline{g}$ è una trasformazione biettiva di X in X.

Lemma 1 *Se un gioco finito, $G = (X, Y, L)$, è invariante per una trasformazione iniettiva g, allora G è anche invariante per g^{-1}.*

dimostrazione. Poiché il gioco è invariante per g, è dato $L(x, y) = L(\overline{g}(x), g(y))\ \forall x \in X\ e\ y \in Y$. Se si rimpiazzano y con $g^{-1}(y)$ e x con $\overline{g}^{-1}(x)$, otteniamo $L(\overline{g}^{-1}(x), g^{-1}(y)) = L(x, y)\ \ \forall x \in X\ e\ y \in Y$. $\diamond$

Lemma 2 *Se un gioco finito, $G = (X, Y, L)$, è invariante per due trasformazioni iniettive g_1 e g_2, allora G è invariante anche per la trasformazione composta $g_2 g_1$, definita come segue: $g_2 g_1(y) = g_2(g_1(y))$.*

dimostrazione. Dato $L(x, y) = L(\overline{g}_1(x), g_1(y))\ \forall x \in X\ e\ y \in Y$ e $L(x, y) = L(\overline{g}_2(x), g_2(y))\ \forall x \in X\ e\ y \in Y$, si ottiene

$$L(x, y) = L(\overline{g}_2(\overline{g}_1(x)), g_2(g_1(y))) = L(\overline{g}_2(\overline{g}_1(x)), g_2 g_1(y)) \quad \forall y \in Y\ e\ x \in X.$$

da cui G è invariante per $g_2 g_1$. $\diamond$

Queste dimostrazioni indicano infine che:

$$\overline{g_2 g_1} = \overline{g}_2 \overline{g}_1 \quad e \quad \overline{g^{-1}} = \overline{g}^{-1} \tag{2.6}$$

Quindi la classe di trasformazione, g su Y, per le quali il problema è invariante forma un gruppo $\mathcal{G}$ con composizione data dall'operatore di moltiplicazione. L'elemento identità, e, del gruppo è la trasformazione identità $e(y) = y\ \forall y \in Y$. L'insieme $\overline{\mathcal{G}}$ di trasformazioni $\overline{g}$ su X, è anch'esso un gruppo, con identità $\overline{e}(x) = x\ \ \forall x \in X$. L'equazione (2.6) prova che $\overline{\mathcal{G}}$ è isomorfo a $\mathcal{G}$. Quindi avremmo potuto analizzare il problema dal punto di vista del giocatore I e saremmo arrivati agli stessi gruppi $\overline{\mathcal{G}}$ e $\mathcal{G}$.

Definizione 7 *Un gioco finito $G = (X, Y, L)$ è detto invariante sotto un gruppo $\mathcal{G}$ di trasformazioni, se $L(x, y) = L(\overline{g}(x), g(y))\ \ \forall x \in X\ e\ y \in Y$ si verifica per ogni $g \in \mathcal{G}$*

Definizione 8 *Dato un gioco finito $G = (X, Y, L)$, invariante sotto un gruppo $\mathcal{G}$ di trasformazioni iniettive di Y, una strategia mista $\mathbf{q} = (q(1), ..., q(n))$ per il giocatore II è detta invariante sotto $\mathcal{G}$ se*

$$q(g(y)) = q(y) \qquad \forall y \in Y,\ g \in \mathcal{G}$$

Allo stesso modo una strategia mista $\mathbf{p} = (p(1), ..., p(m))$ per il giocatore I è detta invariante sotto $\mathcal{G}$ o $(\overline{\mathcal{G}})$ se

$$p(\overline{g}(x)) = p(x) \qquad \forall x \in X,\ \overline{g} \in \overline{\mathcal{G}}$$

Due punti y_1 e y_2 in Y sono equivalenti se esiste un $g \in \mathcal{G}$ tale che $g(y_2) = y_1$. Tale classe di equivalenza è a volte chiamata *orbita*, quindi y_1 e y_2 sono equivalenti se rimangono sulla stessa orbita. In altre parole, una strategia mista $\mathbf{q}$ per il giocatore II è invariante se assegna la stessa probabilità a tutte le strategie pure nell'orbita. Quanto detto è formalizzato nel seguente teorema.

Teorema 4 *Se un gioco finito $G = (X, Y, L)$ è invariante sotto un gruppo $\mathcal{G}$, allora esistono strategie ottime invarianti per i giocatori.*

dimostrazione. Poiché il gioco è finito, esiste un valore, V, e una strategia ottima mista per il giocatore II, $\mathbf{q}^*$, ossia

$$\sum_{y \in Y} L(x, y) q^*(y) \leq V \quad \forall x \in X \tag{2.7}$$

Si deve dimostrare che esiste una strategia invariante $\tilde{\mathbf{q}}$ che soddisfa la medesima condizione. Sia $N = |\mathcal{G}|$ il numero di elementi nel gruppo $\mathcal{G}$. Si definisce

$$\tilde{q}(y) = \frac{1}{N} \sum_{g \in \mathcal{G}} q^*(g(y))$$

che sostituisce ogni probabilità in un'orbita, con la media delle probabilità nell'orbita. Si ha che $\tilde{\mathbf{q}}$ è invariante poiché per ogni $g' \in \mathcal{G}$

$$\begin{aligned} \tilde{q}(g'(y)) &= \frac{1}{N} \sum_{g \in \mathcal{G}} q^*(g(g'(y))) \\ &= \frac{1}{N} \sum_{g \in \mathcal{G}} q^*(g(y)) = \tilde{q}(y) \end{aligned}$$

Inoltre, $\tilde{\mathbf{q}}$ soddisfa la (2.7), in quanto

$$\begin{aligned} \sum_{y \in Y} L(x, y) \tilde{q}(y) &= \sum_{y \in Y} L(x, y) \frac{1}{N} \sum_{g \in \mathcal{G}} q^*(g(y)) \\ &= \frac{1}{N} \sum_{g \in \mathcal{G}} \sum_{y \in Y} L(x, y) q^*(g(y)) \\ &= \frac{1}{N} \sum_{g \in \mathcal{G}} \sum_{y \in Y} L(\overline{g}(x), g(y)) q^*(g(y)) \\ &= \frac{1}{N} \sum_{g \in \mathcal{G}} \sum_{y \in Y} L(\overline{g}(x), y) q^*(y) \\ &\leq \frac{1}{N} \sum_{g \in \mathcal{G}} V = V \qquad \diamond \end{aligned}$$

Il gioco "pari o dispari", ad esempio, è composto da $X = Y = \{1, 2\}$, $L(1,1) = L(2,2) = -1$ e $L(1,2) = L(2,1) = 1$. Il gioco $G = (X, Y, L)$ è invariante rispetto al gruppo $\mathcal{G} = \{e, g\}$ ove e è la trasformazione identità, mentre g è la trasformazione $g(1) = 2,\ g(2) = 1$. La strategia mista $(q(1), q(2))$ è invariante per $\mathcal{G}$ se $q(1) = q(2)$. Poiché $q(1) + q(2) = 1$, si ottiene che $q(1) = q(2) = 1/2$ è la sola strategia invariante per il giocatore II e, quindi, è una strategia minimax. Allo steso modo, $p(1) = p(2) = 1/2$ è la sola strategia invariante, e quindi minimax, per il giocatore I.

Con un ragionamento simile, si può esaminare il gioco della "morra cinese", che risulta invariante sotto il gruppo $\mathcal{G} = \{e, g, g^2\}$, dove $g(carta = forbici,\ g(forbici) = sasso$ e $g(sasso) = carta$. L'unica strategia invariante, e quindi strategia minimax, dà probabilità $1/3$ ad ogni scelta tra carta, forbici e sasso.

2.1.6 Soluzione generale

Si consideri un gioco finito a somma zero, (X, Y, L) con matrice $m \times n$ $\mathbf{A}$. Consideriamo lo spazio delle strategie X come l'insieme dei primi m interi, e Y come l'insieme dei primi n. Una strategia mista per il giocatore I può essere rappresentata da un vettore di probabilità $\mathbf{p} = (p_1, p_2, ..., p_m)^T$. Allo stesso modo si può rappresentare con $\mathbf{q} = (q_1, q_2, ..., q_n)^T$ la strategia mista per II. Gli insiemi di strategie miste per I e II saranno nel seguito indicati rispettivamente con X^* e Y^*,

$$X^* = \{\mathbf{p} = (p_1, p_2, ..., p_m)^T : p_i \geq 0,\ per\ i = 1, ..., m\ e\ \textstyle\sum_i^m p_i = 1\}$$
$$Y^* = \{\mathbf{q} = (q_1, q_2, ..., q_m)^T : q_j \geq 0,\ per\ j = 1, ..., n\ e\ \textstyle\sum_j^n q_j = 1\}$$

Il vettore unità $e_k \in X^*$ ove è impostato a 1 il k^{esimo} elemento e a 0 i restanti, indica la strategia pura del giocatore I che sceglie la riga k.

Supponiamo che il giocatore II scelga le colonne usando il vettore $\mathbf{q} \in Y^*$. Se il giocatore I sceglie la riga i, il payoff medio per I è

$$\sum_{j=1}^{n} a_{ij} q_j = (\mathbf{Aq})_i \tag{2.8}$$

l'i^{esimo} elemento del vettore $\mathbf{Aq}$. Allo stesso modo, se il giocatore I usa $\mathbf{p} \in X^*$ e il giocatore II la colonna j, allora il payoff medio per I diviene

$$\sum_{i=1}^{m} p_i a_{ij} = (\mathbf{p^T A})_j \tag{2.9}$$

il j^{esimo} elemento del vettore $\mathbf{p^T A}$. Più in generale, se il giocatore I usa $\mathbf{p} \in X^*$ e il giocatore II usa $\mathbf{q} \in Y^*$, il payoff per I risulta

$$\sum_{i=1}^{m} \left(\sum_{j=1}^{n} a_{ij} q_j \right) p_i = \sum_{i=1}^{m} \sum_{j=1}^{n} p_i a_{ij} q_j = \mathbf{p^T A q} \tag{2.10}$$

Supponendo di sapere che il giocatore II userà una precisa strategia $\mathbf{q} \in Y^*$, il giocatore I sceglierebbe quella riga i che massimizza (2.8), oppure, in modo equivalente, sceglierebbe quella $\mathbf{p} \in X^*$ che massimizza la (2.10). Il suo payoff medio è

$$\max_{1 \le i \le m} \sum_{j=1}^{n} a_{ij} q_j = \max_{\mathbf{p} \in X^*} \mathbf{p^T A q} \tag{2.11}$$

La quantità sinistra è il massimo di $\mathbf{p^T A q}$ quando $\mathbf{p}$ indica una strategia pura e, poiché $X \subset X^*$, essa deve essere minore o uguale alla quantità di destra; viceversa, poiché (2.10) è una media delle quantità in (2.8) deve essere minore o uguale al valore più grande in (2.8).

Ogni $\mathbf{p} \in X^*$ che ottiene il massimo di (2.10) è chiamato **best response** o strategia bayesiana contro $\mathbf{q}$. In particolare, ogni riga i che ottiene il massimo di (2.8) è una strategia bayesiana pura contro $\mathbf{q}$.

Nei giochi finiti esistono sempre strategie bayesiane pure contro $\mathbf{q}$ per ogni $\mathbf{q} \in Y^*$. Allo stesso modo, se si conosce quale particolare strategia $\mathbf{p} \in X^*$ il giocatore I userà, allora il giocatore II può scegliere quella colonna j che minimizza (2.9) oppure quel $\mathbf{q} \in Y^*$ che minimizza 2.10. Il payoff diviene quindi

$$\min_{1 \le j \le n} \sum_{i=1}^{m} p_i a_{ij} = \min_{\mathbf{q} \in Y^*} \mathbf{p^T A q} \tag{2.12}$$

Ogni $\mathbf{q} \in Y^*$ che ottiene il minimo di (2.12) è chiamato **best response** o strategia bayesiana per il giocatore II contro $\mathbf{p}$.

Si supponga che II debba comunicare la scelta della sua strategia mista $\mathbf{q} \in Y^*$ prima che I faccia la sua giocata. Questo sembra rendere il gioco più favorevole a I. Se II comunica $\mathbf{q}$, allora certamente I userà una strategia bayesiana contro $\mathbf{q}$ e II perderà l'ammontare calcolato con (2.11). Di conseguenza II deciderà di annunciare quel $\mathbf{q}$ che minimizza (2.11). Il minimo di (2.11) su tutti i $\mathbf{q} \in Y^*$ è chiamato **valore superiore del gioco** (X,Y,L) ed è indicato con $\overline{V}$:

$$\overline{V} = \min_{\mathbf{q} \in Y^*} \max_{1 \le i \le m} \sum_{j=1}^{n} a_{ij} q_j = \min_{\mathbf{q} \in Y^*} \max_{\mathbf{p} \in X^*} \mathbf{p^T A q} \tag{2.13}$$

Ogni $\mathbf{q} \in Y^*$ che ottiene il minimo in (2.13) è chiamata **strategia minimax** per II. Esiste sempre una strategia minimax nei giochi finiti, poiché

la quantità in (2.11), essendo il massimo di m funzioni lineari $\mathbf{q}$, è una funzione continua di $\mathbf{q}$ ed essendo Y^* insieme chiuso, questa funzione assume il minimo su Y^* in qualche punto di Y^*.

$\overline{V}$ è la più piccola perdita media che il giocatore II può assicurarsi, non importa cosa il giocatore I faccia.

Un'analisi simile può essere condotta assumendo che I debba annunciare la sua strategia mista $\mathbf{p} \in X^*$ prima che II faccia la sua scelta. Essendo (2.12) il payoff medio per I avendo annunciato $\mathbf{p}$, egli sceglierà di conseguenza quel $\mathbf{p}$ che massimizza (2.12) ottenendo ciò che viene chiamato **valore inferiore del gioco** e che viene indicato con $\underline{V}$.

$$\underline{V} = \max_{\mathbf{p}\in X^*} \min_{1\le j\le n} \sum_{i=1}^{m} p_i a_{ij} = \max_{\mathbf{p}\in X^*} \min_{\mathbf{q}\in Y^*} \mathbf{p^T A q} \tag{2.14}$$

$\underline{V}$ è la quantità massima che I può garantire a se stesso, qualsiasi cosa faccia il giocatore II. Ogni $\mathbf{p} \in X^*$ che raggiunge il massimo in (2.14) è chiamata **strategia minimax** per il giocatore I. Come per II, è possibile notare che il giocatore I ha sempre una strategia minimax.

Lemma 3 *In un gioco finito, entrambi i giocatori hanno strategie minimax.*

È facile osservare che il valore inferiore è minore o uguale al valore superiore, se così non fosse, ossia $\overline{V} < \underline{V}$, il giocatore I potrebbe garantirsi di vincere almeno $\underline{V}$, mentre il giocatore II non potrebbe più limitare le sue perdite a $\overline{V}$, che porta a una contraddizione.

Lemma 4 *Il valore inferiore è minore o uguale al valore superiore, $\underline{V} \le \overline{V}$.*

Questo lemma segue anche dal principio matematico per cui per ogni funzione a valori reali $f(x, y)$ e ogni insieme, X^* e Y^*,

$$\max_{x\in X^*} \min_{y\in Y^*} f(x, y) \le \min_{y\in Y^*} \max_{x\in X^*} f(x, y)$$

Si noti che $\min_{y'} f(x, y') \le f(x, y) \le \max_{x'} f(x', y)$ per ogni x e y fissati; ora considerando il $\max_x$ alla sinistra la diseguaglianza non cambia, né cambia considerando $\min_y$ sulla destra, da cui il risultato.

Se $\underline{V} \le \overline{V}$, il payoff medio del gioco cade tra $\underline{V}$ e $\overline{V}$. Quando $\underline{V} = \overline{V}$, si raggiunge una situazione stabile.

Definizione 9 *Se $\underline{V} = \overline{V}$, diciamo che il valore del gioco V esiste ed è uguale a $\underline{V}$ ovvero $\overline{V}$. Se V esiste ci si riferirà alle strategie minimax come* **strategie ottime**.

Teorema 5 (Minimax) *Ogni gioco finito ha un valore ed entrambi i giocatori hanno strategie minimax.*

Un corollario importante al teorema indica che se le regole del gioco prevedono che il giocatore II annunci, la strategia mista che ha intenzione di usare, prima che il giocatore I giochi, il vantaggio dato al giocatore I è in realtà apparente in quanto II può semplicemente scegliere la strategia minimax.

Un'ulteriore osservazione utile in questo contesto concerne l'invarianza delle strategie minimax rispetto all'operazione di addizione di una costante ad ogni entrata della matrice di gioco e alla moltiplicazione della matrice di gioco di una costante positiva. Il gioco con matrice $\mathbf{A} = (a_{ij})$ e il gioco con matrice $\mathbf{A}' = (a'_{ij})$ ove $a'_{ij} = a_{ij} + b$, con b un numero reale arbitrario, hanno molti aspetti in comune; infatti il gioco con matrice $\mathbf{A}'$ è equivalente al gioco in cui II paga a I la cifra b, quindi I e II giocano il gioco con matrice $\mathbf{A}$.

Chiaramente ogni strategia usata nel gioco con matrice $\mathbf{A}'$ somma al payoff, ottenuto dal giocatore I giocando le medesime strategie nel gioco con matrice $\mathbf{A}$, il bonus b. Quindi ogni strategia minimax per entrambi i giocatori in un gioco è ancora minimax nell'altro, e il valore del gioco con matrice $\mathbf{A}'$ è b sommato al valore del gioco con matrice $\mathbf{A}$.

Similmente il gioco con matrice $\mathbf{A}'' = (a''_{ij})$ con $a''_{ij} = ca_{ij}$, con c costante positiva può essere considerato come il gioco di matrice $\mathbf{A}$ con un cambiamento di scala[2]; in questo modo le strategie minimax non cambiano e il valore di $\mathbf{A}''$ è c volte quello di $\mathbf{A}$.

Lemma 5 *Se $\mathbf{A} = (a_{ij})$ e $\mathbf{A}' = (a'_{ij})$ sono matrici con $a'_{ij} = ca_{ij} + b$ e $c > 0$, allora il gioco con matrice $\mathbf{A}$ ha le stesse strategie minimax per i giocatori I e II del gioco con matrice $\mathbf{A}'$. Se V denota il valore del gioco con matrice $\mathbf{A}$, allora il valore V' del gioco con matrice $\mathbf{A}'$ soddisfa $V' = cV + b$.*

Affrontiamo ora la dimostrazione del Teorema 5 basandoci sulla programmazione lineare ([13]). Consideriamo, senza perdere di generalità, il gioco dal punto di vista del giocatore I: egli vuole scegliere $p_1, ..., p_m$ tali che massimizzino (2.12) considerando il vincolo $\mathbf{p} \in X^*$. Questo può essere formulato come: scegli $p_1, ..., p_m$ t.c.

$$\text{massimizza} \quad \min_{1 \le j \le n} \sum_{i=1}^{m} p_i a_{ij} \tag{2.15}$$

con i vincoli

$$p_1 + ... + p_m = 1$$

[2]ad esempio considerando differenti valute monetarie

e

$$p_i \geq 0 \quad \forall i$$

Sebbene i vincoli siano lineari, la funzione obiettivo non è una funzione lineare a causa dell'operatore min, comunque è possibile ottenere un programma lineare con un piccolo escamotage. Si aggiunge una nuova variabile v alla lista di variabili del giocatore I, e si richiede che essa sia minore della funzione obiettivo, $v \leq \min_{1\leq j\leq n} \sum_{i=1}^{m} p_i a_{ij}$, quindi si rende v più grande possibile, considerando questi nuovi vincoli. Il problema diviene: scegli v e $p_1, ..., p_m$ t.c.

$$\text{massimizza} \quad v$$

con i vincoli

$$\begin{aligned} v &\leq \textstyle\sum_{i=1}^{m} p_i a_{i1} \\ &\vdots \\ v &\leq \textstyle\sum_{i=1}^{m} p_i a_{in} \\ p_1 + ... + p_m &= 1 \\ p_i &\geq 0 \quad \forall i \end{aligned} \tag{2.16}$$

Con la formulazione del problema (2.16) si ottiene un programma lineare che può essere risolto utilizzando il metodo del simplesso.

Analogamente, è possibile vedere il problema dal punto di vista del giocatore II e giungere alla formulazione di un programma lineare simile. Per il giocatore II il problema è: scegli w e $q_1, ..., q_n$ t.c.

$$\text{minimizza} \quad w$$

con i vincoli

$$\begin{aligned} w &\geq \textstyle\sum_{j=1}^{n} a_{1j} q_j \\ &\vdots \\ w &\geq \textstyle\sum_{j=1}^{n} a_{mj} q_j \\ q_1 + ... + q_m &= 1 \\ q_j &\geq 0 \quad \forall j \end{aligned} \tag{2.17}$$

La teoria della dualità della programmazione lineare mostra come questi due programmi, (2.16) e (2.17), siano duali. Per il Teorema di Dualità si può concludere che i due programmi hanno lo stesso valore, il massimo che il giocatore I raggiunge in (2.16) è uguale al minimo che il giocatore II raggiunge in (2.17). Ciò è esattamente quanto enuncia il Teorema Minimax, possiamo

quindi concludere che il Teorema di Dualità implica il teorema Minimax.

Se il valore del gioco è positivo è possibile trasformare un programma lineare, (2.16), in un altro programma di più facile computazione. Supponiamo quindi che $v > 0$ e siano $x_i = p_i/v$. Il vincolo $p_1 + ... + p_m = 1$ diviene $x_1 + ... + x_m = 1/v$, ove massimizzare v è equivalente a minimizzare $1/v$. È possibile quindi rimuovere v dal problema minimizzando $x_1 + .. + x_m$. Il problema diviene: scegli $x_1, ..., x_m$, t.c.

$$\text{minimizza} \quad x_1 + .. + x_m$$

con i vincoli

$$\begin{array}{c} 1 \leq \sum_{i=1}^{m} x_i a_{i1} \\ \vdots \\ 1 \leq \sum_{i=1}^{m} x_i a_{in} \\ x_i \geq 0 \quad \forall i \end{array} \tag{2.18}$$

Una volta risolto questo problema, il valore del gioco originale può essere calcolato come $v = 1/(x_1 + ... + x_m)$ e la strategia ottima per il giocatore I sarà $p_i = vx_i \quad \forall i = 1, ..., m$.

2.1.7 Il metodo del simplesso nella soluzione di giochi finiti

Quanto segue è l'applicazione del metodo del simplesso nella soluzione del problema (2.18), formulata in [40] e descritta in [13].

Step 1. Si aggiunga una costante a tutti gli elementi della matrice di gioco se necessario per assicurare che il valore sia positivo (nel caso in cui venga sommata tale costante, essa andrà sottratta al valore della nuova matrice di gioco per ottenere il valore della matrice originale).

Step 2. Si crei un *tableau* aumentando la matrice di gioco con i valori -1 lungo il bordo inferiore, 1 lungo il bordo destro e 0 nell'angolo in basso a destra. Si etichettino sulla sinistra le strategie del giocatore I da x_1 a x_m e in alto quelle del giocatore II, da y_1 a y_n.

	y_1	y_2	$\cdots$	y_n	
x_1	a_{11}	a_{12}	$\cdots$	a_{1n}	1
x_2	a_{21}	a_{22}	$\cdots$	a_{2n}	1
$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$
x_m	a_{m1}	a_{m2}	$\cdots$	a_{mn}	1
	-1	-1	$\cdots$	-1	0

Step 3. Si scelga il *pivot* nel tableau , di riga p e colonna q, in modo che rispetti le seguenti proprietà:
a. La cifra nel bordo inferiore della colonna del pivot ($a_{(m+1),q}$) deve essere negativa.
b. Il pivot $a_{p,q}$ deve essere positivo.
c. La riga del pivot, p, deve essere scelta in modo da ottenere il più piccolo rapporto tra la cifra nel bordo destro della riga del pivot e il pivot stesso ($a_{p,(n+1)}/a_{p,q}$), fra tutti i possibili pivot in quella colonna.

Step 4. Si esegua il pivoting come segue:
a. Si rimpiazzi ogni entrata, $a_{i,j}$, non presente nella riga o nella colonna del pivot, con $a_{i,j} - a_{p,j} \cdot a_{i,q}/a_{p,q}$.
b. Si rimpiazzi ogni entrata nella riga del pivot, eccetto per il pivot stesso, con il risultato della divisione tra essa e il pivot.
c. Si rimpiazzi ogni entrata nella colonna del pivot, eccetto per il pivot stesso, con il risultato della divisione tra il suo negato e il pivot.
d. Si rimpiazzi il pivot con il suo reciproco.

Step 5. Si scambi l'etichetta di sinistra nella riga del pivot con l'etichetta in alto nella colonna del pivot.

Step 6. Se permangono valori negativi nella riga del bordo basso si torni allo step 3.

Step 7. Altrimenti, si è giunti alla soluzione:
a. Il valore v del gioco è il reciproco del numero nell'angolo in basso a destra.
b. La strategia ottima per il giocatore I è composta come segue. Quelle variabili del giocatore I che finiscono nella parte sinistra ricevono probabilità zero, quelle variabili che finiscono nel bordo alto ricevono la probabilità pari al valore nel bordo basso della stessa colonna diviso per la cifra nell'angolo in basso a destra.
c. La strategia ottima per il giocatore II è composta come segue. Quelle variabili del giocatore II che finiscono nel bordo alto ricevono probabilità

zero, quelle variabili che finiscono a sinistra ricevono il valore nella stessa riga del bordo destro diviso per il valore nell'angolo in basso a destra.

2.2 Forma estesa

Esprimere un gioco in forma strategica è un modo compatto di descrivere aspetti matematici del gioco stesso, favorendone l'analisi matematica. La forma strategica però non mette in rilievo aspetti molto importanti di alcuni giochi, come il concetto di mossa e di posizione. È stato quindi formulato un modello matematico, chiamato **forma estesa**, che descrivesse in modo adeguato tali concetti.

La forma estesa di un gioco è modellata utilizzando una struttura dati ad albero, ove la radice è considerata la posizione iniziale del gioco, gli archi sono le mosse e i nodi intermedi sono le posizioni intermedie del gioco; chiaramente i nodi terminali indicano le posizioni finali con il relativo payoff.

Molti giochi introducono il concetto di **mossa aleatoria**, come ad esempio il lancio di dadi o l'estrazione a sorte di numeri o carte. In questi giochi le mosse aleatorie hanno un ruolo fondamentale, persino negli scacchi si fa uso generalmente di una mossa aleatoria nel determinare chi per primo avrà il tratto e quindi la prima mossa, il che è considerato un vantaggio. Si assume che il giocatore sia consapevole delle probabilità che regolano i risultati di una mossa aleatoria.

Nella forma strategica di un gioco, si considera che i giocatori prendano le loro decisioni simultaneamente. Formulare un gioco in forma estesa, quindi tramite un albero, comporta una sequenzialità delle mosse e sembra allontanare la possibilità di permettere scelte contemporanee da parte dei giocatori. A questo proposito si introduce il concetto di **insieme informativo** .

$$\begin{pmatrix} 3 & 0 & 1 \\ -1 & 2 & 0 \end{pmatrix}$$

Figura 2.2: Forma strategica

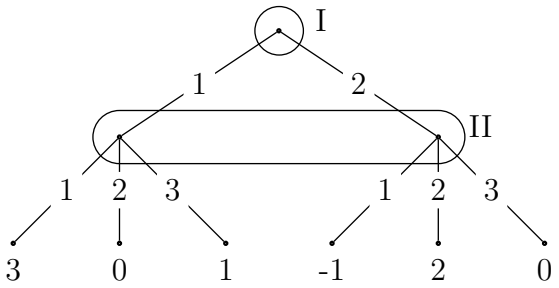


Figura 2.3: Forma estesa

Si consideri il gioco formulato in forma strategica dalla matrice in figura 2.2. Al suo turno, il giocatore II non sa in quale delle due posizioni possibili si trova. Questa situazione viene indicata nel diagramma cerchiando le due

posizioni con una curva chiusa che le contenga entrambe e che rappresenta l'insieme informativo, come mostrato in figura 2.3.

Non tutti gli insiemi di nodi possono formare un insieme informativo. In modo da permettere ad un giocatore di non tenere conto in quale vertice di un dato insieme informativo egli sia giunto, ogni nodo in un insieme informativo deve avere lo stesso numero di archi uscenti. Inoltre gli archi devono avere stesse etichette. Un giocatore che muove da un insieme informativo in realtà sceglie un'etichetta e si presuppone scelga sempre la stessa etichetta dallo stesso insieme informativo.

L'albero di gioco con i payoff, gli insiemi informativi e le etichette sugli archi è conosciuto come **Kuhn Tree**.

Definizione 10 *Un gioco finito a somma zero con due giocatori in forma estesa è dato da:*

1) un albero finito con nodi T,

2) una funzione di utilità che assegna un numero reale ad ogni nodo terminale,

3) un insieme T_0 di nodi non terminali, che rappresentano le posizioni in cui si trovano mosse aleatorie, e per ogni $t \in T_0$ una distribuzione di probabilità sugli archi che partono da t,

4) una partizione dei restanti nodi (non terminali e non in T_0) in due gruppi di insiemi informativi $T_{11}, T_{12}, ..., T_{1k_1}$ per il giocatore I e $T_{21}, T_{22}, ..., T_{2k_2}$ per il giocatore II,

5) per ogni insieme informativo T_{jk} del giocatore j, un insieme di etichette L_{jk} e, per ogni $t \in T_{jk}$, una funzione one-to-one che relazioni L_{jk} nell'insieme di archi che partono da t.

La struttura di un gioco in forma estesa può diventare alquanto complessa. Un tipo di situazione degenere si verifica quando un insieme informativo contiene due nodi che sono collegati insieme da un cammino. In questo caso la convenzione adottata, che vede un giocatore fare una scelta per ogni insieme informativo, rischia di venire meno. La decisione presa dal giocatore deve essere presa ogni volta che l'insieme informativo viene raggiunto, per questo motivo è bene assumere nella definizione di gioco in forma estesa che non vi siano insiemi informativi contenenti due nodi uniti da un cammino nell'albero.

Giochi in cui entrambi i giocatori conoscono le regole, ovvero in cui entrambi i giocatori conoscono il Kuhn tree, vengono chiamati **giochi ad informazione completa** . Giochi in cui uno o entrambi i giocatori non conoscono qualche payoff, o qualche probabilità delle mosse aleatorie, o qualche insieme informativo, o ancora ramificazioni dell'albero di gioco, prendono il nome di **giochi ad informazione incompleta**, o **pseudogiochi**.

2.2.1 Trasformazioni in forma estesa e in forma strategica

Il passaggio da forma strategica a forma estesa è piuttosto semplice ed è stato affrontato nell'esempio in figura 2.2 e 2.3, al contrario passare dalla forma estesa alla forma strategica richiede qualche considerazione ulteriore.

Dato un gioco in forma estesa, si trovano inizialmente X e Y, gli insiemi di strategia pure dei giocatori impiegate nella forma matriciale. Una strategia pura per il giocatore I è una regola che gli dice esattamente quale debba essere la mossa da scegliere in ogni insieme informativo. Siano $T_{11}, ..., T_{1k_1}$ gli insiemi informativi per il giocatore I e siano $L_{11}, ..., L_{1k_1}$ le corrispondenti etichette. Una strategia pura per I è una tupla di k_1 elementi $\mathbf{x} = (x_1, ..., x_{k_1})$ dove per ogni i, x_i è un elemento di L_{1i}. Se in L_{1i} ci sono m_i elementi, il numero delle k_1-tuples e, di conseguenza il numero delle strategie pure per I, è il prodotto $m_1 m_2 \cdots m_k$. L'insieme di tutte queste strategie è X. In modo analogo è possibile ricavare l'insieme delle strategie pure per il giocatore II, Y.

Per poter considerare le mosse aleatorie anche nella forma strategica si fa uso di una convenzione che definisce *payoff casuali* come segue. Dati $\mathbf{x} \in X$ e $\mathbf{y} \in Y$ si suppone che un arbitro giochi la mossa appropriata da $\mathbf{x}$ quando il gioco raggiunge un insieme informativo per il giocatore I, giochi la mossa appropriata da $\mathbf{y}$ quando il gioco raggiunge un insieme informativo per il giocatore II, e giochi la mossa seguendo le giuste probabilità se si tratta di una mossa aleatoria. Il risultato del gioco dati $\mathbf{x} \in X$ e $\mathbf{y} \in Y$ dipende quindi dalla mossa aleatoria selezionata, ed è quindi una quantità casuale.

Normalmente i payoff casuali dovuti all'uso di strategie miste da parte dei giocatori vengono sostituiti con la loro media attesa; si accetta quindi, la stessa convenzione per i payoff casuali dovuti all'uso di mosse aleatorie.

Convenzione *Se per le strategie pure dei giocatori, $\mathbf{x} \in X$ e $\mathbf{y} \in Y$, il payoff è una quantità casuale, si rimpiazza il payoff con il valore medio atteso e si denota questa media con $L(\mathbf{x}, \mathbf{y})$.*

Ad esempio, se dati $\mathbf{x} \in X$ e $\mathbf{y} \in Y$, il giocatore I vince 3 con probabilità $1/4$, vince 1 con probabilità $1/4$ e perde 1 con probabilità $1/2$, allora il payoff medio è dato da $\frac{1}{4}(3) + \frac{1}{4}(1) + \frac{1}{2}(-1) = 1/2$, quindi $L(\mathbf{x}, \mathbf{y}) = 1/2$.

Quindi, dato un gioco in forma estesa, si dice che (X, Y, L) è l'equivalente forma strategica del gioco se X e Y sono gli spazi delle strategie pure dei giocatori I e II rispettivamente, e se $L(\mathbf{x}, \mathbf{y})$ è il payoff medio per $\mathbf{x} \in X$ e $\mathbf{y} \in Y$.

2.2.2 Giochi ad informazione perfetta

Una volta definita la forma estesa di un gioco, è possibile approfondire il concetto di informazione perfetta.

Definizione 11 *Un gioco ad informazione perfetta è un gioco nella cui forma estesa ogni insieme informativo di ogni giocatore contiene un singolo nodo.*

In un gioco ad informazione perfetta, ogni giocatore che deve fare la mossa conosce l'esatta posizione in cui si trova nell'albero di Kuhn. In particolare, ogni giocatore conosce tutte le mosse passate del gioco incluse le mosse aleatorie. Sono ad informazione perfetta giochi quali la dama, gli scacchi, il backgammon, etc.

I giochi ad informazione perfetta hanno una particolare struttura matematica piuttosto semplice. Il risultato principale è che *ogni gioco ad informazione perfetta ridotto alla forma strategica presenta un "saddle point"*, quindi entrambi i giocatori hanno strategie ottime pure. Inoltre, *i "saddle point" possono essere trovati rimuovendo righe e colonne dominate.* Questo ha un'interessante implicazione ad esempio per il gioco degli scacchi. Poiché non ci sono mosse aleatorie, ogni elemento della matrice per gli scacchi deve essere o $+1$ (vittoria per il giocatore I), o -1 (vittoria per il giocatore II), o 0 (parità). Uno di questi numeri deve essere saddle point, quindi o il giocatore I può garantirsi la vittoria, o il giocatore II può garantirsi la vittoria, oppure entrambi possono garantirsi almeno il pareggio. Dal punto di vista puramente teorico, il gioco degli scacchi è un gioco molto semplice. Se, dopo aver scritto la matrice di gioco, c'è una riga con elementi tutti $+1$, il giocatore I vince, se c'è una colonna di elementi -1, il giocatore II vince, altrimenti c'è una riga con elementi $+1$ e 0 e una colonna con elementi -1 e 0 e il gioco finisce in patta con giocata migliore. Chiaramente, in pratica nel gioco degli scacchi non c'è alcuna speranza di trovare sempre una strategia ottima e, in effetti, non si è ancora capito come l'uomo possa tanto bene giocare questo gioco.

2.2.3 Strategie comportamentali

Sfortunatamente, algoritmi che utilizzano la formulazione di un gioco in forma normale sono utilizzabili nella pratica solo per piccoli giochi. Questo è dovuto al fatto che la forma normale è esponenziale sulla dimensione dell'albero di gioco. Una strategia deterministica deve specificare un azione per ogni insieme informativo, quindi il numero totale di possibili strategie

è anch'esso esponenziale sul numero di insiemi informativi, che è di norma correlato alla dimensione dell'albero di gioco [16].

Per i giochi in forma estesa è utile esaminare un metodo differente nella scelta di strategie pure. Ciò di cui necessita un giocatore è scegliere, secondo una certa distribuzione, un arco uscente per ogni insieme informativo. Una **strategia comportamentale** è una strategia che assegna ad ogni insieme informativo una distribuzione di probabilità sulle scelte di quell'insieme.

Si supponga ad esempio che venga distribuita una carta da un mazzo di 52 al giocatore I. Dopo averla vista, egli può scommettere o lasciare, quindi è il turno del giocatore II. Il giocatore I ha 52 insiemi informativi, ognuno con 2 scelte, quindi ha 2^{52} strategie pure. D'altra parte, una strategia comportamentale per I è data dalle probabilità di scommettere per ogni carta che può aver ricevuto, e quindi è composta da solo 52 numeri.

In generale, le dimensioni dello spazio delle strategie comportamentali è molto più piccolo che quello delle strategie miste. Un importante teorema di Kuhn del 1953, enuncia che nei giochi finiti con **ricordo perfetto**, ogni distribuzione sui payoff ottenuta con una strategia mista, può esserlo anche con una strategia comportamentale.

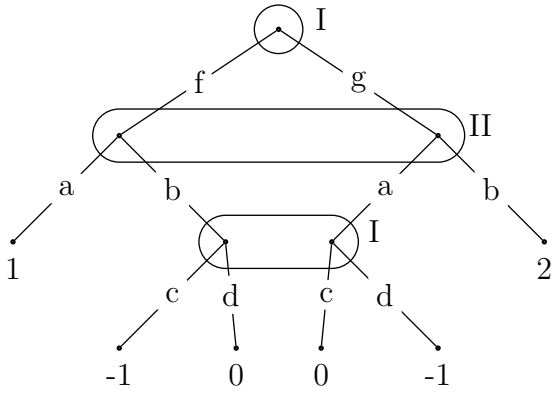


Figura 2.4: esempio

L'albero di gioco nella forma estesa può descrivere situazioni in cui un giocatore ha dimenticato una mossa fatta precedentemente. Si consideri ad esempio quello in figura 2.4, nel secondo insieme informativo il giocatore I non tiene conto della mossa seguita dal primo insieme informativo, ha perciò "dimenticato" la storia delle mosse fatte.

I giochi in cui i giocatori ricordano tutta l'informazione sul passato e tutte le mosse da loro fatte sono chiamati giochi con **ricordo perfetto**.

Per osservare che le strategie comportamentali non sono sempre sufficien-

ti, consideriamo il gioco con ricordo imperfetto di figura 2.4. Dopo averlo ridotto in forma strategica, si ottiene la matrice

$$\begin{array}{c} \\ (f,c) \\ (f,d) \\ (g,c) \\ (g,d) \end{array} \begin{array}{c} \begin{array}{cc} a & b \end{array} \\ \begin{pmatrix} 1 & -1 \\ 1 & 0 \\ 0 & 2 \\ -1 & 2 \end{pmatrix} \end{array}$$

La prima e la quarta riga possono essere eliminate per dominanza e dalla sottomatrice quadrata risultante è facile vedere che le strategie miste ottime per il giocatore I e per il giocatore II sono $(0, 2/3, 1/3, 0)$ e $(2/3, 1/3)$ rispettivamente. Il valore del gioco è $2/3$. Una strategia comportamentale per I è data invece da due numeri, p_f, la probabilità di scegliere f nel primo insieme informativo, e p_c, la probabilità di scegliere c nel secondo. Si ottiene pertanto la strategia mista $(p_f p_c, p_f(1-p_c), (1-p_f)p_c, (1-p_f)(1-p_c))$. La strategia $(0, 2/3, 1/3, 0)$ non è di questa forma, poiché, se il primo componente fosse zero, allora o il secondo o il terzo componente sarebbe zero.

2.3 Giochi ricorsivi e stocastici

In questo paragrafo affronterò l'interessante caso di giochi che hanno al loro interno sottogiochi. Se la matrice di un gioco G ha altri giochi per componenti, la soluzione di G è la soluzione del gioco la cui matrice è ottenuta rimpiazzando ogni sottogioco nella matrice di G per il suo valore. Chiaramente, un gioco che è componente di una matrice può avere anch'esso giochi come componenti, in tal caso il metodo precedente può essere reiterato per ottenere la soluzione. Questo funziona se ci sono un numero finito di passaggi tra sottogiochi.

In alcuni giochi può capitare che il gioco originale si ripresenti sottoforma di sottogioco. Tali giochi sono chiamati **ricorsivi**. Si consideri la seguente matrice di gioco

$$G = \begin{pmatrix} G & 1 \\ 1 & 0 \end{pmatrix}$$

Questo è un gioco infinito. Se i giocatori giocano sempre la prima riga e la prima colonna, il gioco continua per sempre. Diciamo che il giocatore II paga al giocatore I, Q se entrambi scelgono le loro prime strategie pure per sempre e scriviamo

$$G = \begin{pmatrix} G & 1 \\ 1 & 0 \end{pmatrix}, \quad Q$$

Il giocatore II può restringere le sue perdite ad al più 1 scegliendo la seconda colonna. Se $Q \geq 1$, I può garantirsi la vittoria di almeno 1 giocando la sua prima riga per sempre, ma se $Q < 1$, non esiste una strategia ottima per I che gli garantisca la vittoria di almeno 1.

Diciamo che per ogni $\epsilon > 0$ c'è una strategia per I che gli garantisce un guadagno medio di almeno $1 - \epsilon$. In questo caso, la strategia mista $(1 - \epsilon, \epsilon)$ è chiamata ϵ-ottima per I. Essa assicura che prima o poi I sceglierà la riga 2, limitando il payoff tra 0 e 1 e mai Q. Il meglio che può fare il giocatore II contro tale strategia è scegliere la seconda colonna subito, sperando che I scelga la riga 2. Il payoff atteso diviene quindi $1 \cdot (1 - \epsilon) + 0 \cdot \epsilon = 1 - \epsilon$. Riassumendo, per il gioco G, il valore è 1, il giocatore II ha una strategia ottima, ossia la colonna 2; se $Q \geq 1$, la prima riga è ottima per I, altrimenti non c'è strategia ottima per I, ma la strategia $(1 - \epsilon, \epsilon)$ è ϵ-ottimale per I.

La strategia che garantisce ad un giocatore un payoff medio dipendente da ϵ è chiamata **ϵ-ottimale**.

È possibile calcolare il valore v dei giochi ricorsivi, in particolare di G, sostituendo v alla chiamata ricorsiva nella matrice e risolvendo l'equazione

$$v = \quad \text{Val}\begin{pmatrix} v & 1 \\ 1 & 0 \end{pmatrix}$$

È possibile che l'equazione abbia più di una soluzione, tra queste la soluzione più vicina al valore di Q è il valore del gioco.

È possibile generalizzare la nozione di gioco ricorsivo, aggiungendo alla scelta del gioco successivo l'elemento stocastico. Siano $G_1, ..., G_n$ giochi e $p_1, ..., p_n$ probabilità. Si usa la notazione $p_1G_1 + ... + p_nG_n$ per indicare la situazione dove il prossimo gioco che deve essere giocato è scelto a caso secondo la proporzione in cui G_i è scelto con probabilità p_i, per $i = 1, .., n$.

Poiché, dato un numero z, la matrice 1×1 (z) indica un gioco banale in cui II paga a I z, è lecito, ad esempio, usare $\frac{1}{2}G_1 + \frac{1}{2}(3)$ per rappresentare la situazione in cui G_1 è giocato se il lancio di una moneta non truccata da esito 'testa', e II paga a I 3 altrimenti.

Se, infine, viene aggiunta la possibilità di pagare un payoff ad ogni passo prima che il gioco finisca, si parla di **giochi stocastici**. Un gioco stocastico, **G**, consiste di un insieme finito di posizioni, $\{1, 2, ..., N\}$, uno dei quali è specificato come la posizione di partenza. Si indica con $G^{(k)}$ il gioco in cui k è la posizione iniziale. Ad ogni posizione è associata una matrice $\mathbf{A}^{(k)} = \left(a_{ij}^{(k)}\right)$. Se il gioco stocastico è nella posizione k, i giocatori scelgono simultaneamente

una riga e una colonna di $\mathbf{A}^{(k)}$, diciamo i e j, quindi il giocatore I vince l'ammontare di $a_{ij}^{(k)}$. Dopodiché, con probabilità che dipendono da i, j e k, il gioco o termina, o continua in un'altra posizione, che può essere anche la stessa appena giocata. La probabilità che il gioco termini è indicata da $s_{ij}^{(k)}$ e la probabilità che la posizione successiva sia l è indicata da $P_{ij}^{(k)}(l)$, dove $s_{ij}^{(k)} + \sum_{l=1}^{N} P_{ij}^{(k)}(l) = 1$ per ogni i, j e k.

Per essere sicuri che prima o poi il gioco termini, si assume che tutte le probabilità di terminazione siano positive. Il gioco in cui il giocatore I vuole massimizzare il payoff totale accumulato e il giocatore II lo vuole minimizzare, è descritto come segue

$$G^{(k)} = \left(a_{ij}^{(k)} + \sum_{l=1}^{N} P_{ij}^{(k)}(l) G^{(l)} \right)$$

È chiaro che, a differenza dei giochi visti in precedenza, il pagamento di un payoff non implica la terminazione del gioco. Dopo che il pagamento è avvenuto, viene deciso casualmente quale gioco seguirà o se ci sarà terminazione. Poiché non è possibile fissare un limite superiore alla lunghezza del gioco, si può parlare di gioco infinito. Per ogni n i giocatori sono tenuti ad indicare le loro decisioni nella posizione n. In giochi di questo tipo la teoria non garantisce un valore, inoltre, la scelta di una strategia al passo n può dipendere da ciò che è successo in tutti i passi precedenti, quindi lo spazio delle possibili strategie diviene estremamente complesso.

Ciò nonostante, per ogni posizione iniziale, esiste il valore del gioco ed esistono strategie ottime per i giocatori, che indicano le stesse probabilità per le scelte di un giocatore ogni volta che si raggiunge la stessa posizione, senza considerare ciò che è avvenuto prima. Tali strategie vengono chiamate **strategie stazionarie**.

Teorema 6 *Ogni gioco $G^{(k)}$ ha un valore, $v(k)$. Questi valori sono la soluzione unica dell'insieme di equazioni*

$$v(k) = Val \left(a_{ij}^{(k)} + \sum_{l=1}^{N} P_{ij}^{(k)}(l) v(l) \right) \qquad per\ k = 1, .., N. \tag{2.19}$$

Ogni giocatore ha una strategia stazionaria ottima che nella posizione k usa la strategia mista ottima per il gioco con matrice

$$A^{(k)}(\mathbf{v}) = \left(a_{ij}^{(k)} + \sum_{l=1}^{N} P_{ij}^{(k)}(l) v(l) \right)$$

dove $\mathbf{v}$ *rappresenta il vettore di valori* $\mathbf{v} = (v(1), ..., v(N))$.

Per un generico gioco stocastico con molti stati l'equazione (2.19) diviene un sistema piuttosto complesso di equazioni non lineari, è quindi impossibile pensare di risolverlo per ogni sistema. Un semplice metodo iterativo, che si basa sulla dimostrazione di Shapley del teorema (6), permette di approssimare la soluzione. Esso prende il nome di **Shapley iteration**.

Per prima cosa si imposta un vettore iniziale $\mathbf{v}_0 = (v_0(1), ..., v_0(N)) = \mathbf{0}$. Dato $\mathbf{v}_n$, si definisce induttivamente $\mathbf{v}_{n+1}$ tramite l'equazione

$$v_{n+1}(k) = \text{Val}\left(a_{ij}^{(k)} + \sum_{l=1}^{N} P_{ij}^{(k)}(l) v_n(l)\right) \qquad \text{per } k = 1, ..., N.$$

Con $\mathbf{v}_0 = \mathbf{0}$, $v_n(k)$ è il valore del gioco stocastico che comincia nello stato k se avviene la terminazione quando il gioco raggiunge il passo n. La dimostrazione del teorema (6) mostra che $v_n(k)$ converge al vero valore, $v(k)$ del gioco stocastico che comincia allo stato k.

Capitolo 3

Giochi ad informazione imperfetta

3.1 Il gioco del Poker

3.1.1 Finale base nel gioco del Poker

Quello presentato è lo studio di una situazione che si può verificare nel gioco del Poker nell'ultimo round di scommesse quando sono rimasti due giocatori soltanto. Questo sottolinea un aspetto importante del gioco del Poker, come accade per gli Scacchi o il Kriegspiel, ovvero sottolinea il fatto che è possibile imbattersi in fasi finali di gioco, in cui strategie e tattiche trattabili analiticamente divengono importanti [13].

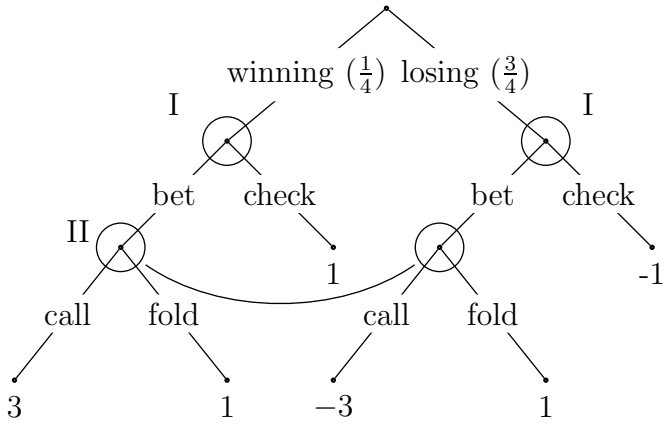


Figura 3.1: "Finale base" del Poker

Questo finale è giocato come segue. Ogni giocatore punta \$1, quindi il giocatore I riceve una carta dal mazzo e la mantiene segreta. Questa è una carta vincente con probabilità di 1/4 e una carta perdente con probabilità di 3/4. Quindi il giocatore I passa (check) o punta (bet). Se decide di passare, deve mostrare la carta: se è vincente vince la puntata, altrimenti la puntata va al giocatore II. Se I scommette, punta \$2 aggiuntivi nel piatto, quindi il giocatore II, non conoscendo quale carta abbia in mano il giocatore I, deve chiamare o lasciare. Se lascia, II perde ciò che aveva puntato (\$1) ad I a prescindere dalla carta dell'avversario; se chiama, aggiunge \$2 al banco. Infine la carta del giocatore I viene mostrata e I vince \$3 da II se ha una carta vincente, perde \$3 altrimenti.

In figura 3.1 è mostrato l'albero di gioco in forma estesa. Il giocatore I ha due insiemi informativi, e in entrambi egli deve scegliere tra due opzioni; quindi ha $2 \cdot 2 = 4$ strategie pure:

(b, b): punta sia con carta vincente che con con carta perdente, (b, c): punta con carta vincente e passa con carta perdente; (c, b): passa con carta vincente e punta con carta perdente; (c, c): passa sempre.
Quindi, $X = \{(b, b), (b, c), (c, b), (c, c)\}$ è l'insieme di tutte le strategie per il giocatore I, siano esse buone o cattive strategie[1]. Il giocatore II ha un solo set informativo [2], quindi $Y = \{c, f\}$ ove

c: se I punta, II chiama; f: se I punta, II lascia.

Si supponga che I usi (b, b) e II usi c. Allora, se I riceve una carta vincente (che capita con probabilità pari a 1/4), egli punta, il giocatore II chiama e I vince \$3. Se però, I riceve una carta perdente (con probabilità pari a 3/4),egli punta, II chiama e I perde \$3. Il guadagno atteso per il giocatore I è pari a

$$L((b, b), c) = \frac{1}{4}(3) + \frac{3}{4}(-3) = -\frac{3}{2}$$

. Calcolando anche i restanti payoff, si ottiene la seguente matrice

$$\begin{array}{c} \\ (b,b) \\ (b,c) \\ (c,b) \\ (c,c) \end{array} \begin{array}{c} \begin{array}{cc} \text{c} & \text{f} \end{array} \\ \begin{pmatrix} -3/2 & 1 \\ 0 & -1/2 \\ -2 & 1 \\ -1/2 & -1/2 \end{pmatrix} \end{array}$$

Si noti che la terza riga è dominata dalla prima e la quarta è dominata dalla seconda. Questo significa, come era prevedibile, che se il giocatore I riceve la carta vincente, allora non gli conviene passare, poiché puntando

[1] ad esempio (c, b) è una strategia piuttosto scadente

[2] in figura è descritto da una linea curva che unisce i due cerchi rappresentanti l'insieme

sarà sicuro di vincere almeno altrettanto, e forse di più. La restante matrice, dopo aver eliminato le strategie dominate, ha valore $V = -1/4$. La strategia ottima per I consiste nel giocare (b, b) con probabilità di $1/6$ e (b, c) con probabilità di $5/6$, mentre la strategia ottima per II consiste nel giocare c e f con probabilità pari a $1/2$ entrambe. È interessante osservare che la strategia ottima per il giocatore I ricorre al bluff, che corrisponde a (b, b).

3.1.2 Il modello di Kuhn

A due giocatori viene distribuita una carta scelta a caso da un mazzo di tre $\{1, 2, 3\}$. Il giocatore I può passare (check) o puntare (bet). Se I punta, II può chiamare (call) o lasciare (fold). Se I passa, II può passare a sua volta o puntare. Se I passa e II punta, allora I può chiamare o lasciare. Se entrambi passano il giocatore con carta più alta vince 1. Se un giocatore punta e l'altro lascia, il giocatore che ha puntato vince 1. Se un giocatore punta e l'altro chiama, il giocatore con carta più alta vince 2 [13].

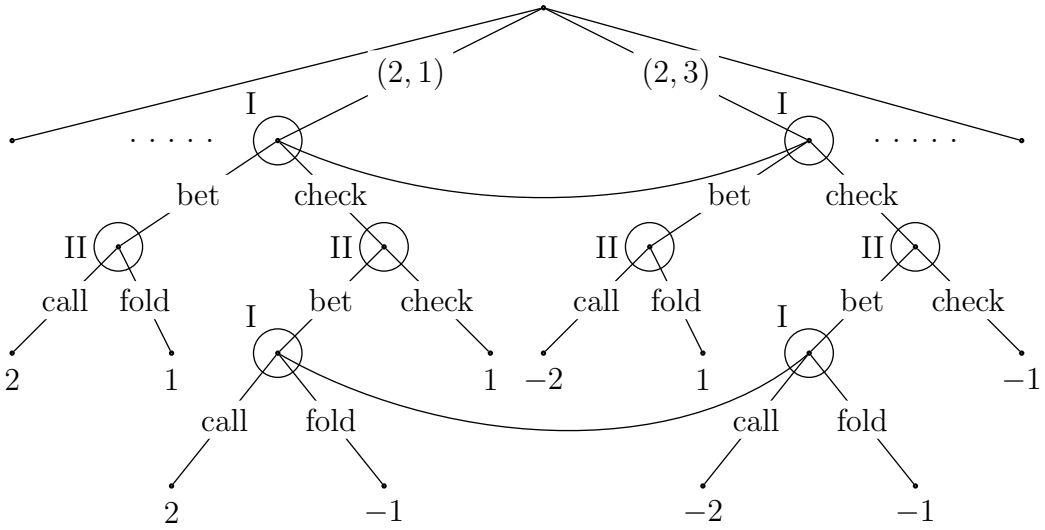


Figura 3.2: Modello del Poker di Kuhn

In figura 3.2 è riportato un particolare dell'albero di gioco in forma estesa, dove il giocatore I riceve la carta 2, mentre il giocatore II riceve la carta 1 o la carta 3. La prima mossa è quindi una mossa aleatoria che ha sei possibili risultati. In figura sono mostrati i due insiemi informativi per I nel caso in cui egli riceva la carta 2, analogamente quindi ci saranno altri due insiemi informativi, per la carta 1 e per la carta 3 rispettivamente. In totale il

giocatore I ha 6 insiemi informativi e, per ogni insieme, egli può scegliere tra due possibilità, si hanno pertanto 2^6 strategie pure.

Generalizzando il modello a k carte, si ha che gli insiemi informativi sono $2k$ e avendo 2 opzioni per ogni insiemi informativo si hanno $2^{2k} = 4^k$ strategie pure. Poiché la forma strategica ha una riga per ogni strategia pura del giocatore I e una colonna per ogni strategia pura del giocatore II, è essa stessa esponenziale in k. Si osservi invece che la dimensione dell'albero di gioco in forma estesa è pari a $9k + 1$, ove 1 indica il nodo radice.

In generale la conversione alla forma normale è esponenziale sia in termini di tempo che di spazio.

Un approccio differente per risolvere giochi ad informazione imperfetta è stato descritto in [12]. tale approccio utilizza la conversione ad una forma alternativa chiamata **sequence form**. In essa, piuttosto che rappresentare le probabilità delle strategie pure, si preferisce rappresentare i pesi di differenti sequenze di mosse. Una sequenza, per un giocatore, corrisponde essenzialmente ad un cammino nell'albero e isola le mosse sotto il diretto controllo del giocatore, ignorando mosse aleatorie e le decisioni dell'avversario. Nel modello di Kuhn con k carte ad esempio, il giocatore I avrebbe $4k+1$ sequenze. Oltre la sequenza vuota[3] egli ha 4 sequenze per ogni carta c: [puntare su c], [passare con c], [passare con c e puntare nell'ultimo round] e [passare con c e lasciare nell'ultimo round]. Allo stesso modo anche il giocatore II ha $4k + 1$ sequenze, oltre quella vuota, per ogni carta d ha 4 sequenze: [punta con d dopo che l'avversario ha passato], [passa con d dopo che l'avversario ha passato], [chiama con d dopo che l'avversario ha puntato] e [lascia con d dopo che l'avversario ha puntato].

Data una strategia comportamentale, il peso di una sequenza per un giocatore è il prodotto delle probabilità delle mosse del giocatore contenute nella sequenza. La probabilità che un cammino sia percorso durante il gioco è il prodotto dei pesi di tutte le sequenze dei giocatori per quel cammino, fratto la probabilità di tutte le mosse aleatorie nel cammino.

La *sequence form* di un gioco a due giocatori consiste di una matrice dei payoff $\mathbf{A}$ e un sistema lineare di vincoli per ogni giocatore. In un gioco a due giocatori, l'i-esima riga di $\mathbf{A}$ corrisponde ad una sequenza σ_I^i per il giocatore I, e la j-esima colonna ad una sequenza σ_{II}^j per il giocatore II. L'elemento a_{ij} è la somma pesata dei payoff nelle foglie raggiunte da questa coppia di sequenze[4]. Se una coppia di sequenze non è coerente con alcun cammino, l'elemento nella matrice è 0. Quindi, ad esempio, l'elemento della matrice per la coppia di sequenze ([punta con 2], [passa con 1 dopo che l'avversario

[3] essa corrisponde alla radice dell'albero

[4] esse sono pesate per le probabilità delle mosse aleatorie incontrate nel cammino

ha puntato]) è 1. L'elemento per ([punta con 2], [passa con 1 dopo che l'avversario ha passato]) è 0, poiché questa coppia non è coerente con alcuna foglia.

Per ogni sequenza σ_I si avrà una variabile x_{σ_I}, e una variabile $y_{\sigma_{II}}$ per ogni sequenza σ_{II} di II. Si può mostrare ([12]) che il payoff del gioco atteso $L(\mathbf{x}, \mathbf{y})$ è pari a $\mathbf{x^T A y}$, che è analogo a quanto ottenuto per la forma strategica. I vincoli per $\mathbf{x}$ e $\mathbf{y}$ devono infine garantire che essi rappresentino delle strategie. Per la forma normale essi asseriscono che i vettori rappresentano delle distribuzioni di probabilità; in questo caso, i vincoli sono derivati dal fatto seguente: se σ è la sequenza per il giocatore i, che porta ad un insieme informativo in cui il giocatore i deve muovere, e $m_1, ..., m_k$ sono le possibili mosse nell'insieme informativo, allora deve essere verificato che $x_\sigma = x_{\sigma m_1} + ... + x_{\sigma m_k}$. Infine, l'ultimo vincolo impone che il peso della sequenza vuota sia pari a 1[5], e che $x_\sigma \geq 0 \ \ \forall \sigma$.

Questa forma è al più lineare nella dimensione dell'albero di gioco, poiché c'è al più una sequenza per ogni cammino nell'albero e un vincolo per ogni insieme informativo. Può essere quindi generata facilmente tramite una visita dell'albero.

Si osservi che la matrice risultante della *sequence form* può essere risolta con un algoritmo standard di programmazione lineare, quale, ad esempio, l'algoritmo del simplesso.

3.2 Gli scacchi invisibili o Kriegspiel

3.2.1 Le origini

Il Kriegspiel fu inventato, sembra, nel 1899 dal giocatore H.M. Temple, cui era stato chiesto di inventare un gioco che imitasse la guerra, in cui la posizione avversaria è all'inizio poco conosciuta e si rivela a poco a poco.

L'idea è semplice: ciascun giocatore dispone di una normale scacchiera con i relativi pezzi e non vede la scacchiera che è a disposizione dell'avversario. Sulla propria scacchiera è tenuto a lasciare i propri pezzi nella posizione raggiunta mentre può disporre i pezzi avversari come e dove crede, cercando di immaginare l'esatta posizione dell'avversario. L'arbitro dispone di una terza scacchiera invisibile ai due giocatori, sulla quale ricostruisce l'esatta posizione. In genere conviene disporre i due giocatori e l'arbitro come in figura 3.3, ove le frecce indicano la direzione dello sguardo ([19]).

Con le regole di Temple, il Kriegspiel raggiunse una certa popolarità nella prima metà del XX secolo in Inghilterra, in Olanda e negli USA, dove

[5]La sequenza vuota ha peso 1 poiché è raggiunta in ogni cammino del gioco

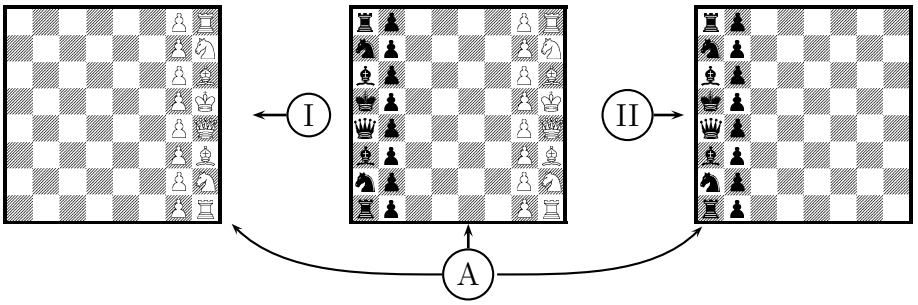


Figura 3.3: Disposizione iniziale

venne chiamato da autori diversi in vari modi: *War Chess*, *Screen Chess*, *Commando Chess* o *Invisible Chess* ([9]). Dalla fine della Seconda Guerra Mondiale viene giocato assiduamente presso l'istituto RAND, un centro di ricerche strategiche finanziato dal governo USA. È ben documentata la passione per il Kriegspiel di alcuni grandi matematici che si trovavano in visita al RAND Institute, primi tra tutti John von Neumann, John Nash (Premio Nobel per l'Economia) e Lloyd Shapley. La passione di questi ed altri matematici più o meno famosi per il Kriegspiel si può spiegare per il fatto che questo gioco richiede in molti casi, soprattutto nel finale di partita, un trattamento matematico basato sulla Teoria dei Giochi ([9]).

3.2.2 Le regole

Riporto in questa sezione le differenti proposte circa le regole del Kriegspiel così come sono state dettagliatamente descritte e riunite in [9].

Storicamente la Gran Bretagna e gli Stati Uniti sono i paesi dove tale gioco è più diffuso. Vi sono principalmente due differenti insiemi di regole, classificati in due famiglie. La prima rappresenta le cosiddette regole inglesi o Eastern rules (regole della East Coast), la seconda le regole americane o Western rules (regole in vigore in California) [31, 21]. Le regole descritte da J.D. Wilkins in [39] sono state usate per anni al *RAND Institute*. Le regole di ICC sono derivate dalle regole RAND, sebbene siano state introdotte alcune varianti che rendono il gioco tramite Internet leggermente più complesso.

Il giocatore che deve muovere sceglie la mossa da tentare. In sostanza deve considerare l'insieme delle proprie mosse pseudolegali, comprese tutte le catture di pedone che crede possibili, scegliendone una, che comunica all'arbitro come il suo "tentativo" (si ricordi che il tentativo non viene comunicato all'altro giocatore). Ad esempio, nella posizione in figura 3.4 le mosse possibili per il Bianco sono ♗b2,♗a3,♗d2,♗e3, ♗f4,♗g5,♗h6, ♔d1, ♔d2,♔e2,♔f2,♔f1, d5,dc5,de5.

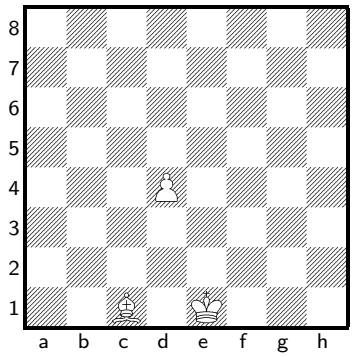


Figura 3.4: Posizione d'esempio

L'arbitro, che conosce la lista delle mosse legali di entrambi i giocatori, risponde ad ogni tentativo con uno dei seguenti messaggi.

L'arbitro dice: "partita terminata". Ovviamente se la lista delle mosse legali è vuota significa che la posizione è di stallo o scacco matto e la fine della partita viene immediatamente annunciata dall'arbitro.

L'arbitro è silente. La mossa passa all'avversario. Supponiamo che la mossa scelta dal Bianco sia legale e che non dia scacco né catturi un pezzo: l'arbitro dice solo "Mossa al Nero". Diremo nel seguito che l'arbitro è silente, ovvero che non ha messaggi per i due giocatori.

L'arbitro dice: "mossa illegale". L'arbitro, che vede la posizione completa, conosce tutte le mosse legali. Quindi o ammette il tentativo, in quanto legale, o lo respinge, in quanto illegale. In caso di tentativo respinto l'arbitro annuncia a voce alta "mossa illegale" ed il giocatore fa un altro tentativo.

Ad esempio, nella posizione del diagramma in figura 3.5 (vista sulla scacchiera dell'arbitro) il Bianco potrebbe tentare la mossa ♗h6.

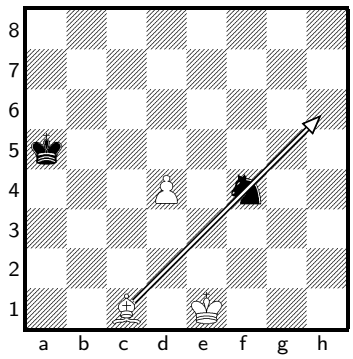


Figura 3.5: Tentativo di mossa che risulta illegale

L'arbitro rifiuta il tentativo ed il Bianco deve trarre due deduzioni alternative: o sulla strada per h6 c'è almeno un ostacolo, oppure l'alfiere è inchiodato sul Re da un pezzo nero in a1 o b1. Potrebbe provare adesso ♗g5 ed otterrebbe un analogo rifiuto. Da notare che anche se prova ♔e2 riceve un rifiuto perché la casa e2 è sotto il controllo di un pezzo nero. In caso di tentativo accettato l'arbitro annuncia "Il Bianco ha mosso" ed il tratto passa all'avversario. Nell'esempio, se il Bianco gioca ♗f4, l'arbitro annuncia "Il Bianco ha mosso: cattura in f4".

L'arbitro dice: "mossa impossibile". Se un giocatore fa un tentativo al di fuori della sua lista di mosse pseudolegali, in sostanza una mossa impossibile, l'arbitro dice "mossa impossibile" ed il giocatore deve fare un altro tentativo. Nell'esempio, una mossa impossibile potrebbe essere ♔e3. *Nota*: la definizione di mossa impossibile è secondo alcuni soggetta al giudizio dell'arbitro. Ad esempio, se l'ultima mossa del Nero prima del diagramma fosse stata "cattura in f4", se il Bianco tenta ♗g5 o ♗h6 l'arbitro dovrebbe dire "mossa impossibile" e non "mossa illegale", perché il Bianco dovrebbe tener conto che in f4 c'è qualcosa. Tuttavia questo è un punto controverso delle regole del Kriegspiel, e in genere viene lasciato al giudizio dell'arbitro.

L'arbitro dice: "scacco". Se una mossa accettata dà scacco, l'arbitro annuncia lo scacco e dice anche da quale direzione: su riga, su colonna, su piccola diagonale, su grande diagonale, oppure di cavallo. Si noti che non viene annunciato quale pezzo dà scacco (anche se in certi casi sarà facile dedurlo). In caso di scacco doppio si annunciano tutte le "direzioni" di scacco.

Variante: nello scacco in diagonale non si annuncia se grande o piccola; inoltre, in caso di scacco doppio viene annunciata solamente la direzione di scacco da parte del pezzo di maggior valore (nella scala Q,R,B,N).

L'arbitro dice: "cattura". L'arbitro annuncia anche le mosse di cattura, però annunciando solamente in quale casa avviene la cattura: per esempio "il Bianco cattura in e4". Nel caso di cattura en-passant secondo le regole inglesi questa si annuncia: "il Bianco cattura en-passant in f6". Invece su ICC le catture en-passant non si annunciano.

Variante RAND: l'arbitro dice anche se l'oggetto catturato è un pedone oppure un pezzo (ma non che tipo di pezzo).

Variante olandese: l'arbitro dice anche se l'entità che cattura è un pezzo oppure un pedone.

Questa lista descrive i messaggi dell'arbitro. Oltre ad essi, in ogni versione del Kriegspiel si dà la possibilità di ricevere un'informazione ulteriore da parte dell'arbitro, laddove sono possibili catture di pedone.

Il giocatore chiede: "Are there any?". Dato che ad ogni turno la

lista delle mosse pseudolegali, comprese le possibili catture di pedone, è di solito molto lunga, prima di iniziare i suoi tentativi un giocatore può chiedere: "ci sono catture di pedone?" (in inglese, "Are there any?"). L'arbitro risponde "No" se non sono possibili catture di pedone, oppure "Prova!". In quest'ultimo caso il giocatore è obbligato a fare almeno un tentativo di cattura (che non sia una mossa impossibile).

Variante: per velocizzare il gioco, l'arbitro annuncia prima di ogni mossa se nella lista delle mosse legali ci sono o no possibili catture di pedone; in tal caso non c'è obbligo per il giocatore di tentare una cattura di pedone.

Variante: Le cosiddette "RAND rules" prevedono che l'arbitro elenchi per ogni semimossa tutte le case in cui è possibile una cattura di pedone. Questa è forse la versione più facile da giocare.

Variante: Su Internet l'arbitro annuncia solo il numero di catture disponibili.

Note:

1. L'arbitro non annuncia mai, in alcuna variante, se un pezzo può catturare qualcosa;

2. le mosse speciali di arrocco o promozione non vengono annunciate; all'inizio della partita occorre avere materiale di riserva, in modo da non perdere tempo per cercare il pezzo di promozione;

3. l'arbitro non può in alcun caso riassumere quale materiale sia rimasto in gioco.

Variante RAND: su richiesta, l'arbitro riassume il numero totale di pedoni e pezzi (ma non il loro tipo) rimasti in gioco per ciascun giocatore.

Variante ICC: Su ICC questa richiesta non si può fare, ma questo bilancio si può tenere a mente perché l'arbitro nei suoi annunci distingue tra catture di pedoni e catture di pezzi.

Sono state descritte le varianti perché purtroppo non esiste un insieme di norme di gioco universalmente accettate che regoli con precisione il comportamento dell'arbitro, specie per quel che riguarda il fatto di annunciare sistematicamente la possibilità di catture di pedone. Quelle che abbiamo esposto vengono dette regole inglesi, o Eastern rules; le varianti costituiscono le regole americane, o Western rules. Esistono tuttavia almeno due varianti delle regole americane: le regole RAND (Western rules) e le regole "stile Cincinnati". Queste ultime sono state adottate per il Kriegspiel su Internet. Infatti, il sito dell'Internet Chess Club www.chessclub.com permette di giocare a Kriegspiel (variante Wild 16).

Le differenze regolamentari si ripercuotono su due aspetti complementari:

gli annunci dell'arbitro e le deduzioni che possono trarre i giocatori. Per apprezzarne le conseguenze si osservi il diagramma in figura 3.6.

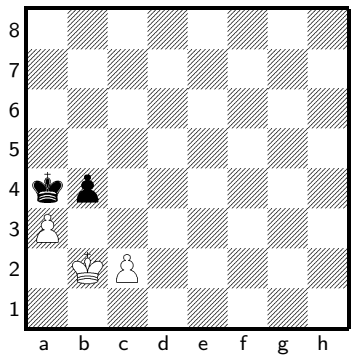


Figura 3.6: Diagramma d'esempio

Con le regole inglesi l'arbitro dice solo: "mossa al Bianco". Sta al Bianco chiedere o meno "Are there any?"; se il Bianco decide di porre questa domanda nella posizione di cui sopra, ottiene dall'arbitro una risposta affermativa, e deve provare almeno una cattura, ovvero ab4, cb3 o cd3.

Con le regole RAND prima della mossa del Bianco l'arbitro annuncia ad entrambi i giocatori "possibile cattura in b4". Il Bianco non ha l'obbligo di tentare una cattura.

Con le regole ICC prima della mossa del Bianco l'arbitro annuncia ad entrambi i giocatori "possibile una cattura". Il Bianco non ha l'obbligo di tentare una cattura.

Se adesso il Bianco invece di catturare muove il pedone in c4, con le regole inglesi l'arbitro dice solo: "mossa al Nero"; con le regole RAND l'arbitro dice: "possibili catture in a3 e c3"; con le regole ICC l'arbitro dice: "possibili due catture".

Anche se è ovvio, va rimarcato che questa differenza regolamentare non ha importanza nei finali senza pedoni.

Nel gioco su ICC la comunicazione sulle possibili catture di pedone è automatica e fatta ad entrambi i giocatori. Tuttavia nelle mosse di cattura viene detto se l'unità catturata è un pezzo o un pedone. Invece, diversamente dalle regole RAND, su ICC non viene annunciata la promozione di un pedone. Siccome le regole RAND stabiliscono che le promozioni vengono annunciate dall'arbitro, allora con queste regole in ogni momento i giocatori possono addirittura chiedere all'arbitro la ricapitolazione del conteggio dei pezzi e dei pedoni rimasti. Questo non è possibile su ICC: dunque si può solamente tenere da soli il conto dei pezzi in gioco separatamente da quello dei pedoni,

e sperare di evitare o almeno indovinare le mosse avversarie di promozione. D'altra parte, ci si può fare un'idea abbastanza precisa delle colonne su cui si trovano i pedoni avversari superstiti.

Non va dimenticato, infine, che su ICC vale la regola di patta se per 50 mosse non si effettuano mosse di pedone o di cattura.

In conclusione, lo spirito delle regole inglesi consiste nel far "pagare" qualunque informazione venga dall'arbitro, per esempio obbligando ad un tentativo di cattura dopo la richiesta "Are there any?". All'opposto, lo spirito delle regole RAND è di accelerare il gioco chiarendo ad ogni turno quali sono le mosse possibili e quanti sono i pezzi e i pedoni avversari rimasti in gioco. Lo spirito delle regole ICC (anche dette "stile Cincinnati") è in qualche modo intermedio tra i due precedenti.

Queste differenze regolamentari costituiscono un problema, perché la scarsa letteratura esistente fa riferimento quasi esclusivamente alle regole inglesi, mentre se si vuole avere un'esperienza di gioco la cosa più comoda è giocare su Internet con le regole "stile Cincinnati".

3.2.3 Considerazioni sul gioco del Kriegspiel

La natura del Kriegspiel è profondamente diversa da quella degli Scacchi [9]. Gli Scacchi ortodossi, come la Dama ed il Filetto, sono giochi ad informazione perfetta: in ogni momento gli avversari hanno le stesse informazioni; d'altra parte il Kriegspiel, la Battaglia Navale, il Poker, Stratego e Risiko sono giochi a informazione imperfetta. Nella Battaglia Navale non si conosce la disposizione dei vascelli avversari. Nel Poker i giocatori non conoscono le carte degli avversari. Stratego è un gioco di scacchiera in cui i giocatori conoscono la posizione delle pedine avversarie ma non il loro valore. Risiko ha in comune con il Kriegspiel il fatto di derivare dai wargame; l'incompletezza di informazione riguarda tuttavia solo l'obiettivo finale di ciascun giocatore.

La presenza di nodi aleatori (*chance node*) non condiziona l'appartenenza di un gioco ad una classe o all'altra, il Backgammon, ad esempio, è considerato un gioco ad informazione perfetta nonostante la presenza dell'alea, sull'altro versante invece il Kriegspiel presenta posizioni ad informazione imperfetta, ma prive di eventi casuali [24].

I giochi ad informazione perfetta, ovvero quelli in cui l'incompletezza dell'informazione può essere eliminata, si prestano ad un'analisi esaustiva con ragionamenti di tipo logico effettuati su tutte le possibili combinazioni di gioco [9].

Uno dei primi risultati della Teoria dei Giochi è formulato dal seguente teorema:

Teorema 7 (di Zermelo (1912)) *Nel gioco degli Scacchi vale una sola delle tre alternative:*
a) Il Bianco può forzare la vittoria.
b) Il Nero può forzare la vittoria.
c) Entrambi possono forzare almeno la patta.

von Neumann ha dimostrato che se vale il *Teorema di Zermelo* si può applicare il *Teorema Minimax*, che si basa sulla costruzione dell'albero di gioco.

Poiché non è possibile ragionare sull'intero albero, come discusso in [9], per poter giocare efficacemente a Scacchi è importante saper prevedere quali sono le strategie dell'avversario, deducendole dalle mosse che fa. Nella pratica agonistica degli Scacchi, tuttavia, l'attività di analisi deduttiva è di solito accompagnata da un'analisi di tipo extralogico, specie psicologica, per esempio guidata dalla conoscenza delle capacità e delle abitudini dell'avversario. È ben noto infatti che alcuni grandi campioni, come ed esempio Emmanuel Lasker o Michail Tal, in alcuni casi non giocavano la mossa logicamente migliore rispetto alla posizione sulla scacchiera, quanto piuttosto la mossa che a loro giudizio aveva maggiori probabilità di mettere in difficoltà l'avversario.

Il Kriegspiel richiede capacità di analisi logica e strategica in misura maggiore che gli Scacchi. Seguendo la definizione fornita in [18], se la tattica è lo sfruttamento combinativo di una debolezza e la tecnica è la capacità di vincere una partita considerata vantaggiosa, la strategia è il piano di gioco. Qualcuno, in modo divertente quanto efficace, ha sintetizzato la differenza tra strategia e tattica dicendo: "la tattica è sapere che cosa fare quando c'è qualcosa da fare e la strategia è sapere che cosa fare quando non c'è niente da fare". La strategia ha per scopo la formazione di debolezze tali da poter essere sfruttate con colpi tattici o in sede tecnica. La sua formulazione deve naturalmente tener conto di tutti gli elementi presenti sulla scacchiera che però, beninteso, varia al variare della posizione. Non è quindi possibile formulare un piano unico sempre valido, ma è anche vero che, data una posizione, talvolta possono formularsi più piani. La scelta del piano dipende allora dall'indole del giocatore, ma non si possono formulare piani che non tengano conto degli elementi strategici presenti nella posizione.

Leoncini dice che un errore comune è credere che la differenza tra un maestro di Scacchi e un principiante risieda nella capacità di calcolo; certo, un bravo giocatore è capace di calcolare con precisione anche lunghe varianti, ma la superiorità è soprattutto di ordine strategico. Il maestro sa che cosa fare in qualsiasi posizione senza bisogno di calcoli approfonditi. La formulazione di un piano riduce drasticamente la necessità del calcolo delle varianti; in questo senso nel corso della partita il maestro può calcolare meno di un principiante

ma vincere lo stesso. È per questo che i forti giocatori possono giocare contro molti avversari contemporaneamente in simultanea e batterli.

Quanto detto ci porta a concludere che, molto più degli Scacchi, il Kriegspiel al 99% si basa sulla strategia.

Come è messo in risalto in [9] il Kriegspiel ha alcuni punti di contatto anche con il Poker. Ad esempio è possibile in qualche misura bluffare. In senso proprio, un bluff nel Poker consiste nell'indurre l'avversario a credere che possediamo carte migliori di quelle che realmente abbiamo. Nel Kriegspiel il bluff avviene inducendo l'avversario a credere che la nostra posizione sia diversa da quella che è. Una differenza importante tra Poker e Kriegspiel è la seguente: mentre nel Poker anche se si conoscono le carte di tutti i giocatori non è possibile predire con precisione quali verranno distribuite dal tallone quando i giocatori ne cambieranno alcune, nel Kriegspiel la somma delle diverse conoscenze dei giocatori è uguale allo stato reale del gioco. In effetti nel Kriegspiel l'arbitro conosce perfettamente tale stato, ed è quindi possibile ipotizzare l'applicazione del Teorema Minimax ad esso, considerando la scacchiera senza incertezza; quindi è possibile delineare la strategia ottimale per entrambi i giocatori in una data posizione e, di conseguenza, giudicare la qualità delle mosse dei due avversari.

3.2.4 La Teoria dei Giochi e gli scacchi invisibili

Lo statunitense Herbert Simon, premio Nobel per l'Economia nel 1978 per le sue ricerche sui processi decisionali all'interno di organizzazioni economiche, fu uno dei primi studiosi a sottolineare l'importanza del gioco degli scacchi per modellare situazioni di complessità decisionale ([9]).

Simon era interessato allo studio del comportamento razionale; egli introdusse la distinzione tra *razionalità sostanziale* e *razionalità procedurale*. Un comportamento si dice *sostanzialmente* razionale quando cerca la *migliore* azione da intraprendere per risolvere un problema. Un comportamento si dice invece *proceduralmente* razionale, quando per risolvere un problema cerca un'azione *accettabile* tenendo conto non solo dell'obiettivo e dei vincoli oggettivi, ma anche delle conoscenze e delle capacità computazionali dell'agente decisore. Questa distinzione è utile per analizzare le decisioni di agenti posti in situazioni decisionali complesse, in cui non è possibile valutare le ultime conseguenze di una decisione. Quando un problema posto ad un agente decisore mette in evidenza i suoi limiti computazionali, l'approccio procedurale permette comunque di affrontare il problema razionalmente.

Secondo Simon, il gioco degli Scacchi rappresenta un campo d'indagine abbastanza interessante per mettere in discussione i canoni della razionalità sostanziale. Simon suggerisce un approccio detto a *razionalità limitata*, che

cerca di prescrivere come l'agente possa decidere cosa fare. Ancora più interessante degli Scacchi si presenta il Kriegspiel, che, certamente, sfugge ad ogni trattamento di razionalità sostanziale ([9]).

Nel Kriegspiel si possono avere meccanismi adatti ad essere trattati nella Teoria dei Giochi formulata da von Neumann ed estesa al lavoro di John C. Harsanyi del 1967-68 [36].

È chiaro che la relazione che passa fra una strategia e un'altra se la prima batte la seconda non è transitiva, può darsi cioè che la strategia A batta la B che batte la C che batte la D e che a sua volta batte la A.

Propongo di seguito una breve apertura di partita tratta da [23]:

1.e4 (l'arbitro dice "il Bianco ha mosso"), ... d5 (l'arbitro dice "Il nero ha mosso; il Bianco ha una presa di pedone in d5")

2.ed5 (l'arbitro dice "il bianco ha mosso, prendendo un pedone in d5"), ... ♗g4 (il nero specula sul fatto che è improbabile che il Bianco indovini l'attacco alla sua donna; L'arbitro dice "Il Nero ha mosso")

3.f3 (il Bianco ipotizza come le cose possano essere andate; l'arbitro dice: "il Bianco ha mosso"), ... ♗d1 (l'arbitro dice "impossibile" e il nero pensa che, per esempio, può esserci un cavallo bianco in e2 o in f3, rinuncia al suo tentativo e gioca ... ♘c6. L'arbitro dice: "Il Nero ha mosso; il Bianco ha una presa di pedone in g4".)

4.fg4 (l'arbitro dice: "il Bianco ha mosso e ha preso un pezzo in g4".)

Lo studio dei finali è, per usare le parole di Magari, affascinante, con i pezzi avversari che oscillano fra la condizione di corpuscoli e quella di onde di probabilità ([23]).

Posizioni vinte a scacchi possono essere perse o patte a scacchi invisibili e viceversa. Occorre inoltre notare che a scacchi invisibili una posizione può essere, anziché semplicemente persa, patta o vinta anche, per esempio, vinta con una certa probabilità. Ciò spiega il motivo per cui è possibile usare la Teoria dei Giochi di von Neumann.

La posizione in figura 3.7, che per semplicità assumiamo essere nota ad ambedue i giocatori, è ovviamente patta, chiunque abbia il tratto, a scacchi ortodossi ed è invece, a scacchi invisibili, "quasi" vinta per il Bianco, nel senso che il Bianco ha la facoltà di scegliere la probabilità con cui vincere purché non esiga la certezza. Più formalmente, preso un numero reale $k < 1$ esiste una strategia per il Bianco che dà probabilità $p > k$ di vincere. Ciò è vero se si prescinde da regole quali quella delle "cinquanta mosse", mentre se è posto un limite al numero di mosse senza catture allora si può calcolare la miglior probabilità per il Bianco.

Supponiamo il tratto al Nero: egli ha la scelta fra i tratti 1. ... ♔b8,

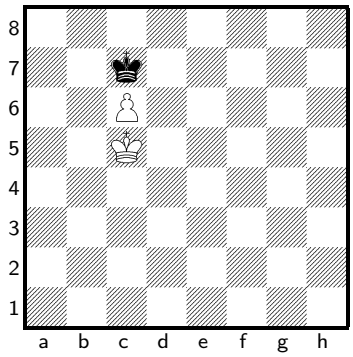


Figura 3.7: Magari 1992

1. ...♔c8, 1. ...♔d8. A scacchi ortodossi dovrebbe senz'altro giocare 1. ...♔c8. Il Bianco giocherà 2. ♔b6 o 2. ♔d6, supponiamo ♔b6. Se il Nero, al tratto precedente, si è posto in b8 o in d8 ora giocherà 2 ...♔c8, altrimenti 2 ...♔b8 o 2 ...♔d8. Il Bianco naturalmente non sa a questo punto se il Re nero è in b8, in c8 o in d8. È chiaro che se il Re nero fosse in c8 al Bianco converrebbe spingere il pedone, ma se il Re nero fosse invece in una casa laterale (b8 o d8) questa mossa porterebbe alla patta. In ogni caso al Bianco (supposto in b6) conviene anzitutto tentare 3. ♔b7. Se questa mossa è possibile il Bianco avrà vinto, altrimenti può ancora scegliere se spingere il pedone o tornarsene, diciamo, in c5 per cominciare da capo la manovra. Se non c'è limite alla lunghezza della partita al Bianco conviene ripetere la manovra senza spingere il pedone se non, diciamo, la k-esima volta che tenta. Il Nero deve giocare al secondo tratto della manovra ...♔c8, altrimenti c'è una probabilità su due che il Bianco si sia posto dal lato giusto e vinca. Così facendo però, quella volta in cui il Bianco spingerà, il Nero si troverà in c8 e perderà. Perciò al Nero conviene giocare qualche volta al secondo tratto ...♔b8 o ...♔d8 e, se lo farà per l'appunto quella volta in cui il bianco spingerà, la partita sarà patta.

La probabilità di vittoria del Bianco è tanto più grande quanto maggiore è k, e tende a 1 al tendere di k all'infinito. Nel caso in cui ci siano limiti alla lunghezza della partita e si sia giunti all'ultima possibile ripetizione della manovra, il Bianco dovrà giocare, dopo il tentativo di avanzare il Re in settima, la spinta, vincendo se il Re nero si troverà in c8 e pattando invece se si troverà, in b8 o in d8. Se, al secondo tratto, il Nero gioca ...♔c8, il Bianco, dopo aver tentato inutilmente l'avanzata in settima, giocherà certamente la spinta, vincendo, quindi il Nero deve in queste circostanze giocare al secondo tratto ...♔b8 o ...♔d8, pattando se il Bianco si sarà portato sulla stessa colonna e perdendo altrimenti. In questo semplice caso quindi la probabilità

di vittoria del Bianco è 1/2.

3.2.5 Gioco bayesiano

Consideriamo il caso di un gioco a 2 giocatori [30] in cui al giocatore I corrisponde un insieme T_1 di tipi[6] che contiene due elementi: $T_1 = \{I.1, I.2\}$ e al giocatore II corrisponde l'insieme $T_2 = \{II.1, II.2\}$. Un elemento di T_1 o T_2 rappresenta tutta l'informazione necessaria al giocatore per le proprie scelte e che non è conoscenza comune tra tutti i giocatori.

Definiamo due insiemi C_1 e C_2 che contengono le azioni[7] a disposizione dei due giocatori rispettivamente, $C_1 = \{x_1, x_2\}$ e $C_2 = \{y_1, y_2\}$.

Definiamo le funzioni di utilità come $u : (C_1 \times C_2) \times T_1 \longrightarrow \mathbb{R}$ per il primo giocatore, e $v : (C_1 \times C_2) \times T_2 \longrightarrow \mathbb{R}$ per il secondo.

Infine definiamo le distribuzioni di probabilità soggettive di ciascun giocatore nella funzione $p^I : T_1 \longrightarrow \Delta(T_2)$, ossia p^I associa ad ogni tipo in T_1 una distribuzione di probabilità su T_2. Si definisce quindi l'analoga funzione p^{II} per il giocatore II.

Un gioco bayesiano (a 2 giocatori) è quindi:

$$\Gamma^b = ((C_1, C_2), (T_1, T_2), (p^I, p^{II}), (u, v)) \tag{3.1}$$

ove
$C = C_1 \times C_2$ è l'insieme delle azioni possibili dei due giocatori
$T = T_1 \times T_2$ è l'insieme dei tipi possibili dei due giocatori
$u, v : C \times T \longrightarrow R$ sono le due funzioni di utilità
$p^I : T_1 \longrightarrow \Delta(T_2)$
$p^{II} : T_2 \longrightarrow \Delta(T_1)$

Si assume che Γ^b sia conoscenza comune dei giocatori e che ogni giocatore sappia qual'è il suo vero tipo; inoltre il fatto che ogni giocatore conosca il suo tipo è conoscenza comune tra i giocatori.

È bene precisare cosa intendiamo per "azione" e "strategia". Una azione per il giocatore I è un elemento di C_1, mentre una strategia è una funzione che indica per ogni possibile tipo del giocatore I quale sia l'azione che dovrebbe scegliere, ovvero $s_1 : T_1 \longrightarrow C_1$. L'equivalente del concetto di strategia mista nel contesto dei giochi bayesiani può essere quindi, per il giocatore I $s_1 : C_1 \times T_1 \longrightarrow [0, 1]$, ove $\sum_{c \in C_1} s_1(c, t) = 1 \ \forall t \in T_1$.

Per ogni possibile tipo t per il giocatore I si individua così una distribuzione di probabilità su C_1, lo stesso è fatto per il giocatore II.

[6] qui il concetto formale di tipo non è lontano dal significato della parola "tipo" nel linguaggio comune

[7] l'idea di azione è molto vicina quella di strategia (pura) per un gioco in forma normale

Per poter calcolare il payoff atteso e conseguentemente parlare di equilibri, è necessario specificare le distribuzioni di probabilità assegnate a ciascun giocatore, poiché se un giocatore conosce, come è ovvio, il suo tipo gli altri ne sono ignari. In altri termini se il giocatore I è cosciente del suo tipo, diciamo I.1, questo fatto non è conoscenza comune, pertanto egli rischierebbe di scegliere delle azioni errate se non tenesse conto di questo aspetto.

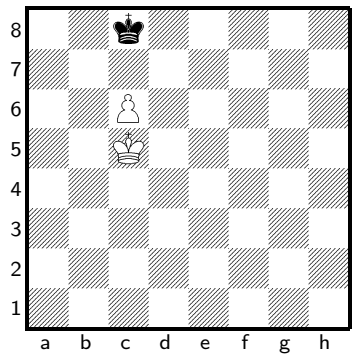


Figura 3.8: tipo I.1 bianco, II.1 nero

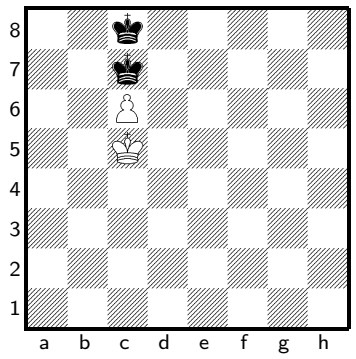


Figura 3.9: esempio di incertezza

Affrontiamo ora il seguente esempio. Sia il tipo I.1 quello che vede posizionati il re bianco in c5 e il pedone bianco in c6 e sia il tipo II.1 quello con il re nero in c8, come mostrato in figura 3.8. Consideriamo che il tratto sia del bianco e che il bianco debba andare in promozione in non più di 4 mosse[8], altrimenti la partita finirebbe per patta. Si ha cioè che gli insiemi dei tipi di entrambi i giocatori sono ridotti ad un singoletto, ossia $T_B = \{I.1\}$ e $T_N = \{II.1\}$.

I.1\ II.1	...♔b8	...♔d8
♔b6	0	1
♔d6	1	0

Tabella 3.1: Tabella dei pagamenti relativi ai tipi I.1 e II.1

Indicando con 1 la vittoria del bianco, con 0 la patta, in tabella 3.1 è proposta la tavola dei pagamenti. Da questa si evince che la strategia mista opportuna assegna la probabilità di 1/2 a ciascuna azione rendendo quindi

[8]ad esempio se si è giunti al 48^o tratto durante il conteggio della regola delle 50 mosse

la probabilità di vittoria per il Bianco pari a 1/2, come era avvenuto nella sezione 4.3.

Se ora introduciamo un secondo tipo per il Nero, indicato con II.2, che vede il re nero in c7, come mostrato in figura 3.7, otteniamo l'insieme dei tipi $T_N = \{II.1, II.2\}$. Nella tabella 3.2 è mostrata la tavola dei pagamenti nel caso il Nero sia del tipo II.2: la strategia mista assegna la probabilità di 1/3 a ciascuna delle tre strategie e il valore del gioco per il bianco è 1/3.

I.1\ II.2	♔c8-♔b8	♔c8-♔d8	♔d8(♔b8)-♔c8
♔b4-♔b6-♔b7	0	1	0
♔d4-♔d6-♔d7	1	0	0
♔d4(♔b4)-♔d6(♔b6)-c7	0	0	1

Tabella 3.2: Tabella dei pagamenti relativi ai tipi I.1 e II.2

La miniatura in figura 3.9 mostra la posizione formalizzata considerando il giocatore bianco del tipo I.1, ovvero T_B, e il giocatore nero del tipo $T_N = \{II.1, II.2\}$. In questo caso la probabilità che il giocatore II (il Nero) sia del tipo II.1 oppure del tipo II.2 è di 1/2 in ciascun caso.

In questo semplice esempio la strategia migliore per il Bianco è quella di giocare inizialmente le mosse ♔b6 o ♔d6 in modo equiprobabile e scoprire con probabilità 1 a quale tipo appartenga il Nero. Se infatti riceve dall'arbitro la risposta "illegale" il Nero sarà del tipo II.2, altrimenti sarà del tipo II.1. L'esempio mostra comunque come tale approccio consista nel considerare giochi ad informazione incompleta come caso particolare di giochi ad informazione completa e sarà approfondito nella sezione 3.2.6.

3.2.6 Giochi ad informazione completa ma imperfetta

Tra i giochi ad informazione completa e quelli ad informazione incompleta la differenza sostanziale può essere superata formulando la "natura" che seleziona i tipi dei giocatori, e quindi delle coppie di tipi, con una data probabilità. Questa interpretazione considera quindi giochi ad informazione incompleta come caso particolare di giochi ad informazione completa, ma imperfetta.

Il gioco del Kriegspiel è considerato ad informazione imperfetta, in quanto nell'albero della forma estesa si ha la presenza di insiemi informativi con più di un elemento, ed entrambi i giocatori sono in grado di descrivere l'albero di gioco.

D'altra parte, considerando il caso specifico di un finale, la posizione dei pezzi avversari è incognita. Nel finale di torre ad esempio, indicheremo

l'incertezza sul re nero disegnando nella scacchiera del Bianco numerosi re neri laddove si ha una probabilità positiva che vi sia effettivamente il re avversario. In questa particolare situazione il gioco può essere considerato ad informazione incompleta, in quanto il giocatore non ha gli strumenti per poter costruire l'albero delle mosse avversarie.

L'analisi nella sezione precedente e l'esempio sul finale di pedone, torna utile in questo caso. Anche nel caso dei finali, in cui il gioco prende inizio da una posizione incerta, a differenza del gioco generale in cui si incomincia dalla posizione iniziale classica degli scacchi, è possibile formulare la "natura" che posiziona il re avversario formulando una distribuzione di probabilità sulle posizioni dei pezzi avversari.

Quanto detto porta alla conclusione che il Kriegspiel può essere considerato ad informazione completa, ma imperfetta; adatto quindi a uno studio e a una soluzione tramite ricerca sull'albero di gioco.

3.2.7 Situazioni di gioco senza componente stocastico

Si consideri la posizione del finale di re e torre contro re in Kriegspiel, descritta in figura 3.10. Sia il tratto al Bianco.

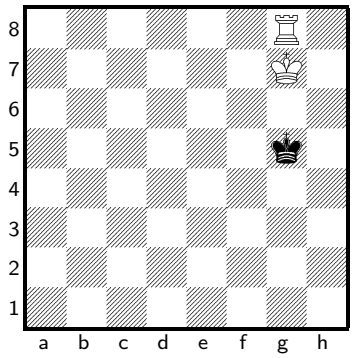


Figura 3.10: esempio

L'albero di gioco, che ha per radice la posizione in figura, ha una prima diramazione pari alle 11 mosse possibili del Bianco. Per ogni mossa, il re nero ha, in 2 casi, a disposizione 3 scelte, in altri 2 casi 6 scelte, e, infine, in 7 casi 5 scelte. Le scelte del nero in questo caso compongono gli insiemi informativi per il Bianco, che non conosce la posizione ove il Nero ha mosso.

Si hanno 32 insiemi informativi per il Bianco, per un totale di 53 nodi; continuando, la diramazione dell'albero ha una "esplosione" numerica consistente, molto difficile da trattare senza raffinamenti.

Se si ha intenzione di riscrivere il gioco in forma strategica, è bene ricordare che essa è esponenziale sul numero degli insiemi informativi, pertanto, solo alla seconda profondità nell'albero, si avrebbero $32^{\overline{x}}$ righe, dove con $\overline{x}$ considero la media del numero di mosse possibili per il Bianco. Se la forma estesa aveva una natura matematica di difficile gestione, la forma strategica diviene impossibile da trattare. Si consideri l'albero in figura 3.11, ottenuto giocando dalla miniatura in figura 3.10 la mossa ♖f8; le mosse possibili del Nero sono in questo caso 3, ♚h5, ♚h4 oppure ♚g4. Il bianco è quindi di fronte a due insieme informativi, di 2 elementi e di 1 rispettivamente. Il numero di mosse possibili diviene pari a 21 (7 per il re e 14 per la torre, per il primo insieme informativo), diviene pari a 19 per il secondo. L'albero si dirama quindi per $21 + 21 + 19 = 61$ nodi ulteriori.

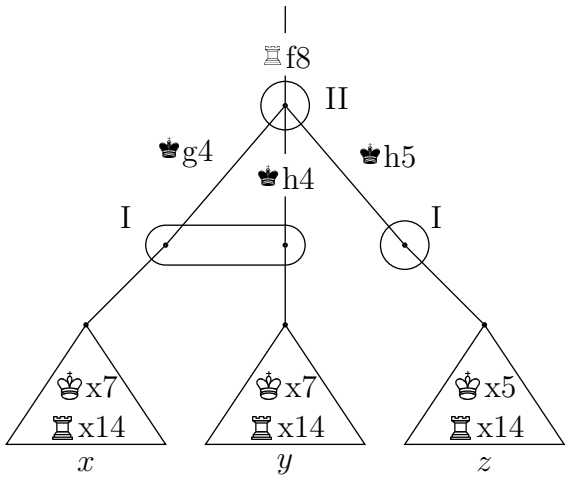


Figura 3.11: Il giocatore I ha un insieme informativo con più di un elemento

Si potrebbe quindi pensare di trasformare il gioco in forma sequenziale, come si è fatto per il gioco del Poker descritto da Kuhn; anche in questo caso, però, persiste la pesante mole numerica.

In figura 3.11, le variabili x, y e z indicano la valutazione dei sottorami, che può essere eseguita, ad esempio, da una funzione di valutazione. Se è presente un insieme con più di un elemento, in generale, non è possibile applicare un algoritmo di ricerca, come quello minimax, per trovare una soluzione ottima o approssimata ad una certa profondità di ricerca. Se il Nero avesse una priorità sulle sue scelte che portano ad uno stesso insieme informativo del Bianco, allora si potrebbe ipotizzare una visita dell'albero per dedurre quale tra le scelte avversarie è più pericolosa; un insieme informativo con più di un elemento non preclude comunque l'eventualità che vi siano

mosse equiprobabili, quindi rende generalmente impraticabile la via della ricerca sull'albero.

Una possibile soluzione a questo inconveniente consiste nel lavorare sugli insiemi informativi del Bianco accorpandoli in un'unica posizione, chiamata *metaposizione*, descritta in [34]. Riunire in uno stato quegli stati ottenuti da mosse differenti, ma equiprobabili, permette di rappresentarle contemporaneamente, senza il vincolo di doverne scegliere una in particolare. Chiaramente, mosse con priorità diverse, possono essere rappresentate senza perdita di informazione tramite una metaposizione.

Si consideri, per il momento, la possibilità di riunire in un'unica mossa quelle mosse del Nero che portano ad uno stesso insieme informativo, e ottenere così metaposizioni in cui si ha incertezza sulla posizione dei re avversari. L'esempio in figura 3.11 viene semplificato come in figura 3.12. Le mosse ♚g4 e ♚h4, che portano allo stesso insieme informativo, portano ora ad una metaposizione unica. Ricordando la definizione (11), stiamo trasformando questa posizione di finale del Kriegspiel da gioco ad informazione imperfetta a gioco ad informazione perfetta.

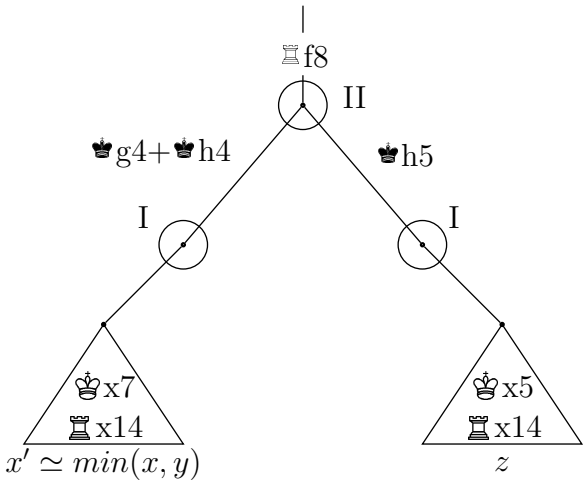


Figura 3.12: Il giocatore I ha due insiemi informativi con un solo elemento

Nel caso dell'esempio, l'accorpamento delle mosse porta a sottoalberi di gioco la cui valutazione sarà data dal minimo valore ottenuto dai sottorami originali che si avevano prima dell'unione. Quindi, sebbene sia stato riformulato, il problema resta nella pratica difficile. In effetti, dire che $x' \simeq min(x, y)$, ove con x' si indica la nuova valutazione della metaposizione, comporta sia il calcolo di x che quello di y. Ne segue che una ricerca in profondità non trae alcun giovamento dalla trasformazione effettuata.

Un miglioramento nelle prestazioni può essere ottenuto adottando una funzione di valutazione statica, oltre a quella ricorsiva, in grado di dare un giudizio durante la ricerca. Si può scegliere una funzione che valuti posizioni classiche, ma poiché stiamo introducendo le metaposizioni, è bene considerare funzioni giudicatrici di metaposizioni.

Il concetto di metaposizione non può più essere scartato, in quanto è l'unico in grado di permetterci la gestione di più posizioni equivalenti alla stessa profondità.

In fase di progettazione la procedura di distinzione tra le mosse del Nero, che portano a differenti insiemi informativi del Bianco, diviene alquanto complessa. Per conoscere i diversi insiemi informativi, bisogna esaminare le possibili mosse del bianco e quindi procedere a ritroso di un passo per effettuare la divisione. Questo si traduce in uno sforzo computazionale rilevante.

Si è quindi scelto di dare all'albero di gioco una forma più congeniale, ma equivalente, riformulando l'incertezza dei giocatori. La metamossa del Nero non comprende più solo quelle mosse che portano ad uno stesso insieme informativo, bensì comprende tutte le mosse possibili che egli può effettuare a partire dalla metaposizione in cui si trova. Si sta, per così dire, trasformando un ipotetico finale del gioco del Kriegspiel in uno equivalente, in cui si devono fronteggiare più re neri allo stesso tempo.

Così facendo si perde però l'idea di insieme informativo come insieme di nodi da cui si dipartono le medesime mosse. Per far fronte a questa mancanza si introducono le metamosse o **mosse pseudolegali**, mosse, cioè, della cui legalità il giocatore non è al corrente.

Le mosse del Nero accorpate nell'esempio precedente, sono quindi ♚h5, ♚h4 oppure ♚g4. L'albero di gioco diviene quello mostrato in figura 3.13.

La cardinalità delle mosse pseudolegali è da considerarsi pari al massimo numero di mosse legali per ciascuna posizione, ossia $|mossepseudolegali| = max(|mosselegali|)$. Nell'esempio quindi si considerano sempre 21 mosse effettive, anche se si è nel caso con 19 mosse legali, non potendo distinguere tra le tre diverse situazioni in cui ci si trova.

Perché il gioco sia equivalente al precedente è necessario introdurre nell'albero le informazioni date dall'arbitro. Siano esse, ad esempio "scacco" (C), "illegale" (I), oppure nessuna risposta (S), allora dopo ogni mossa pseudolegale si attende una delle tre risposte dell'arbitro. L'albero ha, quindi, una diramazione pari a $3 \cdot i$ ove con i indico il numero di metaposizioni. Nell'esempio in figura 3.10, si hanno 32 insiemi informativi, ma solo 11 metaposizioni, l'albero ha quindi una diramazione pari a $3 \cdot 11 = 33$ nodi.

In figura ho evidenziato le metaposizioni con un doppio cerchio, per

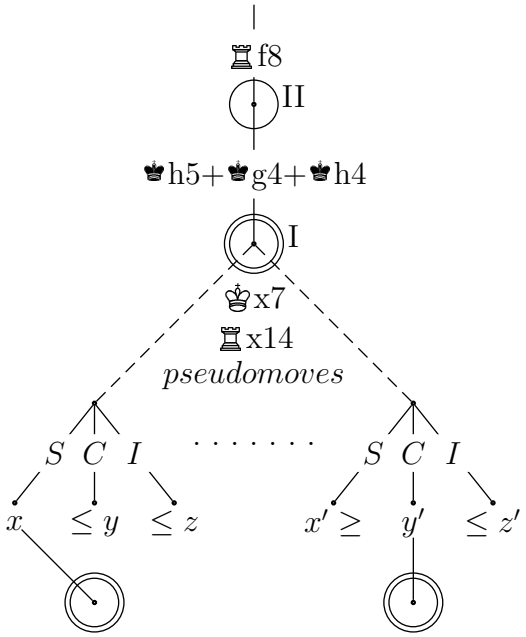


Figura 3.13: Si ha un'unica metaposizione

differenziarle dal formalismo usato per gli insiemi informativi; ho inoltre raffigurato le mosse pseudolegali tramite un arco tratteggiato.

È possibile utilizzare un'euristica per accorpare nell'albero di gioco anche le risposte dell'arbitro, scegliendo ogni volta la peggiore tra le tre possibili. Come detto, per poter dare un giudizio al valore di ogni metaposizione verrà scritta una funzione di valutazione statica, che permetterà di unificare le risposte ricevute dopo ogni pseudomossa. È interessante osservare il comportamento dell'algoritmo a seconda delle diverse scelte possibili nel caso in cui le valutazioni delle risposte siano equivalenti, ossia $x = y = z$. A tal proposito rimando al Capitolo successivo, in cui verranno approfonditi questi aspetti del giocatore artificiale e la loro implementazione.

3.2.8 Situazioni di gioco con componente stocastico

Quanto detto è sicuramente corretto per quelle posizioni del gioco in cui non si ha a che fare con mosse aleatorie, ma nel gioco del Kriegspiel, come si è visto, sono numerosi i casi in cui la soluzione è data da strategie probabilistiche.

Di seguito mostrerò una possibile soluzione nel caso si debbano gestire mosse aleatorie. Si consideri l'esempio in figura 3.7 con il tratto al Nero, che è stato risolto tramite una strategia mista probabilistica nella sezione 4.3.

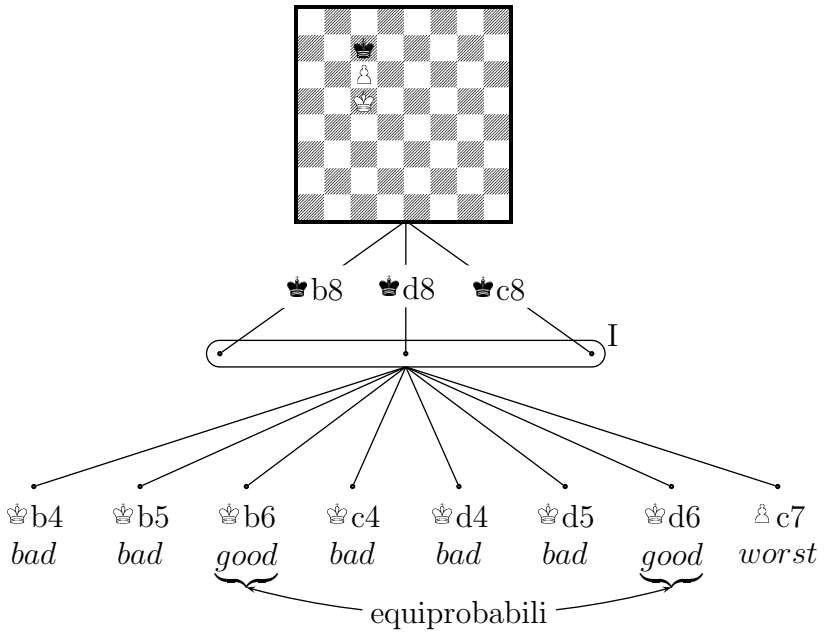


Figura 3.14: Due mosse sono equiprobabili

L'accorpamento delle mosse del Nero, che portano ad un'unica *metaposizione*, conducono in questo particolare caso allo stesso insieme informativo, pertanto le pseudomosse del Bianco uscenti possono essere considerate mosse legali a tutti gli effetti.

Si supponga di avere a disposizione una buona funzione di valutazione che ci permetta di classificare le scelte per il Bianco. Ad esempio, essa potrebbe essere basata sui tre fattori seguenti in ordine di importanza:

1. Il Pedone non deve rischiare di essere mangiato;
2. Deve essere privilegiato l'avanzamento del Pedone;
3. L'avanzamento in settima del Pedone può essere eseguito solo se il Re è in settima;
4. Il Re deve essere adiacente al Pedone.

La mossa ♙c7 sarebbe classificata come la peggiore e quindi scartata; essendo il re nero in b8, in c8 o in d8, il Pedone rischierebbe di essere mangiato e il gioco finirebbe per patta. Le mosse ♔b4, ♔c4, ♔d4 avrebbero un voto basso, in quanto allontanano il Re dal Pedone; le mosse ♔b5 e ♔d5 avrebbero un voto migliore, ma non alto, poiché non aiutano la spinta del pedone, che è l'obiettivo da perseguire. Infine le due mosse ♔b6 e ♔d6 avrebbero voto maggiore (in quanto non si allontanano dal Pedone e ne aiutano l'avanza-

mento), ma equivalente. L'equivalenza numerica della valutazione fatta per queste due mosse è inevitabile.

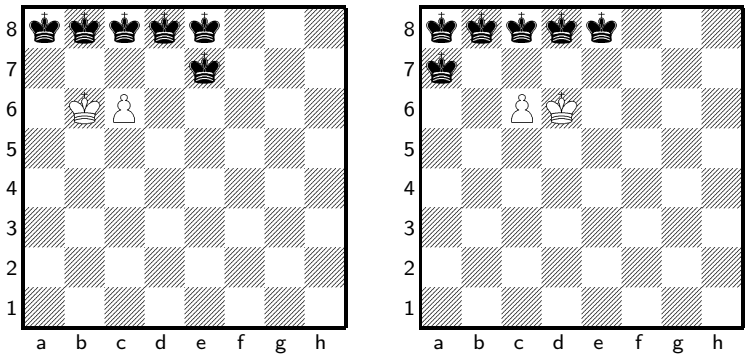


Figura 3.15: Metaposizioni ottenute con ♔b6 e ♔d6

In figura 3.15 si osserva la simmetria delle due metaposizioni ottenute con le due mosse ♔b6 e ♔d6. Tale simmetria si traduce nel fatto che non esiste alcun valore numerico che vada bene per una metaposizione e non per la simmetrica; in altri termini nell'albero di ricerca si hanno due nodi con lo stesso identico voto statico. L'algoritmo può in questo caso utilizzare un generatore di numeri casuali e attribuire un voto maggiore ad uno dei due stati, con probabilità pari a 1/2. Possiamo quindi parlare di *metaposizioni aleatorie*.

Questo è un punto molto importante. È bene notare che, in presenza di metaposizioni aleatorie, ogni visita dell'albero di gioco diviene essa stessa aleatoria.

Una conseguenza a questo fatto è che non sarà più possibile utilizzare accorgimenti per velocizzare il calcolo delle valutazioni, come strutture dati quali le tabelle hash di trasposizione o le chiavi di Zobrist, in quanto ad ogni visita non si considererebbe l'incertezza aleatoria, che è invece necessaria.

Nel nostro esempio, consideriamo, senza perdere di generalità, che venga scelta la mossa ♔b6, ottenendo la metaposizione di sinistra in figura 3.15. Le pseudomosse per il Bianco ♔a7, ♔a6 e ♔a5 verrebbero scartate, in quanto il Re si allontanerebbe dal Pedone; le mosse ♔b5 e ♔c5 verrebbero scartate, poiché non di aiuto all'avanzamento del Pedone; la mossa c7 sarebbe scartata non essendo il re in settima; infine le mosse ♔b7 e ♔c7 sarebbero considerate entrambe buone.

Supponiamo quindi che si giochi ♔c7 senza risposta da parte dell'arbitro: l'obiettivo sarebbe presto raggiunto giocando ♔b7 al turno successivo e, in caso di mossa illegale, ♔d7 e quindi l'avanzamento del pedone. Se la mossa ♔c7 riceve risposta di mossa illegale, l'algoritmo proverebbe la restante ♔b7,

a cui segue il Pedone in settima nel caso di risposta silente dell'arbitro. Se anch'essa risulta illegale restano le mosse ♔b5 e ♔c5. Per la nostra funzione di valutazione esse risultano equivalenti, si fa quindi uso del generatore di numeri casuali per tirare a sorte su quale sia la migliore. Se viene estratta ♔b5 non si hanno ulteriori "estrazioni" e il Bianco muove al turno seguente in ♔b6; se viene scelta ♔c5 si ripropongono le mosse equivalenti ♔b6 e ♔d6 e tutto ricomincia. In questo caso si può calcolare che la probabilità di passare al lato simmetrico, ossia a giocare la mossa ♔d6, è pari a 1/4, mentre la probabilità di restare nello stesso lato della scacchiera e tentare nuovamente la mossa ♔b6 è pari a 3/4. Ricordando la soluzione data da Magari, questo procedimento non è ottimo.

Si può quindi migliorare la funzione di valutazione aggiungendo un ulteriore punto:

5. Tra le mosse che portano il Re ad una riga inferiore a quella del Pedone, prediligi quelle che portano il Re nella medesima colonna del Pedone.

In questo modo, tra le mosse ♔b5 e ♔c5, verrebbe scelta la seconda, essendo il pedone in c6. Quindi con probabilità pari a 1/2 si cambierebbe lato.

Si noti come non sia stato complesso rappresentare la base di conoscenza per risolvere questo finale in una sequenza di punti. Formalizzare la funzione di valutazione vista in una struttura matematica è possibile ed è quindi realistico implementarla concretamente.

3.2.9 Il fattore di Branching nel Kriegspiel

Alla luce di quanto visto nelle sezioni precedenti trovo utile riportare le considerazioni esposte in [9], circa l'analisi del problema di costruire un programma in grado di giocare a Kriegspiel e la valutazione della *complessità computazionale* del Kriegspiel, di cui una misura tradizionale è data dal *fattore di diramazione* (*branching*) dell'albero di gioco. Nell'analisi computazionale dei giochi il fattore di diramazione è importante perché definisce proporzionalmente il carico computazionale necessario per esplorare l'albero di gioco: maggiore il fattore di diramazione medio, più tempo impiega un computer in media per analizzare le conseguenze di una mossa. Come è noto, nel caso degli Scacchi il fattore di diramazione medio è dell'ordine di 30-35 mosse legali per posizione. Quindi un programma che gioca per forza bruta dovrebbe esplorare circa mille varianti al primo livello, un milione al secondo livello, un miliardo al terzo, e così via. Nel caso del Kriegspiel questo fattore

è maggiore, in quanto vanno considerate le mosse pseudolegali. Ipotizzando un fattore doppio, arriviamo a 60-70 tentativi per posizione.

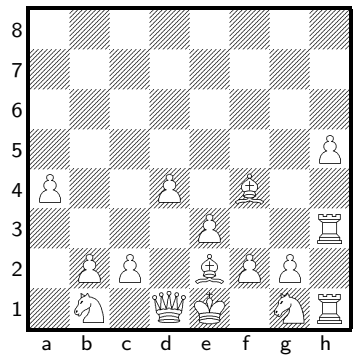


Figura 3.16: Diagramma d'esempio

Ad esempio, nella posizione mostrata in figura 3.16 il Bianco ha 38 tentativi pseudolegali, di cui 10 tentativi di cattura di pedone. Se contiamo le mosse possibili in questo modo però trascuriamo il fatto che una mossa in Kriegspiel consiste di una sequenza di tentativi. Sequenze diverse causano risposte diverse dell'arbitro, quindi portano ad avere informazioni diverse sullo stato della scacchiera. Occorre dunque contare tutte le possibili sequenze di semimosse. L'albero delle sequenze possibili in ciascuna posizione è molto grande: se le mosse pseudolegali in una certa posizione sono N, allora le sequenze possibili sono $N!$, ovvero $N \cdot (N-1) \cdot (N-2) \cdot ... \cdot 2$. Quindi nella posizione precedente il Bianco avrebbe in teoria 38! mosse possibili. In realtà se valgono le regole americane l'arbitro avrà comunicato se sono possibili catture di pedone: in caso negativo le sequenze possibili sono 28!.

Non considerando sequenze di mosse, ma mosse legali, per alcuni autori [7] il conto dei tentativi va moltiplicato per il numero di posizioni alternative in cui possono trovarsi i pezzi avversari compatibilmente con lo sviluppo della partita. Nel caso banale in cui l'avversario non abbia pezzi invisibili, l'albero di gioco per il Kriegspiel è grande tanto quanto quello per il gioco classico degli scacchi. Se sono introdotti pezzi "invisibili" [6], ogni nodo del gioco deve essere espanso per ogni mossa possibile per ogni possibile combinazione di posizioni del pezzo incognito dell'avversario. Supponendo di avere m pezzi nascosti, se ogni pezzo invisibile ip_i ha una probabilità positiva di occupare n_i case, allora il fattore di branching è approssimativamente il fattore di branching degli Scacchi moltiplicato per la combinazione delle case del pezzo invisibile che possono essere occupate. Questo viene descritto con la seguente formula:

$$Branching\ Factor = 35 \times n_1 \times n_2 \times n_m$$

Se ogni giocatore ha due pezzi nascosti e ciascun pezzo ha una media di solo quattro case occupate con probabilità positiva, allora il fattore di branching medio approssimato del gioco degli scacchi invisibili è pari a $35 \times 16 = 560$. Considerando tre pezzi "invisibili", si ottiene un branching factor pari a $35 \times 64 = 2240$. Assumendo che tali pezzi siano presenti sulla scacchiera da circa 50 mosse, allora l'albero di gioco diviene dell'ordine di 2240^{50} nodi, ovvero di circa 10^{150} nodi.

In Kriegspiel il concetto di insieme informativo corrisponde a quello dell'insieme delle conoscenze che un giocatore ha sulla posizione dei pezzi avversari. Data una posizione dell'albero di gioco, chiamiamo figli AND i nodi corrispondenti a tentativi che si basano sullo stesso insieme informativo [9]. L'insieme dei figli di una posizione, se esistono insiemi informativi diversi, è partizionabile in diversi insiemi che chiamiamo nodi OR perché dovuti a informazioni interpretate in modi alternativi.

Dunque secondo [7] il fattore di diramazione si misura moltiplicando le cardinalità dei diversi insiemi OR.

Questo è corretto considerando l'aspetto formale dell'albero in cui vengono esaminate le mosse legali svolte per ciascuna posizione. Considerando l'albero in figura 3.11 la diramazione può infatti essere considerata pari a $2 \cdot 21 + 1 \cdot 19 = 61$, valutando opportunamente il fattore a 21 e 19.

Considerare le pseudomosse e le metaposizioni permette di ridurre il fattore di branching in modo considerevole. Come abbiamo visto, la cardinalità delle mosse pseudolegali è da considerarsi pari al massimo numero di mosse legali per ciascuna posizione; il fattore di diramazione diviene quindi pari alla massima cardinalità dei nodi OR. In questo modo ([9]), lo sviluppo passato della partita permette la deduzione di informazioni sulla possibile disposizione dei pezzi avversari, senza aumentare inefficacemente le possibilità di gioco.

Capitolo 4

Il finale Re e Torre contro Re

4.1 Introduzione

Lo studio di questo finale di partita elementare per il gioco degli scacchi è stato affrontato a partire dalla fine del XIX secolo con approccio procedurale, con la stesura cioè di una strategia 'ragionevole', raffinandosi nei decenni successivi in un approccio sostanziale, ossia nei termini di un cammino minimo che conduca alla soluzione del problema ([27]). In [5] è riportata una soluzione di questo finale per gli scacchi scritta in prolog.

Nel paragrafo 4.1.1 riporterò il primo storico tentativo, nel paragrafo 4.1.2 il calcolo delle simmetrie che porta a definire il numero minimo di posizioni considerabili nel finale KRK, calcolato già in [10].

Nei paragrafi 4.3 e 4.4 approfondirò alcuni insiemi di direttive e consigli formulati in [19] e [4] da cui è possibile ricavare una procedura completa che risolva questo finale algoritmicamente. La soluzione deterministica tracciata in [4] ha alcuni aspetti in comune con un lavoro recente riportato in [37], che si è rivelato utile nell'implementare i passi principali della procedura. Successivamente, nel paragrafo 4.5 approfondirò lo studio condotto e l'algoritmo di ricerca scritto per il finale KRK e scriverò a tal proposito un programma in grado di giocare questo finale, descrivendo una possibile struttura per la rappresentazione dei dati, un algoritmo di ricerca e di valutazione delle mosse specifico per il Kriegspiel che si distingue da un semplice approccio a forza bruta per l'utilizzo di accorgimenti euristici. Infine nel paragrafo 4.6 saranno mostrati i test e le analisi svolte sul giocatore artificiale.

Il programma presentato è un 'motore di gioco', poiché sviluppa solamente la componente algoritmica. Per poter permettere sviluppi futuri ed allargare la risoluzione del problema anche ad altri finali piuttosto che all'intero gioco del Kriegspiel, nella stesura del codice ho posto la mia attenzione a mantenere

generali le strutture impiegate; ad esempio ho preferito un vettore indicizzato su 6 elementi, rappresentanti i possibili pezzi del gioco completo, piuttosto che una unica variabile espressamente creata per la torre. Sempre a tal proposito ho scelto di impiegare il linguaggio di programmazione C, dando importanza alla velocità di esecuzione richiesta in un programma di questo tipo.

4.1.1 Torres y Quevedo

Nell'ultimo decennio del XIX secolo, con l'intenzione di dimostrare che molte delle attività che noi chiamiamo 'intelligenti' possono essere riprodotte meccanicamente, Torres y Quevedo ideò e costruì una macchina in grado di giocare il finale di re e torre contro re. Le informazioni circa la data di costruzione non sono precise, probabilmente egli costruì un prototipo intorno al 1890 che raffinò negli anni seguenti con sporadiche migliorie. La versione finale fu presentata all'Esposizione Universale di Parigi del 1914, ma venne dimenticata con l'avvento della prima guerra mondiale. Si trova oggi nella Escuela Tecnica Superior de Ingenieros de Caminos, presso il Politecnico di Madrid.

Come descritto in [2], la macchina non era efficiente, in quanto dava scacco matto sempre nella prima riga e poteva impiegare più di 50 mosse; tecnicamente la partita terminava in questo caso per patta.

L'algoritmo ideato da Torres y Quevedo è il seguente. La scacchiera è divisa in 3 parti: due zone per la torre laterali e le due colonne centrali (Fig. 4.1). La macchina domanda le seguenti domande nell'ordine presentato e, con risposta affermativa, esegue le azioni corrispondenti. Quindi aspetta per la mossa dell'avversario.

1. se il re nero si trova nella stessa zona della torre, allora muove la torre nella parte opposta della scacchiera (seguendo la mossa indicata in Fig. 4.1), altrimenti

2. se la distanza verticale tra il re nero e la torre è maggiore di 1, allora muove la torre in basso di una casa, altrimenti

3. se la distanza verticale tra i due re è maggiore di 2, allora muove il re bianco in basso di una casa, altrimenti

4. se la distanza orizzontale tra i re è dispari allora muove la torre di una casa orizzontalmente, altrimenti

5. se la distanza orizzontale tra i re è 0 allora muove la torre giù di una casa, altrimenti

6. muove orizzontalmente il re bianco di una casa verso il re nero.

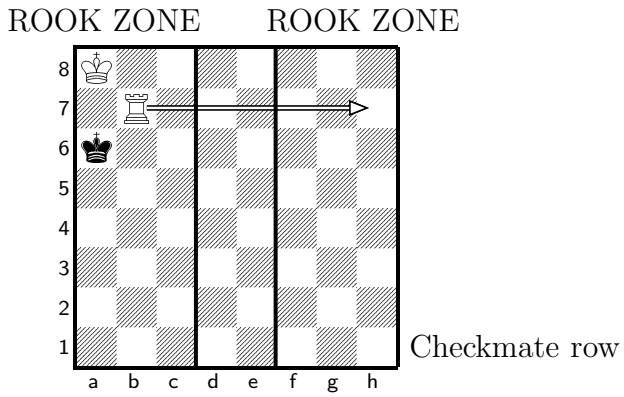


Figura 4.1: Schema iniziale della procedura di Torres y Quevedo

La strategia si presenta semplice, ma un avversario consapevole della sequenza decisionale del bianco può forzare la macchina a dare il matto in 61 mosse, dalla posizione di partenza in Fig. 4.1.

Un'altra caratteristica della strategia, che ne limita l'applicabilità, è la descrizione della posizione iniziale che prevede il re bianco in a8 e la torre in b7 come in figura e permette all'avversario di scegliere solo la casa di partenza del re nero.

4.1.2 Simmetrie

Come mostrato in [1], considerando le simmetrie nella disposizione dei pezzi in finali di partita senza pedoni, ogni posizione del re bianco può essere riflessa o ruotata per essere condotta nelle dieci case, dette canoniche, evidenziate in figura 4.2. In queste posizioni il re nero può occupare tutte quelle case non adiacenti al re bianco, inoltre, qualora il re bianco si presenti sulla diagonale delineata dalle case {a1, b2, c3, d4}, è possibile fare in modo che le posizioni occupate dal nero sopra la diagonale siano riflesse sotto la stessa. Il numero totale di posizioni legali in cui si possono disporre i due re eliminando le combinazioni ridondanti sulla scacchiera, diviene pari a 462.

Le dieci case {a1, b1, c1, d1, b2, c2, d2, c3, d3, d4} sono considerate le posizioni canoniche per il re bianco. Il posizionamento dei re sulla scacchiera può essere fatto tramite il procedimento descritto di seguito.

Riflessione del re bianco - la riflessione lungo gli assi indicati in figura 4.2 conduce il re bianco in una posizione canonica:

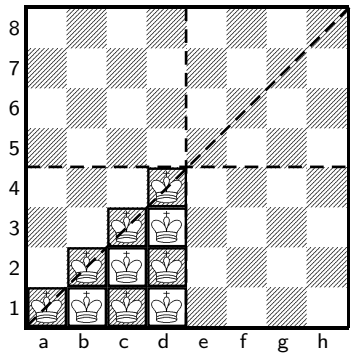


Figura 4.2: Posizioni canoniche

se il re bianco si trova sopra l'asse orizzontale (dalla riga 5 e maggiori), riflettere il re bianco al di sotto;

se il re bianco si trova a destra dell'asse verticale (dalla colonna e), riflettere il re bianco sulla parte sinistra della scacchiera;

se il re bianco si trova sopra la diagonale (da a1 a h8), riflettere la sua posizione sotto la diagonale (ovvero in una delle dieci posizioni canoniche).

Riflessione del re nero - se il re bianco si trova nelle case a1, b2, c3 o d4 e il re nero sopra la diagonale da a1 a h8, riflettere la posizione del re nero sotto la diagonale.

Nel calcolo totale delle posizioni è interessante considerare quelle legali, ove cioè il re nero non sia adiacente a quello bianco. Posto il re bianco in a1 e considerando il re nero riflesso nelle 36 case sotto la diagonale, delle quali 3 gli sono negate a causa dell'adiacenza con il re bianco, si ottengono 33 case possibili; posto il re bianco in b2, c3, d4 le posizioni possibili per il nero scendono a 30. Se il re bianco si trova sotto la diagonale è necessario considerare tutte le 64 posizioni per il nero a cui vanno sottratte quelle adiacenti, che sono 6 nel caso in cui il re bianco sia sulla prima riga, sono 9 altrimenti. Il calcolo completo è il seguente:

$$33 + 3 \cdot 30 + 3 \cdot 58 + 3 \cdot 55 = 462$$

Inserendo nella scacchiera la torre bianca[1], con un ragionamento analogo al precedente si possono diminuire le 28644 ($462 \cdot 62$) posizioni totali a 23424 posizioni legali, ove in particolare il re nero non risulta sotto scacco.

[1] non è in questo caso possibile considerare simmetrie

Come vedremo in seguito, nel gioco del Kriegspiel dovremo lavorare con metaposizioni, ovvero con posizioni che descrivono un insieme di posizioni, come vengono proposte in [33]. Graficamente queste saranno descritte da diagrammi con un numero di re neri variabile, che indica il grado di incertezza del giocatore bianco circa la posizione dell'avversario.

E' interessante valutare quante metaposizioni sia necessario affrontare per il finale KRK. Questo può essere approssimato fissando la posizione dei pezzi del bianco e considerando il numero di modi in cui è possibile scegliere n posizioni per il re nero fra le restanti posizioni libere. Se assumiamo come caso pessimo per il bianco ♖a1 e ♔b1, si hanno 52 possibili posizioni che non sono controllate dal bianco. Le possibili metaposizioni sono quindi:

$$\sum_{1 \leq n \leq 52} \binom{52}{n} = 2^{52} - 1 \tag{4.1}$$

Per queste posizioni le riflessioni del re nero rispetto alla diagonale a1-h8 descritte in [1], non diminuiscono la complessità numerica del problema con il quale ci stiamo confrontando. Non siamo quindi in grado di studiare completamente questo finale per forza bruta.

4.2 Il finale KRK su scacchiera infinita

Riporto in questo paragrafo l'interessante soluzione al problema del finale di re e torre contro re su scacchiera infinita, data da Llyod Shapley e Alexander Matros durante il Nono Festival di Teoria dei Giochi tenutosi a Stony Brook (NY, USA, 13-30/07/98).

Shapley espresse la sua convinzione che dare scacco matto usando il re ed una torre in Kriegspiel fosse sempre possibile e descrisse un metodo per potersi ricondurre dal caso su scacchiera infinita al caso finito [26].

Impostiamo il problema come segue:

1. La scacchiera è un quarto di un piano.
2. Il re e la torre bianchi sono posti come in figura 4.3.
3. Il re nero è posto in una posizione sconosciuta al bianco.
4. Il bianco deve giocare per la vittoria e vincere con il 100% delle probabilità.

La soluzione consiste di diversi passi:

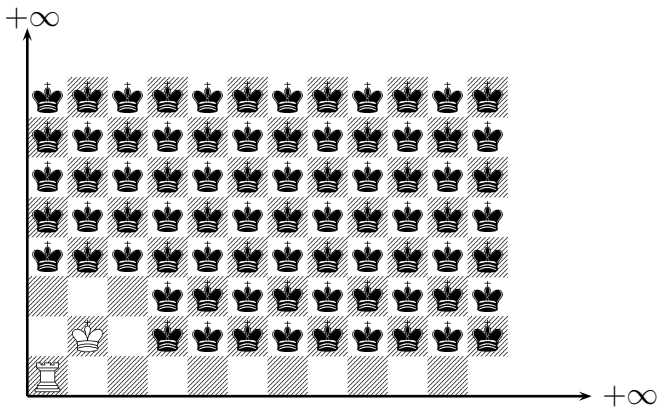


Figura 4.3: Shapley, 1960

Passo 1 - L'intento di questo passo è quello di dare scacco in modo da evitare che il re nero catturi la torre. Prima di tutto, è bene assicurarsi che il re nero non sia situato né nelle colonne a, b o c né nelle righe 1, 2 o 3. Il bianco gioca ♖a3, ♖b3, ♖c3, ♖c1, ♖c2, ♖c3. Se durante tali mosse il bianco da scacco si procede con il passo 2, altrimenti siamo sicuri che il nero è da qualche parte oltre la terza riga e la terza colonna.

Il bianco gioca ♖c1. Dopo avere mosso il re nero è oltre la seconda riga e oltre la destra della colonna c. E' necessario estrarre una colonna dalle 10 colonne dopo la terza (c) con probabilità di 1/10. A meno che la mossa non dia scacco, il bianco gioca ♖c1 e controlla nuovamente le prime tre righe giocando ♖c2, ♖c3, ♖c1. Quindi estrae con probabilità di 1/20 una colonna tra le 20 successive alla colonna c. Shapley chiamò questa procedura algoritmo a cinque mosse.

Procedendo come descritto l'algoritmo aumenta il numero di colonne a 30, 40, ecc. Riferendoci alla Teoria della Probabilità possiamo concludere che, prima o poi, il bianco darà scacco.

Una volta ricevuta la risposta di scacco dall'arbitro sappiamo che la posizione del re nero è ristretta come mostrato in Fig. 4.4.

Passo 2 - Si muove la torre 100 colonne a destra dalla colonna in cui si è dato scacco e si ripete la stessa procedura con l'intento di dare scacco su una riga. Quindi, con probabilità di 1/10 si muove la torre su una delle prime 10 righe. Se si da scacco si segue il passo 3, altrimenti si continua estraendo una riga su 20 con probabilità di 1/20. Se non si ottiene uno scacco dopo 97 mosse si sposta la torre 100 colonne a destra e si continua con l'algoritmo.

Grazie alla Teoria delle Probabilità possiamo concludere che prima o poi avverrà uno scacco con probabilità pari al 100%. Dopo lo scacco sappiamo

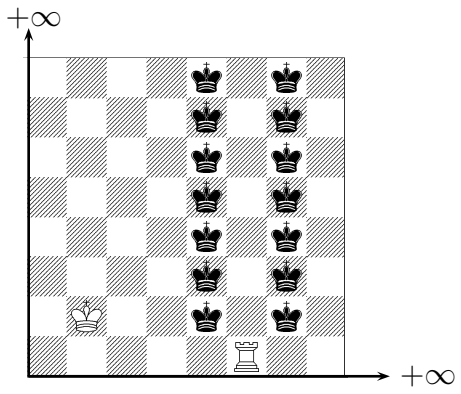


Figura 4.4: Posizione dopo lo Step 1

che la posizione del re nero è ristretta dal rettangolo mostrato in fig 4.5.

Passo 3 - Si muove la torre 100 righe sopra la riga dello scacco. Durante le 98 mosse successive il re bianco muoverà lungo la diagonale principale. Se non si ottiene la posizione descritta in figura 4.5, allora è possibile spostare la torre 100 colonne a destra e muovere il re bianco lungo la diagonale principale per le 98 mosse successive. Questo è il caso in cui il re bianco si muove di una velocità doppia rispetto alla velocità del re nero.

La posizione in figura 4.5 deve essere la conclusione dell'algoritmo.

Passo 4 - Si risolve il caso su scacchiera finita.

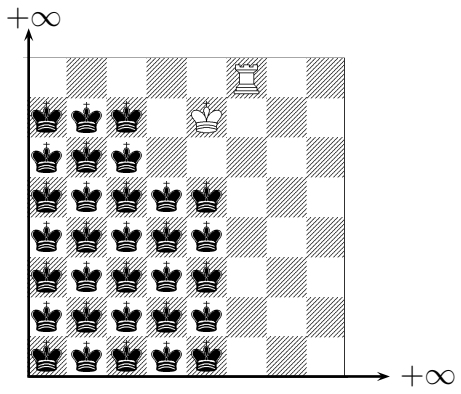


Figura 4.5: Posizione dopo lo Step 3

4.3 La soluzione di Magari

Un algoritmo per qualsiasi posizioni iniziale in cui il Bianco non abbia alcuna informazione circa il re nero, è data in [19]. La procedura comprende più passi.

Per prima cosa il Bianco deve giungere ad una configurazione simile a quella mostrata in figura 4.6.

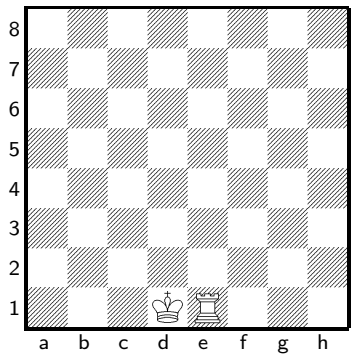


Figura 4.6: Posizione principale secondo la procedura di Magari

Il secondo passo consiste nel cercare il re nero con mosse del tipo ♔d2, ♖e2, ♔d3, ♖d3, ..., ♔d8, ♖d8:

se l'arbitro non dice mai "scacco" allora il re nero è nella metà di sinistra della scacchiera, altrimenti è possibile velocizzare la strategia e considerare opportune mosse a seconda della posizione in cui ci si trova. Mostriamo la prima ipotesi nella metaposizione in figura 4.7

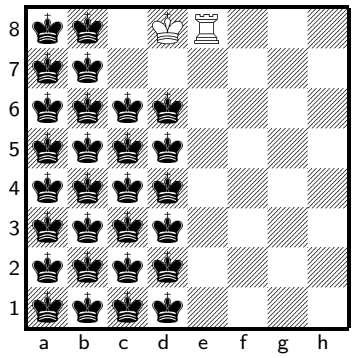


Figura 4.7: Il Nero è costretto nella metà sinistra della scacchiera

Magari descrive questo diagramma con un'analogia con il concetto di "onda di possibilità" della fisica quantistica. Secondo questa metafora il Re Nero nel diagramma non va considerato come un corpo con una posizione ben precisa, quanto piuttosto un'*onda*, ovvero "un insieme di possibilità". È il Re

Bianco che deve dissipare quest'onda addentrandovisi ([9]). Nei diagrammi in figura 4.8 viene descritta la manovra avvolgente del Re Bianco, che procede sistematicamente se l'arbitro non dice mai "mossa illegale".

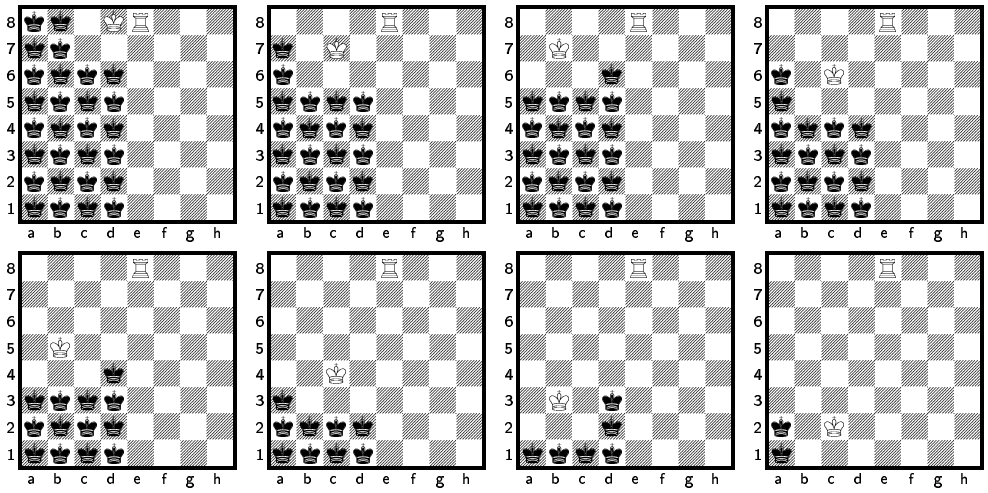


Figura 4.8: Procedura di Magari

Nella posizione finale di questa fase, dove il Re Bianco si trova in c2, il Bianco matta con ♖a8#.

Se in qualche punto l'arbitro dice "mossa illegale" il Bianco individuerà più facilmente la posizione del re nero e potrà usare la torre (facendo attenzione a non lasciarla indifesa) per costringere più rapidamente alla resa il Nero.

4.4 La soluzione di Boyce

Analizzeremo ora la soluzione fornita da Jim Boyce in [4] al finale di Re e Torre contro Re in Kriegspiel. Egli considera solo le posizioni iniziali dove il Bianco può conseguire lo scacco matto contro qualsiasi difesa avversaria, anche la più preveggente.

La strategia del Bianco consiste di diversi passi. Per prima cosa, si accerta che la torre sia salva dalla cattura da parte del nero, quindi conduce il gioco ad una posizione dove tutte le possibili case del re nero sono comprese in un rettangolo che ha un angolo in comune con la scacchiera e l'angolo opposto descritto dalla torre. Il Bianco vuole che il suo re sia in questo rettangolo in modo da evitare che la torre sia sotto attacco. Quindi il nero viene costretto ad occupare sempre un'area minore (in termini di righe e colonne) sulla scacchiera e raggiunge uno stato in cui è facile dare scacco matto.

4.4.1 La strategia e le direttive che la compongono

L'articolo in [4] descrive un insieme di direttive semplici che considerano le seguenti possibili posizioni:

1. Entrambi i re sono nello stesso quadrante della scacchiera delineato dalla torre bianca, come mostrato in figura 4.9.

2. Il re nero è racchiuso in uno o due quadranti della scacchiera.

3. La torre bianca è sicura.

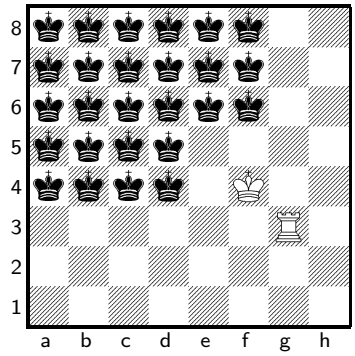


Figura 4.9: Posizione iniziale nella procedura di Boyce

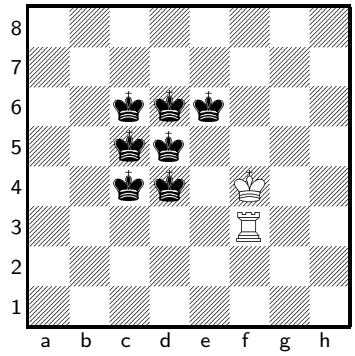


Figura 4.10: 1. ♔e4 I, ♖f3

È possibile approfondire le posizioni del primo tipo nel modo seguente:

1a Re nero costretto ad una riga (o colonna)

1b Re nero costretto a due righe

1c Re nero costretto a tre righe

1d Re nero costretto a quattro righe

1e Re nero costretto a più di quattro righe

1 - Consideriamo la figura 4.9 . Il bianco tenta 1. ♔e4 I, ♖f3. Ottiene così la posizione descritta in figura 4.10.

Dopo la mossa del nero, il bianco tenta:

2. ♔e4 I, ♔e3 I, ♔f5

raggiungendo la miniatura in figura 4.11.

Ora il bianco gioca:

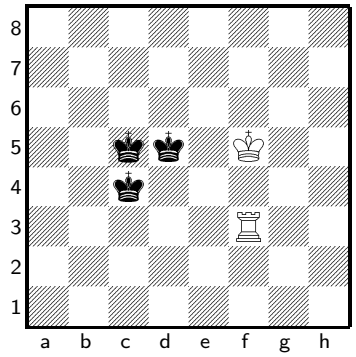


Figura 4.11: Diagramma raggiunto dalla fig.4.10 giocando 2. ♔e4 I, ♔e3 I, ♔f5

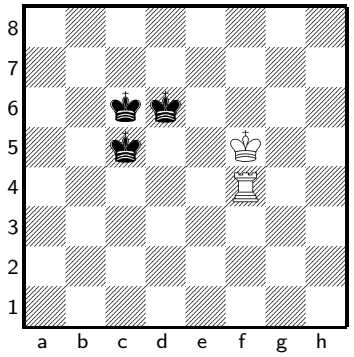


Figura 4.12: Diagramma raggiunto dalla fig.4.11 giocando 3. ♔e4 I, ♖f4

3. ♔e4 I, ♖f4

Questa particolare sequenza di mosse riduce le dimensioni del rettangolo e conduce ad una posizione simile a quella ottenuta dal bianco dopo la sua prima mossa, descritta in figura 4.12, permettendo quindi di reiterare il procedimento.

Se la mossa ♔e4 giocata in 2. o 3. risulta legale, allora la posizione ottenuta è simile a quella in figura 4.9 ed è possibile reiterare la procedura.

Se invece tale mossa è legale in 1. allora si procede con 2. ♖f3. Se la risposta da parte dell'arbitro è silente ritorniamo ad una posizione simile a quella iniziale in 4.9; se invece si riceve la risposta di scacco allora il bianco gioca

2.♖f3 +; 3. ♖f4

ottenendo la miniatura in figura 4.13, che lo conduce in una posizione del secondo tipo.

L'ultima possibilità nelle posizioni del primo tipo è che 2. ♔e3 sia legale. In questo caso il bianco gioca

3. ♔e4 I, ♖f4

ottenendo quanto descritto in figura 4.14.

Si noti che in figura 4.14 i due re non sono più nello stesso rettangolo. Boyce mostra che il re bianco può ritornare nel quadrante senza aumentare la dimensione dello stesso. Nel caso in figura 4.14 ad esempio con

1. ♔f3 2. ♔g4 3. ♔g5 4. ♖h4

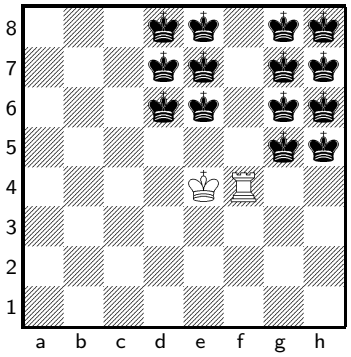


Figura 4.13: Posizione del secondo tipo

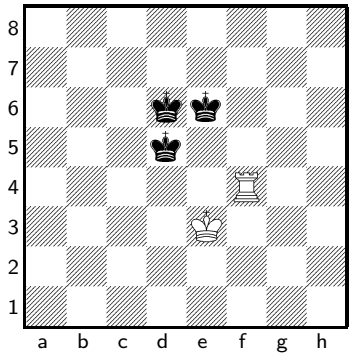


Figura 4.14: Diagramma ottenuto dalla fig. 4.10 giocando 2. ♔e4 I, ♔e3; 3. ♔e4 I, ♖f4

2 - La figura 4.15 mostra una posizione in cui il re nero può essere posizionato in due quadranti adiacenti. In questa situazione il bianco può giocare, ignorando eventuali messaggi di scacco, la strategia seguente:

1. ♔b2 2. ♖c3 3. ♔b3 4. ♔b4 5. ♖a3

ottenendo la posizione in 4.16.

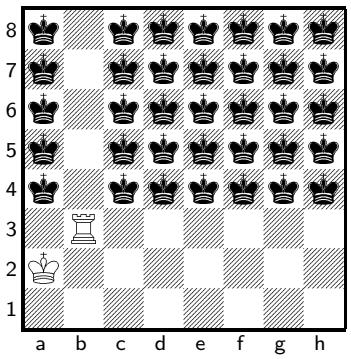


Figura 4.15: Posizione del secondo tipo

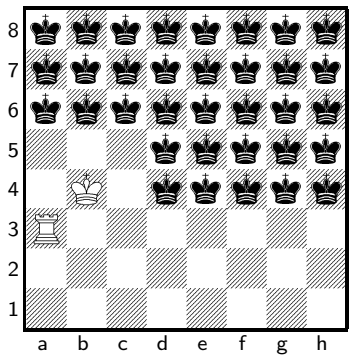


Figura 4.16: Diagramma ottenuto dalla fig. 4.15 giocando 1. ♔b2 2. ♖c3 3. ♔b3 4. ♔b4 5. ♖a3

3 - In qualunque posizione La torre venga a trovarsi, essa deve essere salva. A questo scopo raggiungere e mantenere posizioni del tipo indicato in figura 4.9 aiuta ad assicurare la difesa della torre.

1a - L'analisi precedente non considera le posizioni in cui entrambi i re sono nello stesso quadrante se esso è composto da un'unica riga o colonna.

La figura 4.17 mostra una posizione di questo tipo. Il giocatore bianco può spingere il re nero nell'angolo della scacchiera e forzare così lo scacco matto, ma nel farlo deve fare attenzione a non incorrere in casi di stallo.

Dalla figura 4.17 il gioco si può concludere con

1. ♔d8 I, ♖g7 2. ♔d8 3. ♔c7 (e non 3. ♔c8=) 4. ♖g6 5. ♖a6#

Quando il re nero impedisce una mossa di re, la torre può essere mossa sulla stessa riga costringendo il re nero a ritirarsi e permettendo nella mossa successiva la mossa di re precedentemente impedita.

1b - Nel caso in cui il re nero sia confinato in due righe (o colonne) il bianco può migliorare la strategia descritta nel caso generico[2], poiché non è necessario avanzare la torre per spingere il re avversario all'angolo. Dalla figura 4.18 il gioco può continuare con

♔e7 2. ♔d7 I, ♔d6 3. ♔d7 4. ♔c7 I, ♔c6 I, ♔d8 5. ♔c7 6. ♔b6 I, ♔c8 7. ♖a6#

Se in 5. ♔c7 è illegale il bianco può giocare 5. ... ♖h7. Due mosse dopo, 7. ♖h7= si rivelerebbe un grave errore.

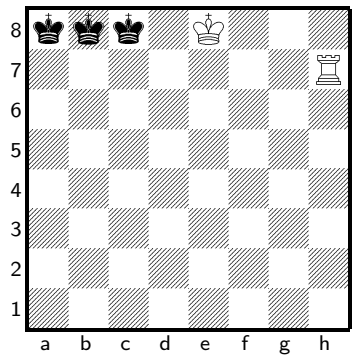


Figura 4.17: Il re nero è costretto ad una riga

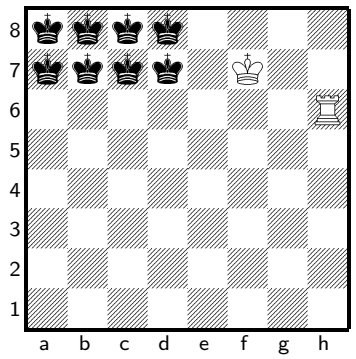


Figura 4.18: Il re nero è costretto a due righe

1c - Se il re nero è confinato a tre righe è possibile trovare alcune posizioni in cui le direttive espresse precedentemente possono essere migliorate. Ad esempio in figura 4.19 il re nero può essere in uno dei due quadranti delineati dopo che il bianco a mosso ♖f5+. Una strategia efficace è

1. ♖f6 2. ♔e7

[2]indicato a partire dalle posizioni del primo tipo

Se ♔e7 è legale, il re nero si trova nel quadrante di destra, altrimenti in quello di sinistra.

La figura 4.20 mostra un ulteriore caso in cui il bianco può migliorare la propria strategia, egli può dare scacco matto giocando

1. ♔c7 I, ♖e6 [3] 2. ♔d7 I, ♔d6 3. ♔e7

e dà scacco matto in nove mosse come in **1b**.

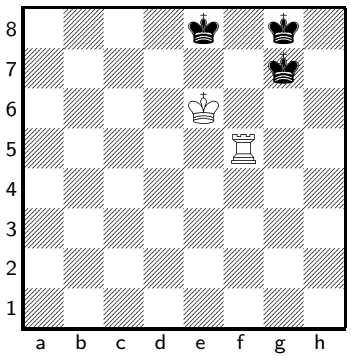


Figura 4.19: Il re nero è costretto a tre righe

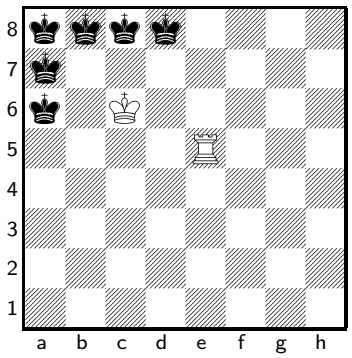


Figura 4.20: Il re nero è costretto a tre righe

1d - Consideriamo ora il caso in cui il re nero sia confinato a quattro righe, come in figura 4.21 che mostra una miniatura simile a quella in figura 4.20. Il bianco ha intenzione di giocare ♖f4. Se gioca la mossa ora e da scacco, il re nero sarà in uno dei due quadranti descritti dalla torre, ma il bianco sarà troppo lontano dall'angolo per la strategia ideata nella miniatura di figura 4.19. Il bianco può quindi seguire una strategia più veloce che quella espressa nel punto **2** giocando 1. ♔e6 I, ♖g5 riducendo la dimensione più piccola tra quella delle righe e quella delle colonne. Se è legale può quindi giocare 2. ♖f4 ed ora, se riceve scacco, si trova nel caso descritto in figura 4.19.

1e - Se il re nero è confinato a più di quattro righe l'azione del giocatore bianco tende a restringere l'area il più velocemente possibile. dalla figura 4.22 il gioco può continuare con

1. ♖g1 2. ♔e3 3. ♖f1

Se una delle due mosse di torre da scacco, la mossa successiva del bianco consiste nello spostare di due case a sinistra la torre in modo da ottenere una posizione in cui entrambi i re sono nello stesso quadrante e la dimensione più piccola è al più di quattro case.

[3] se ♔c7 è legale il bianco può vincere con un gioco simile, ma più semplice

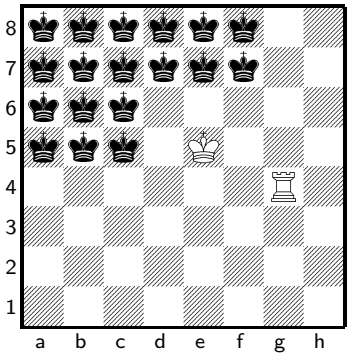


Figura 4.21: Il re nero è costretto a quattro righe

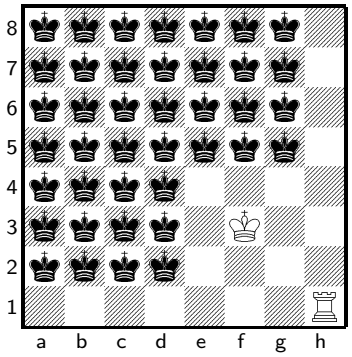


Figura 4.22: Il re nero è costretto a più di quattro righe

4.4.2 L'algoritmo

La strategia di Boyce, come detto, consta di una serie di norme e direttive formulate in base alla posizione in cui il giocatore bianco si trova, ma non descrive una vera e propria procedura meccanica indipendente, che è invece ciò a cui vorremmo giungere. Il mio lavoro, in questo frangente, è stato quello di ridefinire alcuni passi e collimare le diverse direttive in modo da ottenere un motore di gioco unico per il finale di re e torre contro re.

Per prima cosa è necessario allargare il campo d'azione a tutte le possibili posizioni che si possono verificare inizialmente e non solo ai casi favorevoli, ovvero a quelle posizioni in cui la torre è salva e in cui ci si trova in una posizione del primo o secondo tipo descritta da Boyce.

Poiché il numero totale di posizioni in cui il bianco può incorrere corrisponde al numero di metaposizioni calcolato in 4.1 non era possibile creare uno schema di mosse ed è stato quindi scritto un algoritmo di ricerca per favorire le mosse del bianco che più probabilmente lo possono portare in una posizione iniziale.

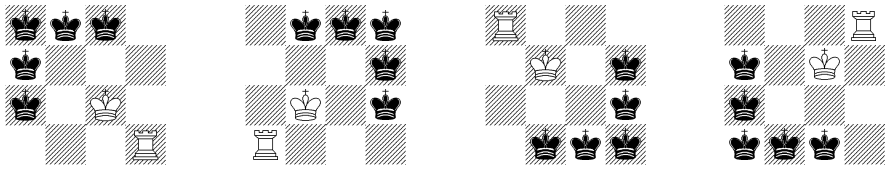


Figura 4.23: Le quattro posizioni iniziali

Nella procedura per passi, ispirata alle direttive di Boyce, le posizioni iniziali sono del tipo mostrato in figura 4.23 in cui il re bianco si trova in una casa adiacente alla torre e diagonale ad essa. In particolare si fa riferimento

alla miniatura in figura 4.9 e se ne considerano le quattro possibilità ottenute per riflessione e per simmetria.

Per poter avviare la procedura è innanzitutto necessario giungere ad una delle posizioni iniziali. Per fare questo si è fatto uso di un algoritmo di ricerca che visita l'albero delle pseudomosse fino alla profondità richiesta e ne valuta i nodi. Riporto in tabella 4.1 lo pseudocodice relativo a tale algoritmo sebbene non abbia ancora approfondito fino ad ora le tecniche implementative scelte nella stesura del giocatore artificiale, queste verranno adeguatamente svolte nella sezione 4.5. Mi limito ad indicare il compito svolto da alcune funzioni richiamate dall'algoritmo. La funzione $playPlayerMove(c, m, ans)$, come riportato nella sezione 4.5.6 aggiorna la mossa m nella scacchiera di riferimento del giocatore di colore c avendo ricevuto dall'arbitro risposta ans; la funzione $updateAfter(c, ans, cap)$ aggiorna la scacchiera di riferimento del giocatore c con la metamossa dell'avversario ipotizzando che essa abbia ricevuto risposta ans; infine $playerTakeBack(c)$ ripristina la scacchiera di riferimento del giocatore c a prima dell'aggiornamento. Come detto, rimando alla sezione 4.5 per ulteriori dettagli su ciò che intendo per "scacchiera di riferimento" e "metamossa" e alla sezione 4.5.6 per delucidazioni sulle funzioni a cui si è fatto riferimento.

La funzione di utilità riportata in tabella 4.2 esclude le mosse che portano la torre bianca in uno stato di possibile attacco da parte del re nero o penalizza tali mosse, nel caso in cui queste non siano evitabili [4]. La funzione ritorna invece un alto valore se si raggiunge una posizione iniziale nella procedura di Boyce, come quelle mostrate in figura 4.23; inoltre favorisce le posizioni in cui la torre è adiacente al re.

Il diagramma di flusso in figura 4.24 mostra la procedura seguita dal giocatore artificiale a partire dalla posizione iniziale A. Le etichette sugli archi indicano una risposta silente da parte dell'arbitro (S), la risposta che indica una mossa illegale (I) e la risposta che avverte dell'avvenuto scacco ($Check$); le mosse sono invece indicate con la miniatura del pezzo che muove seguita dalla direzione in cui muove, che viene rappresentata dai punti cardinali n (north),s (south),e (est),w (west).

È interessante notare la simmetria del diagramma che sottolinea le due direzioni principali in cui il Bianco può tentare di arginare la "macchia" dei re neri. Egli può infatti ridurre la dimensione delle righe oppure quella delle colonne, muovendo quindi il re inizialmente verso ovest (w) oppure verso nord (n).

Le figure 4.25 e 4.26 mostrano i diagrammi relativi alla procedura imple-

[4] può capitare inizialmente quando il re e la torre bianchi si trovano in righe e colonne differenti

```
BoyceSearch (int c, int depth, int *from, int *to, int ans) {
    int v, max, f, t;
    max=−100;
    if(depth==0)
        return BoyceEvaluate(c);

    Generate the white's legal moves Γ;

    for each moves j ∈ Γ
    {
        if(j already played)
            continue;

        playPlayerMove(c,j,NONE);
        if(ans!=NONE)
            updateAfter(c,NONE,cap);
        else
            enemyWorst(c);

        if(v==50) {
           *from=j.from;
           *to=j.to;
           playerTakeBack(c);
           return v;
        }
        if(v==−100 || v==−5) {
           playerTakeBack(c);
           continue;
        }

        v+=BoyceSearch(c,depth-1, &f, &t, ans, dir);
        if(v>max) {
           *from=j.from;
           *to=j.to;
           max=v;
        }

        playerTakeBack(c);
    }

    return max;
}
```

Tabella 4.1: L'algoritmo di ricerca basato sulla procedura di Boyce

```
INPUT:  c player color
OUTPUT: integer that represents the value of the metaposition

int BoyceEvaluate (int c) {
  int i, res, pos, king, rook;

  res=0;
  for each squares i on the board {
      if(WR is under attack on i) {
          res=-100;
          return res;
      }
      pos=boycePosition(c);
      if(pos is an initial position) {
          res+=50;
          return res;
      }
  }
  if(WR is adjacent to WK)
      res+=5;
  else {
      res-=5;
      return res;
  }
  return res;
  }
```

Tabella 4.2: La funzione di valutazione per l'algoritmo *BoyceSearch*

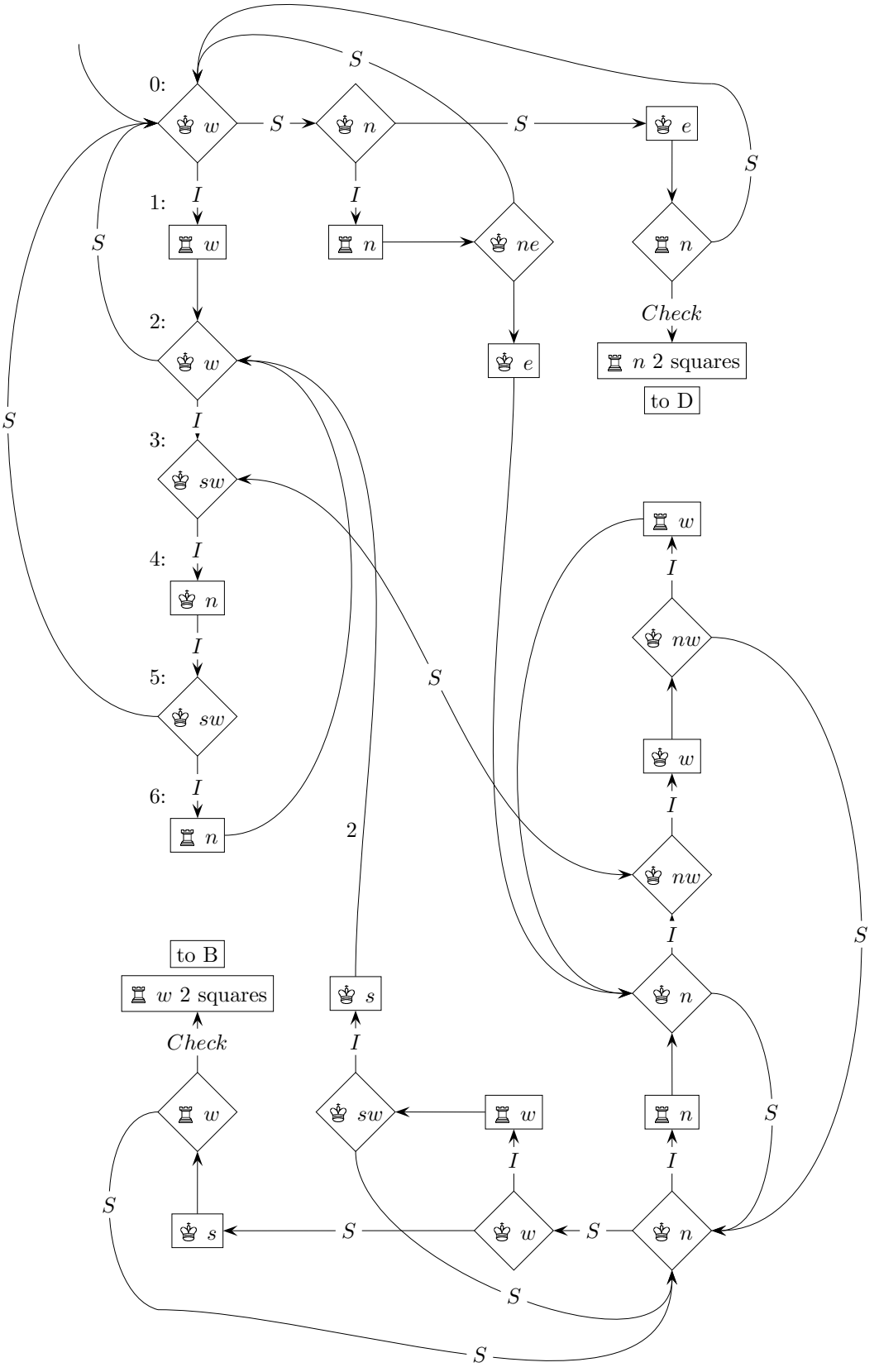


Figura 4.24: Diagramma delle mosse dalla posizione iniziale A

mentata nel giocatore artificiale. Rispettivamente la prima raffigura lo schema delle mosse seguito a partire dalla posizione descritta in figura 4.9, dopo aver giocato la mossa ♔e4 con risposta "mossa illegale"; la seconda riporta il diagramma delle mosse seguito a partire sempre dalla posizione descritta in figura 4.9 ma dopo aver giocato la mossa ♔e4 con risposta dell'arbitro "silente".

In entrambe sono indicate le mosse e le miniature d'esempio partendo dalla posizione iniziale del primo tipo, ma è chiaro che nell'implementazione si è tenuto conto delle restanti tre posizioni iniziali ottenute con simmetrie e riflessioni. I nodi del diagramma che contengono la voce "goto n" indicano che, dalla posizione ottenuta, la procedura continua tornando ai nodi del livello n (indicati con la linea tratteggiata); "goto 0" ad esempio indica che l'algoritmo procede nuovamente dalla posizione iniziale. Laddove è presente la voce "goto n simmetrico" si intende che il passo da seguire è al nodo di livello n che tende a diminuire la dimensione dello spazio delle righe se fino ad ora si stava diminuendo lo spazio delle colonne e viceversa. L'etichetta "goto X pos." indica che la procedura si sposta in uno dei quattro diagrammi riferito alla posizione iniziale X, che risulta essere una posizione iniziale riflessa (o per righe, o per colonne) di quella corrente.

Quando l'algoritmo confina il re nero in una o due righe, o simmetricamente una o due colonne, la sequenza di passi del giocatore artificiale non segue più quella proposta nel diagramma, ma diviene fedele alle direttive proposte da Boyce e riportate nella sezione precedente. Visto la semplicità, non riporto un diagramma di flusso che le rappresenta.

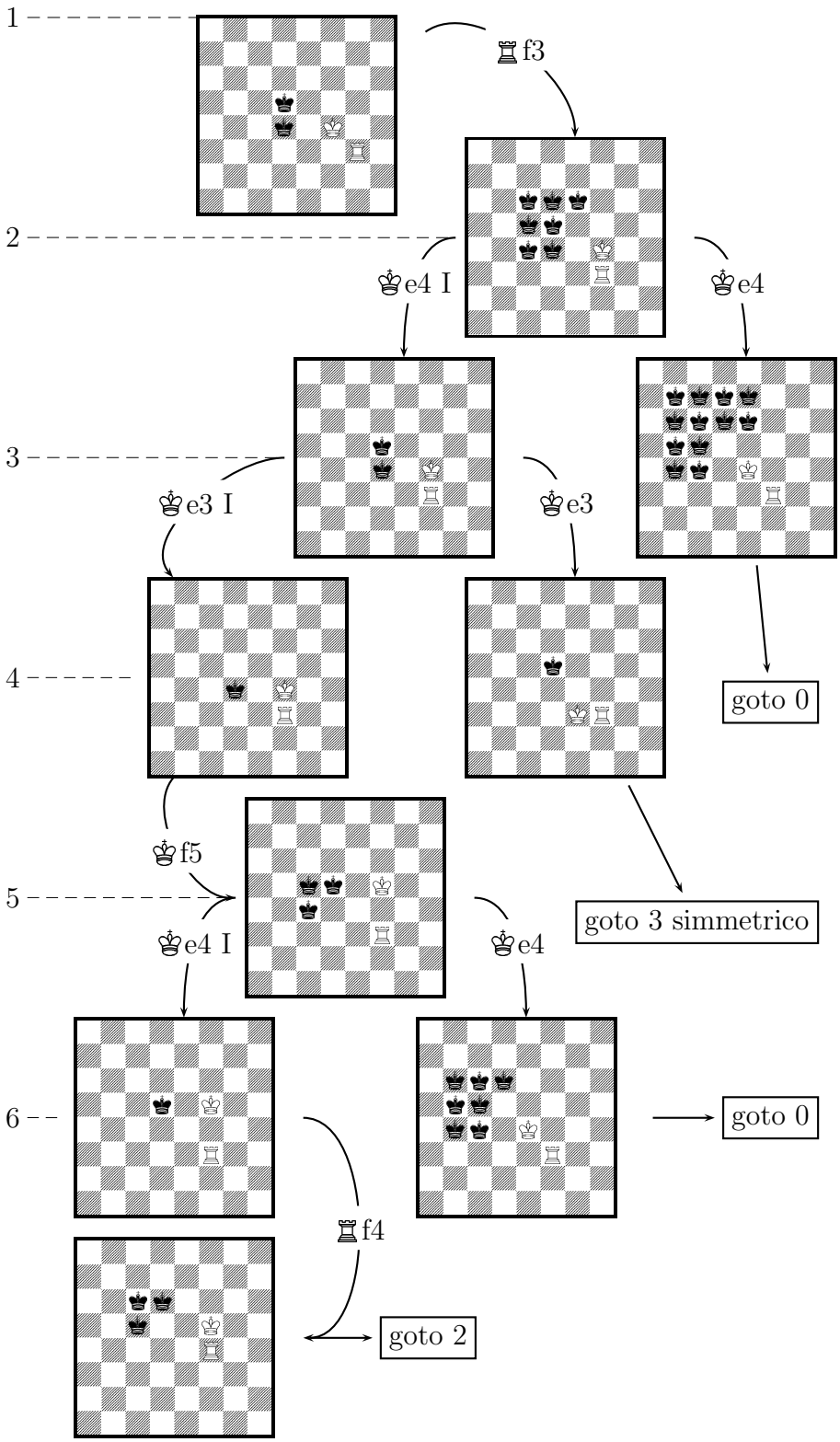


Figura 4.25: Diagramma grafico della procedura di Boyce, primo caso

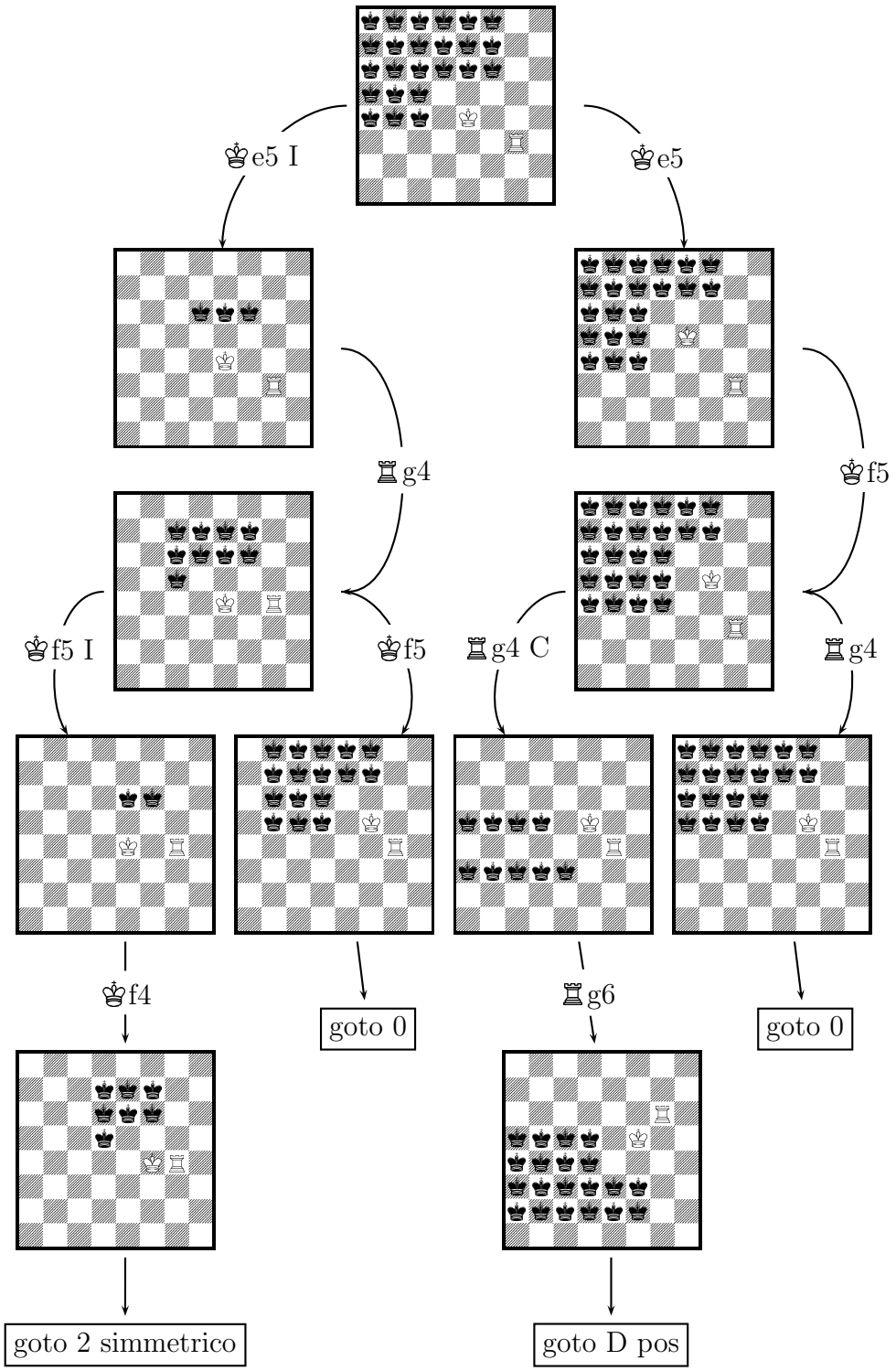


Figura 4.26: Diagramma grafico della procedura di Boyce, secondo caso

4.5 La soluzione tramite algoritmo di ricerca

4.5.1 La struttura dell'arbitro

L'arbitro, il cui codice è inserito nei file $referee.c$ e $referee.h$, deve essenzialmente ricevere le mosse o i tentativi di mosse da parte dei giocatori e restituire delle risposte nel caso di scacco o di illegalità, o tacere nel caso normale. La mossa viene giocata tramite la funzione $playMove$, in seguito alla quale le possibili risposte che l'arbitro restituisce sono le seguenti:

Check	la mossa ha dato scacco
Illegal	la mossa è illegale
0-1 {Black mates}	il Nero dà scacco matto
1-0 {White mates}	il Bianco dà scacco matto
0-0 {Stalemate}	la partita termina per stallo
0-0 {Draw by fifty moves rule}	la partita termina per la regola delle 50 mosse

Sebbene le due ultime risposte non siano in realtà contemplate nel gioco del Kriegspiel, ho preferito comunque lasciare nell'arbitro l'implementazione del conteggio per questi casi di terminazione in modo da non escludere eventuali variazioni sulle regole del gioco. Nel caso specifico del finale di torre è chiaro che la prima risposta non sarà mai data, potendo il gioco terminare per vittoria del bianco o per stallo.

4.5.2 La rappresentazione della scacchiera

La rappresentazione della scacchiera dell'arbitro è costituita da due vettori di interi $piece[x]$ e $color[x]$ che indicano rispettivamente la presenza di un pezzo nella casa x appartenente al giocatore di colore $color[x]$. Il vettore dei pezzi può assumere i valori costanti dell'insieme {PAWN, KNIGHT, BISHOP, ROOK, QUEEN, KING, EMPTY}, rispettivamente $\{1, 2, 4, 8, 16, 32, 64\}$; come si può notare ho assegnato a ciascuna costante un valore pari ad una potenza di 2, questo tornerà utile nella formulazione della scacchiera e quindi delle mosse dei giocatori.

Chiamerò "scacchiera di riferimento" del Bianco (o dualmente del Nero) la struttura dati nella quale i giocatori mantengono traccia delle posizioni dell'avversario.

I vettori *color* e *piece* hanno la medesima semantica dei corrispondenti definiti per la scacchiera dell'arbitro; il motivo per cui sono indicizzati su 128 elementi e non su 64, ossia sul numero delle case (come si potrebbe supporre), è dovuto a motivi prettamente tecnici, dettati cioè da necessità sulla velocità

di calcolo e saranno presto spiegati. Il vettore di booleani *attack* tiene traccia delle case in cui vi è presente un pezzo che corre il rischio di cattura. L'intero q tiene traccia del numero di pezzi in gioco dell'avversario. Infine l'intero a 64 bit *key* rappresenta la chiave di Zobrist che mappa la posizione presente nella scacchiera e che sarà sfruttata nella gestione della tabella di trasposizione.

La "scacchiera di riferimento" è quindi definita tramite la struttura seguente:

```
typedef struct {
    int color[128];
    int piece[128];
    BOOL attack[128];
    int q;
    long long key;
} board;
```

Il vettore *boardbo*[2], indicizzato sui colori dei giocatori {WHITE,BLACK}, diversifica le due scacchiere per i giocatori, questo perché non potendo vedere le posizioni dell'avversario ciascuno di essi deve fare affidamento ad una proprio set informativo indipendente, che come detto chiameremo "scacchiera di riferimento".

4.5.3 La rappresentazione delle mosse

Ho rappresentato la mossa tramite una struttura chiamata *move* che tiene traccia della casa di partenza (tramite l'intero $from$), della casa di arrivo (to), del pezzo che sta muovendo (who), infine tiene traccia del valore (v) che indicherà la valutazione della mossa da parte di una funzione apposita che vedremo nel seguito.

```
typedef struct {
    int from;
    int to;
    int who;
    int v;
} move;
```

Questa struttura viene utilizzata sia dall'arbitro che dai giocatori, nel caso dell'arbitro però, il valore v non viene considerato, poiché non viene assegnato in questo caso nessun giudizio alla mossa.

4.5.4 Lo schema 0×88

Le celle di una scacchiera regolare vengono normalmente indicizzate con numeri da 0 a 63, ove 0 indica la posizione a8, 1 indica b8, fino a 63 che indica h1. Nella soluzione 0×88 suggerita in [28], vengono aggiunte 8 ulteriori colonne ottenendo una scacchiera formata da 128 celle.

È possibile immaginare due scacchiere 8x8 affiancate, ove a8..h8 sono indicizzati da 0..7 e alla destra della colonna h vi sono 8 celle inutilizzate, in questo modo a7 diviene 16, b7 diviene 17, a6 diviene 32 e così via.

La formula per ogni cella diviene $row \cdot 16 + column$. Quanto detto è mostrato in figura 4.27.

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
16	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
32	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**	**43**	**44**	**45**	**46**	**47**
48	**49**	**50**	**51**	**52**	**53**	**54**	**55**	**56**	**57**	**58**	**59**	**60**	**61**	**62**	**63**
64	**65**	**66**	**67**	**68**	**69**	**70**	**71**	**72**	**73**	**74**	**75**	**76**	**77**	**78**	**79**
80	**81**	**82**	**83**	**84**	**85**	**86**	**87**	**88**	**89**	**90**	**91**	**92**	**93**	**94**	**95**
96	**97**	**98**	**99**	**100**	**101**	**102**	**103**	**104**	**105**	**106**	**107**	**108**	**109**	**110**	**111**
112	**113**	**114**	**115**	**116**	**117**	**118**	**119**	**120**	**121**	**122**	**123**	**124**	**125**	**126**	**127**

Figura 4.27: Il sistema 0×88

L'utilità di questa scacchiera è quella di permettere una intuitiva e veloce generazione delle mosse. Supponiamo di lavorare con una scacchiera 8×8 e di avere una torre nella posizione a6 indicizzata con 16. Una funzione di generazione delle mosse per generare una mossa di una cella sulla stessa colonna somma 8 all'indice della cella di partenza, nell'esempio si ottiene 24. Per poter continuare sulla stessa colonna deve essere verificata la condizione che l'indice di destinazione sia minore di 64. Nell'esempio lo è, pertanto si continua iterando il procedimento ottenendo i valori 32, 40, 48, 56, dopo il quale si ottiene 64 che risulta essere fuori dalla scacchiera. Sulla stessa colonna sono possibili le mosse in direzione opposta a quelle calcolate, pertanto la funzione di generazione ripartirebbe da 24 decrementando di 8 l'indice ad ogni passo, ottenendo 8, 0, ed infine -8 che risulta essere fuori dalla scacchiera poiché minore di 0. Il test effettuato in questo procedimento è il seguente:

$$\text{if } ((\text{index} < 0) \,||\, (\text{index} >= 64))$$

Tale test, in realtà composto da due vincoli, può essere migliorato riducendolo ad un vincolo solo nel modo seguente

if (!(index & 0×40))

che comprende entrambi i casi, sia che l'indice sia fuoriuscito dalla scacchiera dal basso che dall'alto. Nel primo caso il bit 0×40 è pari ad 1 in quanto il valore è maggiore o uguale a 64 e nel secondo caso tale bit è 1 per via della rappresentazioni in complemento a due dei numeri negativi.

Questo metodo risolve elegantemente i casi di generazione di mosse verticali, ma non quelli nel caso di mosse orizzontali, ovvero su una stessa riga. Ad esempio supponiamo che la torre sia in a8 potremmo aumentare l'indice di 1 fino ad arrivare ad h8, indicizzato con 7, che incrementato ulteriormente ritornerebbe 8 ossia a7 che è sempre nella scacchiera, mentre avremmo dovuto terminare la generazione delle mosse. Lo stesso accade andando verso sinistra, decrementando di 1 l'indice.

La scacchiera 0×88 risolve la generazione delle mosse in orizzontale offrendo l'ausilio del bit sentinella 0×08, spento in tutti gli indici della scacchiera di sinistra e acceso in tutti quelli della scacchiera di destra. Perciò se la torre si trova in h8 incrementando l'indice otteniamo 8 che ha il bit 0×08 acceso e interrompiamo la generazione; allo stesso modo dalla cella a7, che ha il bit 0×08 spento, decrementando l'indice 16 otteniamo 15 che ha invece il bit sentinella acceso e interrompiamo la ricerca. Il ragionamento precedente fatto sulla scacchiera standard di 64 celle per gli spostamenti verticali può essere riproposto nella scacchiera a 128 celle e tale ragionamento può essere combinato con il bit sentinella, ottenendo il vincolo seguente

if (!(index & 0×88))

da cui prende il nome la rappresentazione.

4.5.5 La generazione delle mosse utilizzando lo schema 0×88

Un esempio di funzione generatrice per le mosse del re di colore *colour* sulla scacchiera dell'arbitro a partire dalla posizione indicata dall'intero *square* è dato dalla funzione seguente:

```
GenerateNormalMoves (int square, int *ptab, int colour) {
  for (; *ptab; ptab++) {
     int index = square + *ptab;
     if(!(index & 0x88))
          if(colour!=color[index])  GenerateMove(square,index);
  }
}
```

ove il vettore di interi che le viene passato durante la chiamata è

```
int KingMoves[] = { 17, 16, 15, 1, -17, -16, -15, -1, 0 };
```

che rappresenta tutte le possibili direzioni in cui si può spostare il re.

Oltre al test 0×88 viene imposto un ulteriore vincolo per evitare che la generazione continui una volta incontrata una cella già occupata da un proprio pezzo.

La funzione di generazione per le mosse di pezzi che possono compiere salti, come l'alfiere, la torre e la regina è leggermente più complessa:

```
GenerateSlidingMoves(int square, int *ptab, int c) {
 for (; *ptab; ptab++) {
   int index;
   for(index=square+*ptab; !(index&0×88); index+=*ptab)
        if(piece[index]==EMPTY) {
            GenerateMove(square,index);
        } else {
            if(c!=color[index])
               GenerateMove(square,index);
               break;
        }
  }
}
```

Riporto infine il vettore per le direzioni della torre utilizzate nel programma, tralasciando i vettori relativi alle mosse della regina, dell'alfiere e del cavallo poiché seguono lo stesso principio.

```
int RookMoves[] = { 16, 1, -16, -1, 0 };
```

La funzione *GenerateMove* si occupa di riempire un vettore d'appoggio $refereeMove$ indicizzato sull'intero $moveIndex$ e di incrementare tale indice.

```
void GenerateMove(int from, int to) {
    refereeMoves[moveIndex].from=from;
    refereeMoves[moveIndex].to=to;
    ++moveIndex;
}
```

Questo vettore sarà utile durante l'algoritmo di ricerca per rilevare casi illegali e inferire posizioni sulla scacchiera.

4.5.6 La funzione che gioca la mossa e quella che la ritratta

Dopo aver esaminato il caso relativo alla generazione delle mosse pseudolegali, affronto in questo paragrafo la funzione incaricata di giocare la mossa sulla propria scacchiera. Poiché il giocatore può ritenere giocabili anche mosse illegali, essendo inconsapevole della reale posizione dei pezzi avversari, nel giocare la mossa egli aggiorna la propria scacchiera di riferimento. Qualora una mossa si dimostri illegale si possono distinguere due casi: il re sta muovendo in una posizione che risulta coperta dal re avversario, oppure la torre muove in una cella oltrepassando celle occupate dall'avversario. Si tenga presente che nel caso particolare del finale di torre contro re il secondo caso non può avvenire, si può quindi considerare che una mossa illegale può essere giocata solo dal re. Un ulteriore mossa particolare avviene quando la torre da scacco al re avversario, caso in cui la propria scacchiera viene aggiornata imponendo che il re avversario sia nella stessa riga o nella stessa colonna della torre. Le mosse per le quali l'arbitro non offre suggerimenti permettono di aggiornare la propria scacchiera considerando libere le celle intorno al proprio re e quelle poste nella riga e nella colonna della torre. La figura 4.28 mostrano un esempio di quanto detto a partire da un posizione iniziale che vede il re in c5 e la torre in f3.

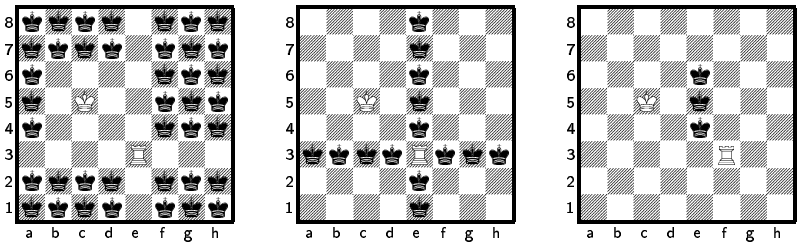


Figura 4.28: Risposte possibili dell'arbitro

La funzione $playPlayerMove(int\ color, move\ m)$ svolge gli aggiornamenti relativi ai casi descritti. Essa accetta in input il colore del giocatore in modo da utilizzare esclusivamente la sua scacchiera con la mossa m e la risposta data dall'arbitro $answer$.

Riporto lo pseudocodice relativo a $playPlayerMove$ nella tabella 4.3. La funzione richiama la corrispondente $GenerateMove$ per il giocatore, simile a quella vista nel paragrafo precedente e di conseguenza $GenerateSlidingMove$ e $GenerateNormalMove$ per aggiornare le posizioni del re avversario nella propria scacchiera di riferimento.

```
INPUT:  c player color,  m move to play,  answer referee's answer
OUTPUT: void, it modifies the c coloured player's reference board
   playPlayerMove(int c, move m, int answer) {
    int i, n, who, beforeColor, beforePiece;

    playerHistory[c][playerHistoryIndex[c]]=bo[c];
    playerHistoryIndex[c]++;
    remove the piece from square m.from and put it on m.to;

    switch(answer) {
     case NONE:
              n = number of opponent's moves according to player c
              for(i=0;i< n;++i)
                if the opponent's king is on square Moves[i].to
                  remove the king
              if(who & KING)
                  removeAttacking((int)m.to,opponent,c);
              break;
     case ILLEGAL:
              if the opponent has the last piece (the king) {
                  removeFarKings((int)m.to,c);
                 }
                 Undo the move
              break;
     case CHECK:
              if(who & KING)
                  find the rook which has checked the opponent's king

              n = number of opponent's moves according to player c
              for(i=0;i< n;++i) {
                  if the opponent's king is on square Moves[i].to
                     mark square Moves[i].to and remove the king
              }
              for(each square s) {
                  if the opponent's king is on square s
                     remove the king
                  if square s is marked {
                     remove the mark from s
                     put the opponent's king on s
                  }
              }
              break;
    }
  }
```

Tabella 4.3: La funzione *playPlayerMove*

La funzione complementare alla *playPlayerMove* è quella incaricata a ritrattare la mossa e tornerà utile durante l'algoritmo di ricerca. È chiamata *playerTakeBack*(*intc*) e, per ripristinare lo stato, sfrutta il vettore playerHistory. Il suo codice diviene così molto snello:

```
void playerTakeBack(int c) {
      playerHistoryIndex[c]–;
      bo[c]=playerHistory[c][playerHistoryIndex[c]];
 }
```

4.5.7 L'algoritmo di ricerca

La prima difficoltà nello scrivere l'algoritmo di ricerca è stato scegliere con quale tipo di visita analizzare l'albero delle mosse. L'aspetto principale risiede nel fatto che, in Kriegspiel, la mossa, oltre che essere una mossa normale di scacchi, viene descritta esaustivamente dalla risposta data dall'arbitro. Questo implica che una valutazione della mossa può essere fatta solo conoscendo tale suggerimento. Si presenta quindi il problema di come prevedere la risposta non potendo conoscerla a priori, salvo alcune rare eccezioni.

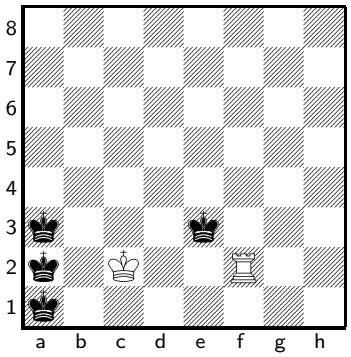


Figura 4.29: Esempio di metaposizione

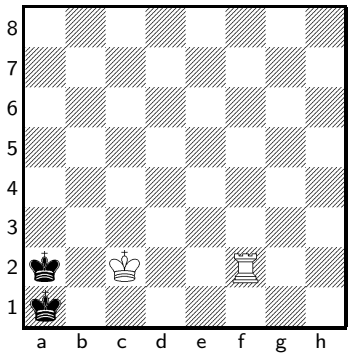


Figura 4.30: ♔b1 Illegal

Consideriamo ad esempio di essere nella situazione mostrata in figura 4.29. Il re si trova in c2 e la torre in f2, mentre l'informazione in possesso del bianco circa la posizione del re nero lo vede con probabilità di 1/4 in a1, a2, a3 o e3. Se il re muovesse in b1 ottenendo la risposta Illegale (fig. 4.30) la mossa si rivelerebbe buona, diminuendo l'incertezza circa la posizione del nero e ottenendo l'imprigionamento del re nero. Se invece muovendo il re in b1 la risposta dell'arbitro fosse nulla (fig. 4.31 e 4.32) si giungerebbe in uno stato potenzialmente pericoloso in quanto la torre rischierebbe di essere

mangiata e la partita finirebbe per patta. In conclusione, a partire dallo stato descritto in figura 4.29 la mossa ♔b1 è sconsigliabile.

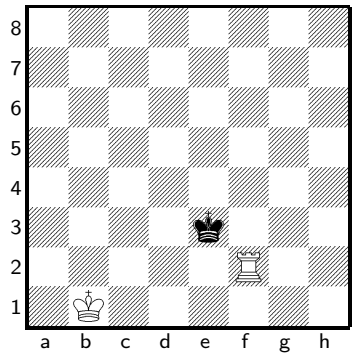


Figura 4.31: ♔b1

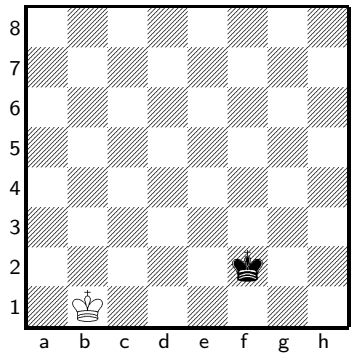


Figura 4.32: . . . ♔f2 Draw

Prima formulazione dell'algoritmo

In questa parte propongo una prima versione dell'algoritmo di ricerca, pubblicata recentemente in [3].

La soluzione che ho adottato per risolvere questo problema è stata quella di effettuare una prima valutazione estemporanea durante la generazione delle mosse pseudolegali considerando sia il caso di risposta dell'arbitro nulla sia il caso di scacco o illegale, e inserendo nel vettore di mosse possibili quella con valore minore. In altre parole il giocatore artificiale considera di ogni mossa il caso peggiore e mette a disposizione dell'algoritmo di ricerca solo quella.

Nell'esempio precedente avrebbe inserito la mossa ♔b1 con risposta nulla e avrebbe scartato ♔b1 con risposta Illegale, essendo

$$\text{val}(\text{♔b1 Empty}) < \text{val}(\text{♔b1 Illegal}).$$

Agendo in questo modo il costo computazionale diminuisce notevolmente rispetto ad un primo algoritmo che procede ad inserire nel vettore le mosse con entrambe le risposte, poiché dimezza il numero di elementi nell'array e formula una situazione più simile al(caso degli scacchi tradizionali. La funzione che implementa questa prima scrematura euristica sarà approfondita nel seguito.

La soluzione adottata a questo problema, consiste nel ridurre tutte le mosse di un pezzo avversario in una unica metamossa. Questa può essere considerata come l'accorpamento di tante mosse contemporanee quante sono le posizioni possibili. Ad esempio nel caso in questione si possono considerare

tanti re avversari presenti sulla scacchiera quante sono le posizioni possibili del re nero e calcolare la scacchiera risultante dal gioco contemporaneo di tutti questi re. Si noti che tale metodo delinea un accorgimento snello ed efficace per il calcolo delle mosse dell'avversario, a differenza degli algoritmi per gli scacchi classici che ad ogni chiamata ricorsiva decrementano la profondità di ricerca e analizzano l'azione dell'avversario, questo metodo permette di saltare tale passaggio esaminando ad ogni round solo le proprie mosse. Questo comporta quindi il dimezzamento della profondità di ricerca richiesto per ottenere risultati accettabili.

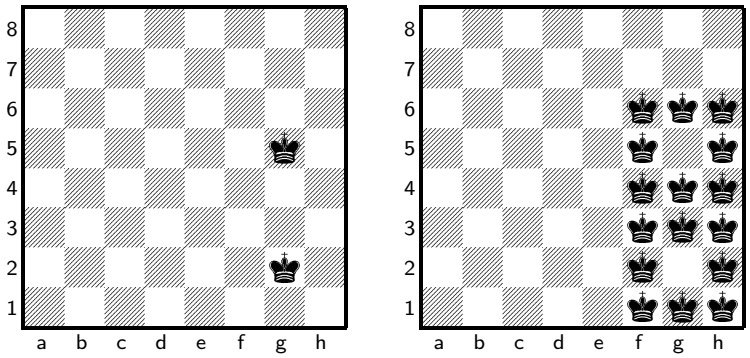


Figura 4.33: Esempio di una metamossa del Re

Le miniature di figura 4.33 descrivono lo stato raggiunto giocando la metamossa a partire da una scacchiera ove il re nero è posizionato con probabilità di 1/2 in g2 o g5.

L'algoritmo di ricerca procede generando le mosse da analizzare alla profondità desiderata, applicando così una prima valutazione delle mosse e la scelta della risposta dell'arbitro che deve essere considerata per ciascuna mossa; quindi se è giunto alla profondità di ricerca desiderata restituisce il valore massimo calcolato nella generazione, altrimenti gioca la mossa e, nel caso in cui a muovere non sia il re con risposta illegale dell'arbitro, chiama se stessa ricorsivamente decrementando la profondità; ritratta la mossa e somma al valore della mossa quello ritornato dalla chiamata ricorsiva, infine aggiorna il valore massimo per quella data profondità.

Riporto in tabella 4.4 lo pseudocodice che esegue l'algoritmo descritto, pubblicato in [3].

Come si evince dal codice e dalle considerazioni precedenti, il valore della mosse viene modificato durante la ricerca ed evolve sintetizzando un giudizio sulla strategia percorsa. Se non si operasse un tale aggiornamento una mossa potrebbe ottenere un buon risultato anche attraversando stati in cui si incorre nel rischio della cattura della torre e questo sarebbe riprovevole. La figura

```
First Search Algorithm (int depth) {

    generate the white's legal moves Γ;

    for each moves j ∈ Γ
    {
       if(rook plays the move j)
          j.value=Min(evaluate(j,check),evaluate(j,silent))
       if(king plays the move j)
          j.value=Min(evaluate(j,illegal),evaluate(j,silent))
    }

    for each moves j ∈ Γ
    {
          if (depth! = 1) {
              makemove(j);
              generate the opponent's metamove;
              if(!CheckHash(depth−1,&value))
                 j.value += Search(depth-1);
              unmakemove();
          }
          if (j.value > max)
              max=j.value;
    }

    RecordHash(depth,max);

    return max;
}
```

Tabella 4.4: Il primo algoritmo di ricerca

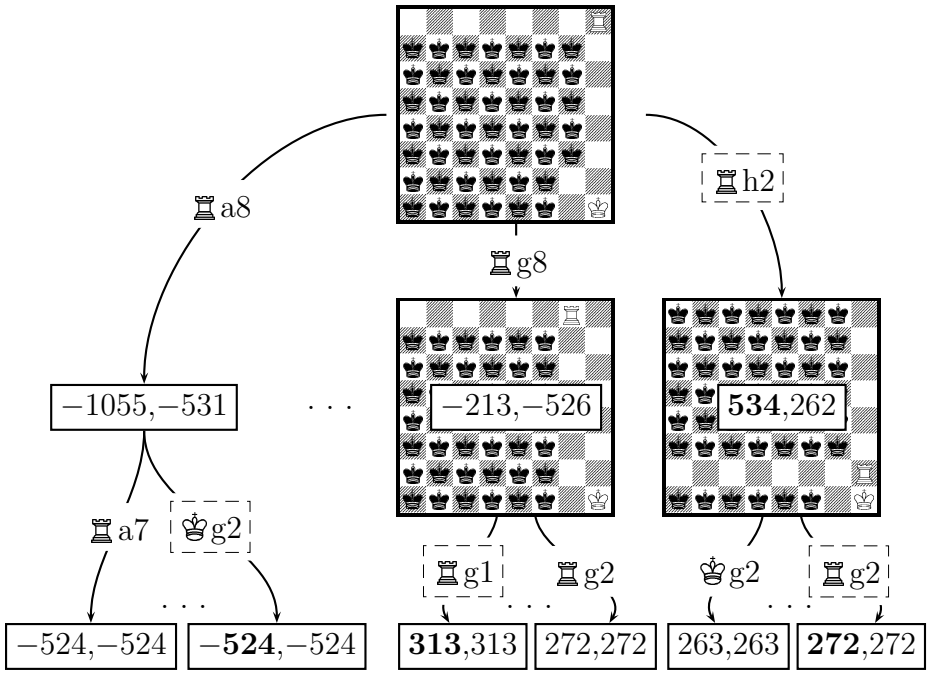


Figura 4.34: Un albero di ricerca di profondità 2

4.34 descrive graficamente un esempio di visita attraverso il grafo dell'albero di ricerca, ove alla destra di ogni nodo viene riportata la prima valutazione fatta e sulla sinistra il valore complessivo padre-figlio aggiornato nella ricerca.

Se non avessimo aggiunto la valutazione statica del nodo a quella ricorsiva, alla prima profondità le mosse avrebbero rispettivamente ottenuto −524, 313 e 272, quindi la seconda mossa (che ha una pessima valutazione statica) sarebbe stata scelta dall'algoritmo di ricerca, mentre la terza mossa (che ha una valutazione statica migliore) sarebbe stata scartata.

Formulazione definitiva

La stesura dell'algoritmo proposta in questa sezione è la versione affinata del precedente, base da cui si è partiti per la formulazione di una procedura di ricerca generalizzata anche in finali di partita differenti, come sarà approfondito nel capitolo 5.

A differenza della precedente versione, la previsione euristica sulla risposta dell'arbitro non viene effettuata durante la generazione per ogni tipo di mossa. Inizialmente si controlla se la posizione raggiunta è già stata analizzata precedentemente durante la partita, nel qual caso l'albero di ricerca viene potato e l'algoritmo termina restituendo il valore e la mossa già calcolati;

```
Search Algorithm (int c, int depth, int *from, int *to) {
    int t, f, val, who, max;
    BOOL badHash;

    max=−1000000;
    if(depth==0)
        return  Evaluate(c);
    if(CheckHash(depth,c,&val,&f,&t,&who))
        if(f, t, and who don't denote a recently played move) {
          *from=f; *to=t;
          return val;
        }

    generate the c coloured player's legal moves Γ;
    for each moves j ∈ Γ   {
        if(j.to, j.from, j.who denote a recently played move) {
            badHash=TRUE; continue;
        }

        \*****first case*********\
        playPlayerMove(c, j, NONE);
        if(playerMaybeStale(c)) {
            playerTakeBAck(c); continue;
        }
        updateAfter(c,NONE);
        silent= Evaluate(c);
        playerTakeBack(c);

        \*****second case*********\
        if(!(j.who&KING)) {
          playPlayerMove(c, j, CHECK);
          updateAfter(c,NONE);
          checked= Evaluate(c);

          if(silent<checked) {
              restore the reference board to the first case
              val=silent;
          } else val=checked;
        } else
              val= Evaluate(c);
```

```
        if(val==10000) {
            *from=j.from; *to=j.to;
            playerTakeBack(c);
            return val;
        }
        if(val==−1000) {
            playerTakeBack(c); continue;
        }

        val+= Search Algorithm(c, depth−1, &f, &t);

        if(val>max) {
            *from=j.from; *to=j.to; max=val;
        }
        playerTakeBack(c);

        if(!badHash && max!=−1000000)
          RecordHash(depth,c,max, better from, better to, better who);

        return max;
    }
```

Tabella 4.5: L'algoritmo di ricerca

in particolare però si evita di restituire una mossa che rientra nell'insieme di mosse giocate recentemente, giocando le quali cioè, si corre il rischio di ripetere le stesse azioni in un ciclo chiuso. Per l'eventuale approfondimento della tabella di trasposizione rimando alle sezioni 4.5.10 e 4.5.11.

Se non si è trovata un'occorrenza nella tabella hash di trasposizione, si procede alla generazione di tutte le mosse del giocatore, chiamando le funzioni descritte in sezione 4.5.5. Per ogni mossa, se si tratta di una mossa la cui esecuzione comporta il rischio di ciclo, la si scarta e si passa direttamente a quella successiva. Altrimenti si procede all'implementazione della scrematura euristica sulle risposte dell'arbitro. Dapprima si considera che la risposta sia silente (giocando $playPlayerMove(c, mossa, NONE)$), si esegue un controllo sulla scacchiera di riferimento ottenuta per accertarsi che essa non rappresenti un potenziale stato di stallo ($playerMaybeStale(c)$), quindi si genera la metamossa avversaria, considerando che anche questa riceva risposta silente ($updateAfter(c, NONE)$). Viene salvato il risultato ritornato dalla funzione di valutazione nella situazione ottenuta. Questa procedura è indicata nelle

pseudocodice come "first case". Ad essa segue una seconda fase che considera l'eventualità di una mossa di scacco. Per il re viene semplicemente chiamata la funzione di valutazione, in quanto il re non può dare scacco, per la torre si considera una risposta di scacco ($playPlayerMove(c, mossa, CHECK)$), ma una risposta silente nella generazione della metamossa dell'avversario ($updateAfter(c, NONE)$).

Come nella prima formulazione dell'algoritmo, per ogni mossa x del giocatore l'intenzione è quella di ipotizzare l'eventuale risposta dell'arbitro peggiore tra quella silente x_s e quella di scacco x_c, e quindi scegliere la mossa migliore, ovvero scegliere tra i casi $x_s \neq x = x_c$ o $x_c \neq x = x_s$. Per questo si confrontano i risultati ottenuti nei due casi dalla funzione di valutazione: se il valore della mossa con risposta silente è strettamente minore (quindi strettamente peggiore) del valore con risposta di scacco ($x_s < x_c$), si considera che $x = x_s$, altrimenti si considera verificata l'uguaglianza $x = x_c$. Questo significa che, nel caso in cui $x_s = x_c$ la scelta che si compie è quella di considerare vera $x = x_c$ e quindi supporre di aver dato scacco. In altri termini, questo significa considerare peggiore la scacchiera di riferimento del giocatore nel caso di scacco rispetto al caso con risposta nulla.

Se la valutazione dello stato ha un voto pari a 10000, allora la mossa valutata da scacco; l'albero di ricerca è quindi drasticamente potato e l'algoritmo termina. Se il valore è pari a -1000 invece, la torre si è messa sotto attacco e la mossa viene quindi scartata senza esaminarne i sottoalberi. Questi valori sono ritornati dalla funzione di valutazione che sarà approfondita nella sezione 4.5.9, a cui rimando ulteriori approfondimenti.

L'algoritmo procede quindi ricorsivamente fino alla profondità 0.

Infine viene aggiornata la variabile max che mantiene il valore della mossa migliore trovata, e, nel caso in cui si sia effettivamente trovata una mossa migliore e non se ne sia scartata nessuna per evitare cicli, si provvede ad inserire l'occorrenza della mossa e il suo valore nella tabella di trasposizione, ricorrendo alla funzione $RecordHash$.

4.5.8 Evitare lo stallo

Per evitare di incorrere in stati in cui la partita finirebbe per stallo, durante l'algoritmo di ricerca viene chiamata la funzione $playerMaybeStale()$ riportata in tabella 4.6. Questa è una semplice funzione che marca le case occupate dal giocatore di colore c e tutte quelle sotto attacco dai pezzi di colore c sulla scacchiera di riferimento dello stesso giocatore. Quindi procede con la generazione delle mosse dell'avversario, per ogni re avversario presente sulla scacchiera di riferimento di colore c. Se viene trovata anche solo una mossa in cui il re avversario muove su una casa non marcata, allora l'eventualità di

INPUT: **c** player color
OUTPUT: boolean that is true when the c reference board is in a potential state of stalemate

```
BOOL playerMaybeStale(int c) {
    generate the c coloured player's legal moves Γ;
    for each moves j ∈ Γ  {
        mark j.from
        mark j.to
    }

    for each opponent king on the c player's reference board  {
        generate the opponent king's legal moves Γ;
        for each moves i ∈ Γ  {
            if(i.to is not marked)
                return TRUE;
        }
    }
    return FALSE;
}
```

Tabella 4.6: L'algoritmo che evita lo stallo

uno stallo è impossibile, altrimenti lo stallo ha una certa probabilità. Considerando il caso peggiore, se la funzione $playerMaybeStale()$ ritorna 'vero' si considera la mossa pericolosa e quindi la si scarta.

4.5.9 La funzione di valutazione

La funzione di valutazione è il cuore centrale del giocatore artificiale, la bontà delle scelte infatti dipende in gran parte dalla giusta valutazione delle mosse.

La funzione di valutazione della prima stesura dell'algoritmo di ricerca è stata la base che ha portato, con alcuni miglioramenti, alla funzione d'utilità definitiva. Per questo motivo riporto dapprima la funzione di valutazione presentata in [3] e in seguito le differenze sostanziali con la versione finale.

Prima formulazione

Ho utilizzato una funzione lineare pesata come la seguente:

$$Evaluate(S) = c_1 f_1(S) + c_2 f_2(S) + ... + c_5 f_5(S) \tag{4.2}$$

ove $c_1, c_2, .., c_5$ sono cinque costanti e $f_1(S), .., f_5(S)$ le funzioni che compongono l'euristica.

Per prima cosa è necessario assicurarsi che la mossa compiuta non metta la torre in una situazione in cui rischia di essere catturata. La prima funzione booleana $f_1(S)$ calcola quindi tale rischio utilizzando la variabile *Attack* descritta nei paragrafi precedenti che viene impostata ad 1 qualora la torre sia sotto attacco e rischi quindi la cattura.

Assicurando che lo stato a cui porta la mossa non mette a rischio la torre, è preferibile avvicinare i due re, diminuendo l'opportunità per l'avversario di andare in stallo e, soprattutto, per permettere al re bianco un'esplorazione della scacchiera; per questo la seconda funzione, $f_2(S)$, calcola la distanza tra il re bianco e la posizione del re nero più lontana tra quelle che si giudicano possibili. Questo è implementato nella funzione $EvalKingDistance$ che somma le distanze per riga e per colonna. Come mostrato in figura 4.35 la distanza tra il re bianco e quello nero è data come la somma delle colonne e delle righe che li separano, mentre la distanza tra il re bianco e l'insieme di posizioni del re nero è calcolata come la distanza massima tra tutti i singoli re neri. In figura 4.35 mostro un esempio in cui questa distanza è 10.

Considerando che sia il turno del giocatore bianco, il re di colore nero si trova sicuramente in uno dei quattro quadrilateri ideali definiti dalla posizione della torre bianca. L'obiettivo, per il giocatore bianco, è allora quello di diminuire l'area del quadrilatero all'interno del quale è presente il re nero, ma anche in questo caso si presenta lo stato d'incertezza dato dalla mancanza

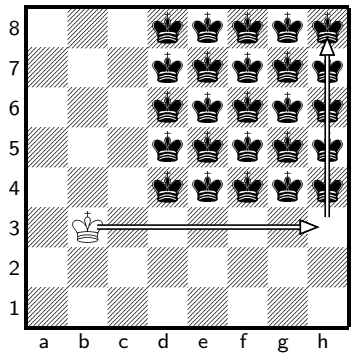


Figura 4.35: Diagramma della funzione *EvalKingDistance*

di conoscenza circa la posizione dell'avversario. La terza funzione, $f_3(S)$, si occupa di questo aspetto tentando, da un lato, di diminuire l'area delle quattro aree, dall'altro di diminuire l'incertezza su quale sia il quadrilatero in cui è effettivamente presente il re nero. Definiamo la funzione $EvalArea$ come

$$f_3(S) = EvalArea(S) = c \cdot (a_1 + a_2 + a_3 + a_4) \tag{4.3}$$

ove $c \in \{1, 2, 3, 4\}$ è il valore che tiene traccia del numero di quadrilateri in cui è possibile che si trovi il re nero, e $a_i (i = 1, ..4)$ indicano il numero di posizioni possibili per il re avversario in ciascun quadrilatero. In figura 4.36 viene proposto il calcolo delle aree nel caso pessimo in cui si ha l'incertezza massima sul re nero ove è evidente il peso dato dal valore c, in questo caso la funzione restituisce 180.

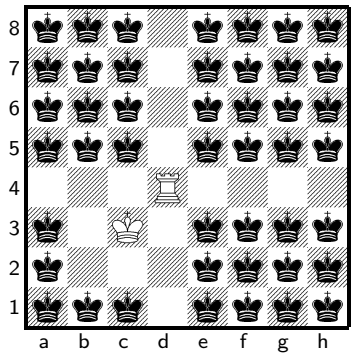


Figura 4.36: Metaposizione dove *EvalArea*= 180

La quarta funzione $f_4(S)$ è booleana e calcola la distanza tra il re bianco e la torre bianca, se la torre bianca è in una casa adiacente al re bianco il valore della mossa viene aumentato di 1.

La quinta funzione $f_5(S)$ valuta migliori le mosse che spingono il re nero ai bordi della scacchiera, questa fa uso di un vettore di 64 elementi indicanti le celle alle quali è associato il valore numerico restituito (4.37).

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & -2 & -4 & -4 & -2 & 0 & 0 \\ 0 & 0 & -4 & -4 & -4 & -4 & 0 & 0 \\ 0 & 0 & -4 & -4 & -4 & -4 & 0 & 0 \\ 0 & 0 & -2 & -4 & -4 & -2 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Figura 4.37: La matrice numerica usata dalla funzione $f_5(S)$

È utile notare che la funzione $f_3(S)$ calcola un valore positivo, ma per poter valutare la mossa migliore è necessario minimizzare questo valore. Questo è fatto adottando il peso della funzione negativo.

La stessa considerazione sulle altre funzioni porta alla formulazione della funzione di valutazione seguente:

$$Evaluate(S) = -420 + 840 \cdot f_1(S) - f_2(S) - f_3(S) + f_4(S) + f_5(S)$$

dove $c_1 = 840$ è un peso che dà a $f_1(S)$ massima priorità.

Formulazione definitiva

La funzione d'utilità presentata nella sezione precedente è stata la base per l'implementazione della funzione $Evaluate(c)$, riportata in tabella 4.7.

La funzione lineare pesata è stata modifica per poter permettere una potatura efficace dell'albero di gioco. Innanzitutto la funzione f_1 è divenuta una sentinella dello stato di rischio, che, se verificato, ritorna un valore fisso che ho scelto pari a -1000. Questa può essere considerata un'etichetta per cui la mossa che porta allo stato con questo valore viene automaticamente scartata. Una prima obbiezione a questa scelta è che alcune mosse che non incorrono in uno stato dove la torre è attaccata, possono comunque avere una valutazione pari a -1000. In tal caso però, avendo scelto un valore molto basso, la mossa scartata avrebbe comunque portato ad uno stato poco buono. La potatura euristica che ne consegue risulta pertanto efficace.

Il vettore $boardAnalysis[]$ corrisponde esattamente alla matrice numerica usata precedentemente nella funzione $f_5(S)$.

INPUT: **c** player color
OUTPUT: integer that represents the value of the metaposition of c's reference board

```
int Evaluate(int c) {
    int res, resAnalysis;

    res=0;
    for(i=0; i<128; ++i) {
        if (c coloured player is attacked on i)
            return −1000;

        if (the opponent king is on i)
            resAnalysis=boardAnalysis[i];

        if((i+1)&0 × 08) i+=8;
    }

    if (playerOneToMate(c)!=−1)
        return 10000;

    res− =EvalKingDistance(king,c);
    res− =EvalArea(rook,king,c);

    if(endingRook(c,rook,king))
        return -1000;

    if (c coloured rook is adjacent to c coloured king)
        res+=10;

    return res;
}
```

Tabella 4.7: La funzione di valutazione

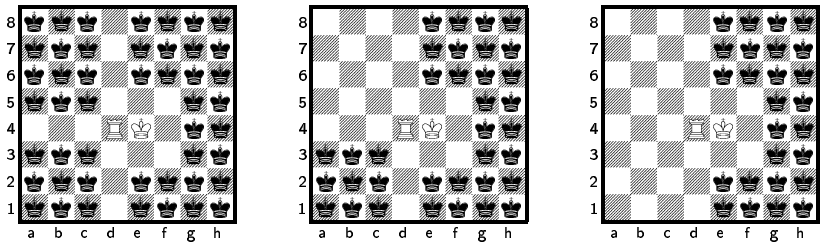
INPUT: **c** player color
OUTPUT: integer that represents the square where c player has to move to checkmate.

```
int playerOneToMate(int c) {
    int count;

    count = number of opponent's king on c reference board;
    if(count>2)
        return -1;
    generate the c coloured player's legal moves Γ;
    for each moves j ∈ Γ  {
        if(j.to!=first or last row || j.to!=first or last column)
            continue;

        if(!(j.who&KING))
            play the move j;
        else
            continue;

        for each opponent's king k on c reference board
            if k is not attacked by c player
                continue;

        updateAfter(c,NONE);
        if (c player risks to be captured)
            continue;
        count = number of opponent's king on c reference board;
        if(count==0)
            return j.to;
    }
    return -1;
}
```

Tabella 4.8: La funzione *playerOneToMate()*

La funzione $playerOneToMate(intc)$, il cui codice è presentato in tabella 4.8 ritorna il valore corrispondente alla cella in cui muovere per dare sicuramente scacco matto all'avversario, in caso contrario ritorna -1. L'implementazione è piuttosto generica e può quindi funzionare anche in casi differenti dal finale che stiamo trattando. In particolare procede seguendo diversi passi. Dapprima, conta il numero di re avversari presenti sulla scacchiera di riferimento del giocatore di colore c; se questi sono più di 2, allora non è possibile dare scacco matto. Prosegue quindi generando tutte le mosse possibili per il giocatore c. Se questi muove in una casa che non appartiene né alla prima o ultima riga, né alla prima o ultima colonna, passa a considerare le mosse successive. Se a muovere non è il re (che non può dare scacco), gioca la mossa e controlla che ogni re avversario sulla scacchiera di riferimento sia sotto attacco. Se ciò avviene, gioca la metamossa dell'avversario con la funzione $updateAfter(c, NONE)$ quindi controlla che la mossa eseguita non abbia messo il pezzo che ha mosso (in questo caso la torre) sotto attacco. In caso negativo passa a considerare le mosse successive, altrimenti conta il numero di re avversari sulla scacchiera di riferimento: se questo è pari a zero allora la metamossa era nulla o, in altri termini, il re avversario non aveva alcuna mossa legale, ossia si è dato scacco matto.

La funzione $EvalKingDistance(king, c)$ corrisponde alla $f_2(S)$, mentre la funzione $f_3(S) = EvalArea(S)$ ha subito una leggera modifica. Si considerino le miniature in figura 4.38. La prima di sinistra viene valutata con $f_3(S) = 4 \cdot (12 + 14 + 9 + 10) = 180$, ma in realtà, grazie alla posizione del re, le aree descritte dalla riga e dalla colonna della torre sono 3 e non 4. La funzione $f_3(S)$ descritta in (4.3) è stata pertanto modificata come segue.



$f_3(S) = 180$ $f_3'(S) = 135$ | $f_3(S) = 99$ $f_3'(S) = 66$ | $f_3(S) = 38$ $f_3'(S) = 38$

Figura 4.38: Confronto tra le due versioni della funzione *EvalArea*

Indichiamo con $f_3'(S)$ la nuova funzione $EvalArea()$. Se il re è alla destra della torre e il re avversario è presente nelle aree della scacchiera con colonne maggiori della colonna della torre (come nelle miniature di figura 4.38), e se il re nero è presente anche in un quadrilatero delimitato dalla torre alla sua

sinistra, allora il valore $c \in \{1, 2, 3, 4\}$ della (4.3) viene decrementato di 1; altrimenti la funzione resta invariata.

Ad esempio nella miniatura di sinistra della figura 4.38 si ottiene $f'_3(S) = 3 \cdot (12 + 14 + 9 + 10) = 135$. La miniatura centrale, veniva valutata con $f_3(S) = 99$, mentre ora con $f'_3(S) = 66$. Infine, la miniatura di destra viene valutata allo stesso modo con entrambe le funzioni, ossia $f_3(S) = 38 = f'_3(S)$; questo perché il re nero non è presente in almeno un quadrilatero alla sinistra della torre.

È chiaro che tale accorgimento è stato eseguito per le altre tre posizioni analoghe per simmetria e riflessione a quella esaminata.

Con l'uso della $f'_3(S)$ al posto della $f_3(S)$ i risultati ottenuti sono stati migliori, a tal proposito rimando alla sezione 4.6 relativa all'esame dei test eseguiti.

La funzione booleana $endingRook(c, rook, king)$ non era presente nella prima formulazione ed è stata aggiunta. Questa considera il rettangolo formato dal re e dalla torre, ovvero avente coordinate dei due angoli opposti dati rispettivamente dalla colonna e della riga minori tra quelle del re e quelle della torre per l'angolo inferiore sinistro, e dalla colonna e dalla riga maggiori per l'angolo superiore destro. Se un re avversario è presente all'interno di tale rettangolo, allora si pota la mossa dall'albero di gioco.

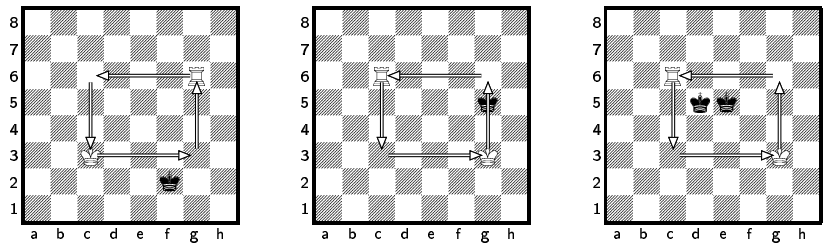


Figura 4.39: Esempio di quadranti della funzione *endingRook*

In figura 4.39 sono riportate alcune miniature d'esempio che descrivono il giudizio restituito dalla funzione $endingRook()$.

Infine, la funzione booleana $f_4(S)$, che calcola se la torre bianca è in una casa adiacente al re bianco, resta invariata, ma il valore della mossa nel caso in cui sia verificata viene aumentato di 10 e non più di 1.

4.5.10 Le chiavi di Zobrist

La scacchiera di riferimento del giocatore è descritta esaustivamente dalle possibili posizioni dell'avversario che il giocatore mantiene oltre a quelle dei

suoi pezzi in gioco. Poiché durante l'algoritmo di ricerca si incorre in stati della scacchiera già precedentemente analizzati anche migliaia di volte in un secondo, ripetere una seconda volta calcoli già compiuti si traduce in un notevole calo di prestazioni. D'altra parte mantenere tutte le posizioni necessarie per poter confrontare i diversi stati della scacchiera attraversati, essendo questi nell'ordine delle migliaia, richiederebbe un'enorme allocazione in memoria. Questo problema viene risolto creando un valore unico in grado di identificare lo stato della scacchiera. Questo valore è tipicamente implementato a 64-bit nel gioco standard degli scacchi e può esserlo anche nel finale di torre in Kriegspiel, anche se non è detto che basti per il gioco completo del Kriegspiel, in quanto nella scacchiera di riferimento si terrebbe traccia non solo delle possibili posizioni del re avversario, che variano da 1 a 60, ma anche di quelle di ciascun pezzo, arrivando ad un massimo di 768 (16*48)[5] posizioni, alle quali andrebbero aggiunte le sicure posizioni dei propri pezzi. Nel finale di torre, 64-bit bastano ad enumerare ogni stato, la probabilità di avere delle collisioni è nulla e può quindi essere ignorata. Si ha infatti, dall'equazione (4.1), che $2^{52} - 1 < 2^{64}$. Ciò non è vero se ci si estende ad altri finali. È bene, in questo caso, analizzare la probabilità di collisione e, seppure molto bassa, è bene evitare che il giocatore artificiale fallisca nel caso in cui ne avvenga una.

Ho implementato il valore univoco adoperando la tecnica delle chiavi di Zobrist [41]. Queste sono vettori tridimensionali di valori casuali a 64-bit Definisco quindi le chiavi di Zobrist come vettori tridimensionali

```
long long zobrist[6][2][128];
```

ove 6 indica il numero di pezzi {PAWN, KNIGHT, BISHOP, ROOK, QUEEN, KING, EMPTY}, 2 il numero di giocatori {WHITE, BLACK}, 128 il numero di celle. Quando il programma viene lanciato esso provvede a riempire l'array con numeri casuali. Per creare il valore univoco per uno stato si procede impostando a zero la chiave per quello stato ed effettuando l'operazione di XOR per ogni posizione possibile dei pezzi dell'avversario e per ogni posizione dei propri pezzi.

La funzione $fillZobrist(intcolor)$ si occupa di impostare la chiave per uno stato nel caso del finale di torre e di ritornarla al chiamante:

[5] le case disponibili per un pezzo avversario sono limitate superiormente dal numero di case non occupate dal proprio giocatore, ossia 64-16=48

```
long long fillZobrist(int c) {
    int i;
    long long key;
    key = 0;
    for(each squares on the reference board with colour c) {
        if(blak king is on square i)
            keyˆ=zobrist[KING][BLACK][i];
        if(blak rook is on square i)
            keyˆ=zobrist[ROOK][BLACK][i];
        if(white king is on square i)
            keyˆ=zobrist[KING][WHITE][i];
        if(white rook is on square i)
            keyˆ=zobrist[ROOK][WHITE][i];
    }
    return key;
}
```

La tecnica delle chiavi di Zobrist viene usata per creare valori che fungano da chiavi per una tabella hash. Una prima proprietà è che i valori generati non sono correlati allo stato cui si riferiscono, nel senso che la mossa di un unico pezzo produce una chiave che è completamente differente da quella dello stato precedente e questo è un aspetto molto buono nell'uso di funzioni hash. Una seconda proprietà, non meno importante, è la possibilità di evolvere le chiavi in modo incrementale. Supponiamo ad esempio di partire da uno stato in cui la torre bianca è presente in a2, ove cioè nella chiave che descrive tale situazione è inserito con l'operatore XOR il valore zobrist[ROOK][WHITE][A2]. Se ripetiamo l'operazione di XOR del valore zobrist[ROOK][WHITE][A2], per le proprietà dell'operatore stesso, la presenza della torre bianca in a2 viene eliminata dalla chiave e si ottiene la stessa chiave che si sarebbe ottenuta calcolando daccapo l'intero valore. Nel programma le chiavi di Zobrist sono state utilizzate per implementare una tabella di trasposizione. Questa è una grande tabella hash che permette di tenere traccia degli stati analizzati durante l'algoritmo di ricerca, qualora infatti si incontri ad una data profondità uno stato presente nella tabella di trasposizione si risparmia un ulteriore ricerca che a tutti gli effetti era già stata compiuta.

4.5.11 La tabella di trasposizione

La tabella di trasposizione principale è un vettore di elementi HASH definiti con la seguente struttura:

```
typedef struct {
    long long key;
    int depth;
    int val;
} HASH;
```

Ciascuno di questi elementi tiene traccia del valore della mossa e della profondità a cui è stata valutata, oltre che dello stato tramite la chiave di Zobrist. Il vettore di elementi HASH è indicizzato tramite le chiavi degli stati, applicando ad esse l'operatore di modulo per il numero di elementi della tabella. Il vettore tst (*Transposition Table*) viene definito come

```
HASH tst[2*MAX_HASH];
```

ed il metodo di accesso ai valori è

```
tst[(int) (bo[color].key % MAX_HASH)+MAX_HASH];
```

ove MAX_HASH è una costante, posta a 1000, e bo[color].key indica la chiave d'accesso della scacchiera. In questo caso la tabella di trasposizione ha quindi 2000 elementi. Il motivo per cui viene messo in atto questo meccanismo è dovuto al fatto che la chiave può essere un numero negativo, mentre è necessario avere un indice positivo.

Poiché, in generale, è possibile che molti stati vengano codificati con lo stesso indice, gli elementi della tabella devono contenere un valore di validazione, per questo obiettivo è stata scelta la chiave di Zobrist dello stato che si presta perfettamente allo scopo.

Il motivo per cui è stato inserito nella struttura il valore della profondità è spiegato dal fatto che durante la ricerca si può evitare l'analisi di uno stato che già era stato esaminato ad una profondità maggiore, viceversa se ci si appresta a visitare uno stato ad una profondità maggiore di quella con cui lo si era precedentemente è bene continuare.

L'algoritmo di ricerca fa quindi uso delle due funzioni $RecordHash$ e $CheckHash$; la prima provvede ad inserire nella tabella di trasposizione la mossa una volta valutata ed il corrispondente valore, la seconda viene invece utilizzata per evitare la chiamata ricorsiva qualora si presenti uno stato già valutato, caso in cui restituisce il valore booleano $TRUE$.

Il codice dell'algoritmo di ricerca esegue la chiamata alla funzione $CheckHash$ come segue:

```
CheckHash(depth,color,&val,&from,&to,&who)
```

dove *depth* indica la profondità alla quale si visita la mossa, *color* il colore del giocatore che sta eseguendo la ricerca, *val* il valore ritornato dalla funzione della mossa precedentemente valutata, questo nel caso in cui l'esito sia $TRUE$. Vengono inoltre restituiti i valori della mossa giocata, indicati da $*f$ (from), $*t$ (to) e $*w$ (who).

Alla fine della ricerca la mossa valutata viene inserita nella tabella di trasposizione tramite la chiamata a $RecordHash$.

Riporto di seguito lo pseudocodice relativo alle due funzioni descritte.

```
BOOL CheckHash(int depth, int c, int *p, int *f, int *t, int *w) {
 HASH *hashpt=&tst[(int)(bo[c].key%MAX_HASH)+MAX_HASH];
      if(hashpt− >key == bo[c].key) {
        if(hashpt− >depth==depth) {
              *p=hashpt− >val;
              *f=hashpt− >f;
              *t=hashpt− >t;
              *w=hashpt− >who;
              return TRUE;
        }
      }
      *p=0;
      return FALSE;
 }


void RecordHash (int depth, int c, int val, int f, int t, int who) {
 HASH *hashpt=&tst[(int)(bo[c].key%MAX_HASH)+MAX_HASH];
      if(hashpt− >key == bo[c].key) {
            if(hashpt− >depth<depth) {
                  hashpt− >depth=depth;
                  hashpt− >val=val;
            }
      } else {
            hashpt− >key=bo[c].key;
            hashpt− >depth=depth;
            hashpt− >f=f;
            hashpt− >t=t;
            hashpt− >who=who;
      }
 }
```

È opportuno ora considerare i miglioramenti delle prestazioni per quanto riguarda la velocità di calcolo con l'aggiunta dell'ausilio delle chiavi di Zobrist. Per valutare questo miglioramento ho fatto riferimento al calcolo della mossa migliore per il bianco a partire da quattro stati differenti con ricerca ad una profondità pari a 3.

Nelle tabelle 4.9 e 4.10 ho riportato i valori utili al confronto, nella descrizione delle scacchiere di partenza ho indicato con K (maiuscola) il re bianco mentre con k (minuscola) il re nero, al solito R indicherà la torre bianca.

Come si può constatare la velocità di esecuzione del programma quasi si dimezza, passando, ad esempio se consideriamo il primo caso, da 611 ms a 391 ms. Questo avviene per una radicale ed efficace potatura dell'albero di ricerca che scarta 1800 nodi circa partendo dal primo stato e ne scarta 3483 partendo dall'ultimo.

posizione di partenza	profondità	nodi esaminati	Tempo(ms)
Ka1; Ra2; kb3	3	4762	611
Ka1; Rh8; kf3	3	5912	751
Ka1; Rd6; kg3	3	5910	761
Kd2; Rf4; kh1	3	10754	1382

Tabella 4.9: Risultati senza le chiavi di Zobrist

posizione di partenza	profondità	nodi esaminati	Tempo(ms)
Ka1; Ra2; kb3	3	2959	391
Ka1; Rh8; kf3	3	2960	390
Ka1; Rd6; kg3	3	3155	420
Kd2; Rf4; kh1	3	7271	951

Tabella 4.10: Risultati con le chiavi di Zobrist

4.6 Test

4.6.1 Esempi di partita

Affronto in questa sezione l'analisi di alcuni finali la cui soluzione è stata mostrata in [9]. Il confronto tra la strategia del giocatore artificiale e quella ottima calcolata a priori, risulta essere di grande aiuto nel dare un giudizio

corretto al programma. La miniatura mostrata in figura 4.40 descrive una posizione per cui le due strategie coincidono.

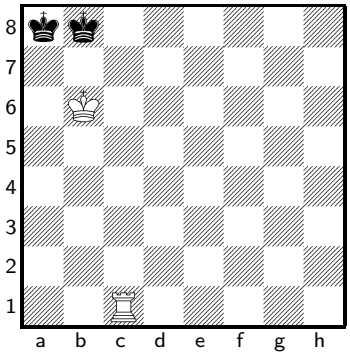


Figura 4.40: Primo esempio

Considerando la scacchiera di figura 4.40 e supponendo che la configurazione iniziale veda il re nero in b8, il programma che utilizza l'algoritmo di ricerca gioca come segue:
1. ♔c7 I, ♔a6; . . . ♔a8
2. ♖c8 #.

Supponendo che la posizione iniziale veda il re nero in a8:
1. ♔c7; . . . ♔a7
2. ♖a1 #

Considerando la scacchiera di figura 4.40 e supponendo che la configurazione iniziale veda il re nero in b8, il programma che utilizza l'algoritmo tratto dalla procedura di Boyce gioca come segue:
1. ♖c5; . . . a8 2. ♔c7; . . . ♔a7
3. ♖a5 #.

Supponendo che la posizione iniziale veda il re nero in a8:
1. ♖c5; . . . ♔b8
2. ♔c7 I, a6; . . . ♔a8
3. ♖c8 #

La soluzione proposta in [9] per la miniatura in figura 4.41 suggerisce che il Bianco giochi 1. ♔c7.

Se l'arbitro accetta, allora il re nero è in a8 e segue . . . ♔a7 2. ♖d6 ♔a8 3. ♖a6 #.

Se l'arbitro risponde "mossa illegale" allora il re nero è in c8 o in b8. Si gioca quindi 1. ♖c7 e:

nel caso di arbitro silente il re nero si trova in a8 ed il matto è banale;

nel caso l'arbitro annunci "scacco" si prosegue con 2. ♔d7: se la risposta è "illegale" il Nero ha mosso in d8. Il Bianco allora vince con 2. ♖c1 ♔e8 3. ♖f1 ♔d8 4. ♖f8 #.

se la risposta è silente il Nero ha giocato 1.... ♔b8 e si prosegue con 2. ♔d7 ♔a8 3. ♔c6 ♔b8 4. ♔b6 ♔a8 5. ♖c8 #.

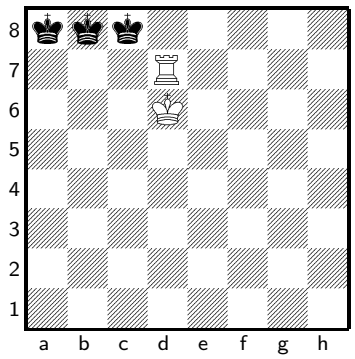


Figura 4.41: Secondo esempio

La strategia del giocatore artificiale basato sull'algoritmo di ricerca è meno efficace, seppur vincente.

Considerando la miniatura in figura 4.41 e supponendo che inizialmente il re nero sia in c8:

1. ♔c7 I, ♔c6; ... ♔b8
2. ♔c7 I, ♖d6; ... ♔c8
3. ♔d7 I, ♔b6; ... ♔b8
4. ♖d8 #

Considerando il re nero in b8:

1. ♔c7 I, ♔c6; ... ♔a8
2. ♔c7; ... ♔a7
3. ♖d5; ... ♔a6
4. ♔b6 I, ♔c6; ... ♔a7
5. ♖c5; ... ♔a6
6. ♔c7; ... ♔a7
7. ♖a5 #

Considerando il re nero in a8:

1. ♔c7; ... ♔a7
2. ♖d5; ... ♔a6
3. ♔b6 I, ♔c6; ... ♔a7
4. ♖c5; ... ♔a6
5. ♔c7; ... ♔a7
6. ♖a5 #

Anche l'algoritmo basato su Boyce prolunga la terminazione della partita. Partendo con il re nero in c8, si ha:
1. ♔c7, ♔e7; ... ♔b8
2. ♖d6; ... ♔b7
3. ♔d7; ... ♔b8
4. ♔c7 I, ♖e6; ... ♔b7
5. ♔c7 I, ♔c6 I, ♔d8; ... ♔a8
6. ♔c7; ... ♔a7
7. ♔b6 I, ♔c8; ... ♔a8
8. ♖a6 #

Partendo con il re nero in b8, si ha:
1. ♔c7 I, ♔e7; ... ♔a8
2. ♖d6; ... ♔a7
3. ♔d7; ... ♔b7
4. ♔c7 I, ♖e6; ... ♔b8
5. ♔c7 I, ♔c6; ... ♔c8
6. ♔c7 I, ♖e8 #

Partendo con il re nero in a8, si ha:
1. ♔c7; ... ♔a7
2. ♖d6; ... ♔a8
3. ♔b6; ... ♔b8
4. ♖d8 #

Si noti come l'algoritmo basato sulle procedure di Boyce giochi ottimamente il caso in cui il re nero è presente in a8. Nei restanti casi invece, prolunga la partita poiché mira ad ottenere una delle quattro posizioni iniziali della procedura (fig. 4.23).

Una volta giocato 1. ♔c7 e aver ricevuto risposta "illegale" la soluzione ottima prevede la mossa ♖c7, mentre il programma gioca ♔c6. Questo comportamento riallarga le possibilità per il nero e, in effetti, è la causa per cui il gioco si prolunga per 7 mosse. Tale scelta è imputabile al fatto che, nell'algoritmo di ricerca, si prediligono(le mosse del re, poiché al re si è dato il ruolo di "scopritore", considerando sempre silente la risposta data ai suoi spostamenti. Questo fa sì che 1.♖c7 riduca, nel caso peggiore (che corrisponde allo scacco), a 2 l'incognita dei re neri sulla scacchiera di riferimento, mentre 1.♔c6...2.♔c7, considerando la ricerca a profondità 2, garantisce un solo re avversario. Si noti inoltre che la mossa 1.♖c7 che dà "scacco" divide l'area dei re neri in due rettangoli peggiorando l'esito della funzione di valutazione, in particolare della funzione $EvalArea()$; la mossa 1.♔c6 lascia unica l'area che contiene i re avversari e ottiene un valore più alto.

L'euristica ideata, che come vedremo si è rivelata buona nella generalità dei casi, fallisce sotto gli aspetti presi in considerazione. Ciò sottolinea che la ricerca compiuta sull'albero di gioco non è ottima e la potatura è un compromesso che può creare perturbazioni sulle scelte del decisore.

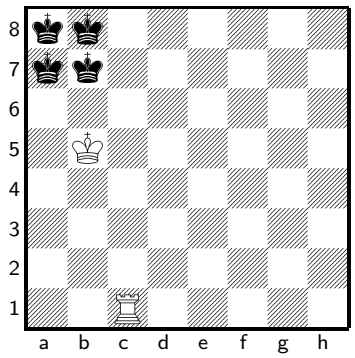


Figura 4.42: Terzo esempio

La soluzione proposta in [9] per la miniatura di figura 4.42, suggerisce al Bianco di tentare la mossa ♔b6. Se la mossa viene accettata ci si riconduce al caso di figura 4.40 e in questo la strategia del giocatore artificiale è impeccabile.

Se l'arbitro dice "mossa illegale", ancora una volta la strategia del programma differisce da quella ottima che prevede di tentare 1.♔c6. Se questa mossa viene accettata, il Re Nero è in a7; il Re Nero muove; 2.♖a1; se il Re Nero è in a6 allora è matto. Se 1...♔a8 allora 2...♔b8 3.♖a2 ♔c8 4.♖a8#. Se 1...♔b8 allora 2...♔c8 3.♖a8#.

Se 1.♔c6 non è accettata, allora il Re Nero è in b7. Si gioca 1.♖c5. Il re nero si muoverà in a7, a8 o b8, quindi si tenta 2.♔b6. Se l'arbitro accetta il Bianco matta in 2 come nell'esempio di figura 4.40; se l'arbitro dice "mossa illegale" il Nero ha giocato 1...♔a7 e si prosegue con 2.♔c6. Il Nero gioca 2...♔a6 o 2...♔a8 o 2...♔b8. Occorre tentare 3.♔c7, se l'arbitro accetta segue 3...♔a7 4.♖a5 #; se l'arbitro rifiuta il Nero ha mosso 2...♔b8 e segue 3.♔b6 e quindi 4.♖c8#.

Considerando la miniatura in figura 4.42 e supponendo che inizialmente il re nero sia in b7, il giocatore artificiale con algoritmo di ricerca, gioca:
1. ♔b6 I, ♖c4; ...♔a8
2. ♖c5; ...♔b7
3. ♖c4; ...♔b8
4. ♖c6; ...♔b7
5. ♔c5; ...♔a8
6. ♔b6; ...♔b8

7. ♔c7 I, ♔a6; . . . ♔a8
8. ♖c8#

Considerando il re nero in a7:

1. ♔b6 I, ♖c4; . . . ♔b7
2. ♖c5; . . . ♔b8
3. ♖c4; . . . ♔b7
4. ♖c6; . . . ♔a8
5. ♔c5; . . . ♔b8
6. ♔b6; . . . ♔a8
7. ♔c7; . . . ♔a7
8. ♔c8; . . . ♔a8
9. ♖a6#

Considerando il re nero in a8:

1. ♔b6; . . . ♔b8
2. ♔c7 I, ♔a6; . . . ♔a8
3. ♖c8#

Considerando il re nero in b8:

1. ♔b6; . . . ♔a8
2. ♔c7; . . . ♔a7
3. ♖a1#

Nei primi due casi, il punto in cui il giocatore artificiale fallisce risiede nel compiere alcune mosse inutili che ritardano la conclusione della partita. Poiché spesso durante la ricerca nell'albero di gioco, si incorre in situazioni con alta incertezza, per evitare cicli, nell'algoritmo di ricerca viene effettuato un confronto tra le mosse esaminate e le ultime mosse realmente giocate; nel caso in cui si verifichi un'identica mossa, a prescindere dalla scacchiera di gioco, essa viene scartata. Questa soluzione è piuttosto grossolana e provoca ripetizioni, soprattutto nella parte terminale dei finali di partita. Tali ripetizioni non sono gravi, poiché, dopo alcune giocate, mosse che, seppur buone, erano state precedentemente scartate, vengono riammesse. Su questo aspetto comunque è bene compiere approfondimenti e raffinamenti futuri.

Considerando la miniatura in figura 4.42 e supponendo che inizialmente il re nero sia in b7, il giocatore artificiale, basato sulla procedura di Boyce, gioca:

1. ♖c4; . . . ♔a8
2. ♔b6; . . . ♔b8
3. ♔c7 I, ♔a6; . . . ♔a8
4. ♖c8 #

Considerando il re nero in a7:

1. ♖c4; . . . ♔b7

2. ♔b6 I, ♔c6 I, ♔a5; ... ♔b8
3. ♔b6; ... ♔a8
4. ♔c7; ... ♔a7
5. ♖a4 #

Considerando il re nero in a8:

1. ♖c4; ... ♔a7
2. ♔b6 I, ♔c6; ... ♔a6
3. ♔b6 I, ♖a4 #

Considerando il re nero in b8:

1. ♖c4; ... ♔b7
2. ♔b6 I, ♔c6 I, ♔a5; ... ♔a7
3. ♔b6 I, ♖b4; ... ♔a8
4. ♔b6; ... ♔b8
5. ♖c4; ... ♔a8
6. ♖c8 #

Anche in quest'ultimo esempio si nota la manovra iniziale dell'algoritmo basato su Boyce che mira dapprima a ottenere una metaposizione di tipo A, B, C o D.

4.6.2 Le prove eseguite

Procedura di Boyce

Il test per l'algoritmo estrapolato dalla procedura di Boyce, descritto in 4.4, è lo stesso eseguito per la formulazione definitiva dell'algoritmo di ricerca, proposto in 4.5.

Il test è stato effettuato facendo giocare tutte le possibili posizioni iniziali del finale di torre e re contro re, con incertezza massima sulla scacchiera di riferimento iniziale del Bianco. La strategia del nero tende ad accentrare il re allontanandolo dagli angoli; a questa è stato aggiunto un fattore casuale sulle mosse più promettenti per il nero e generalizzarne l'esito. Considerando le riflessioni del re nero e le simmetrie il numero totale di partite è risultato essere pari a 23424.

Su 23424 partite, 22946 sono state vinte con una media di 35 mosse per partita, ove la peggiore ha impiegato 66 mosse; 420 sono terminate per patta in 1 mossa, poiché la torre, cercando di avvicinarsi al re bianco, ha mosso in una casa adiacente al re nero ed è stata catturata. Questa cattura risulta inevitabile nella prima mossa, se la torre parte in una posizione scoperta; si verifica raramente, ma, quando si verifica, la probabilità che la torre venga presa è piuttosto alta, pari cioè a 1/8 nel caso ottimo.

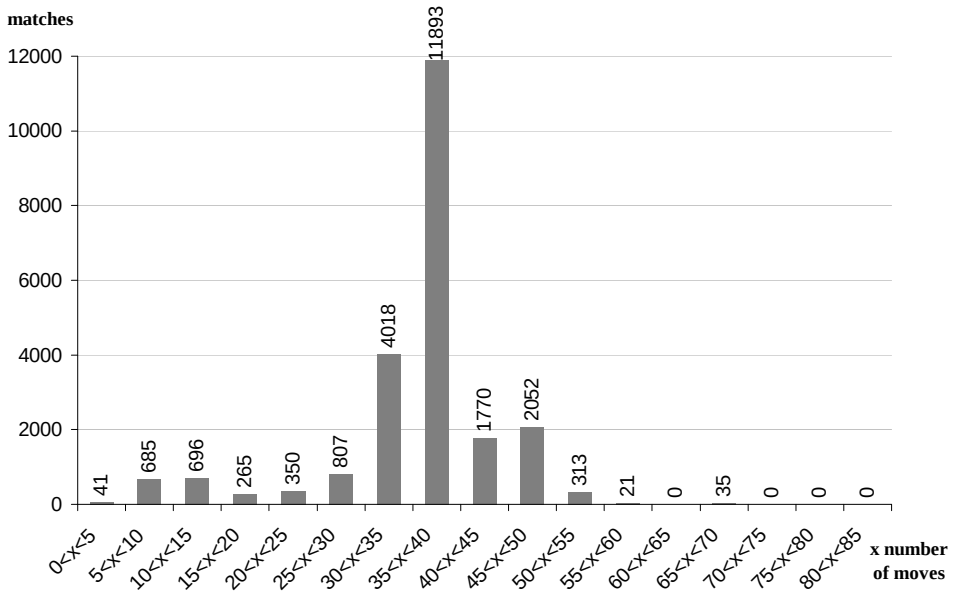


Figura 4.43: Istogramma di tutti i finali di partita, con algoritmo basato su Boyce

Infine 58 partite sono terminate per stallo in 1 o 2 mosse. Questo caso in realtà, come il precedente, non mette alla luce alcun errore grave del giocatore artificiale. Esso non può essere considerato se non nel contesto del test eseguito.

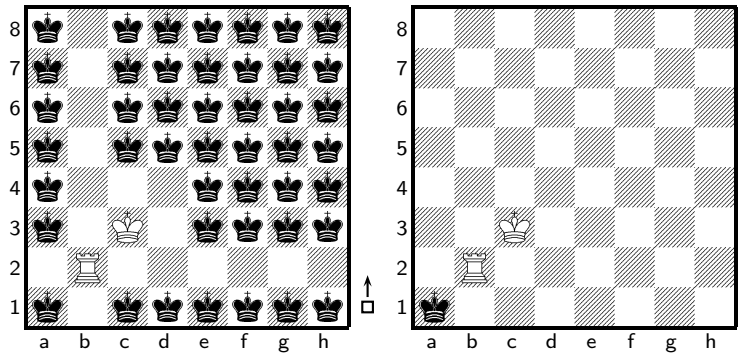


Figura 4.44: Metaposizione in possibile stallo

Le partite finite per stallo corrispondono allo $0,25\%$ di quelle giocate e sono della forma mostrata in figura 4.44, dove sulla sinistra è mostrata la metaposizione dal punto di vista del Bianco, mentre sulla destra la scacchiera dell'arbitro. Con il tratto al Bianco, una posizione di questo tipo non è stallo, a patto che il Bianco muova la torre. È vero altresì che una posizione del genere difficilmente capita al termine di una partita.

La funzione per evitare lo stallo, introdotta con l'algoritmo di ricerca definitivo, evita comunque il verificarsi di terminazioni di questo tipo, dando una penalità alla valutazione della metaposizione che potenzialmente può nascondere uno stallo.

Il $97,96\%$ di partite è quindi vinto, lo $0,25\%$ è stallo, mentre l'$1,79\%$ è patta, poiché inevitabilmente la torre bianca viene catturata.

L'istogramma in figura 4.45 mostra nel dettaglio il numero di partite per numero di mosse impiegate dall'algoritmo tratto da Boyce.

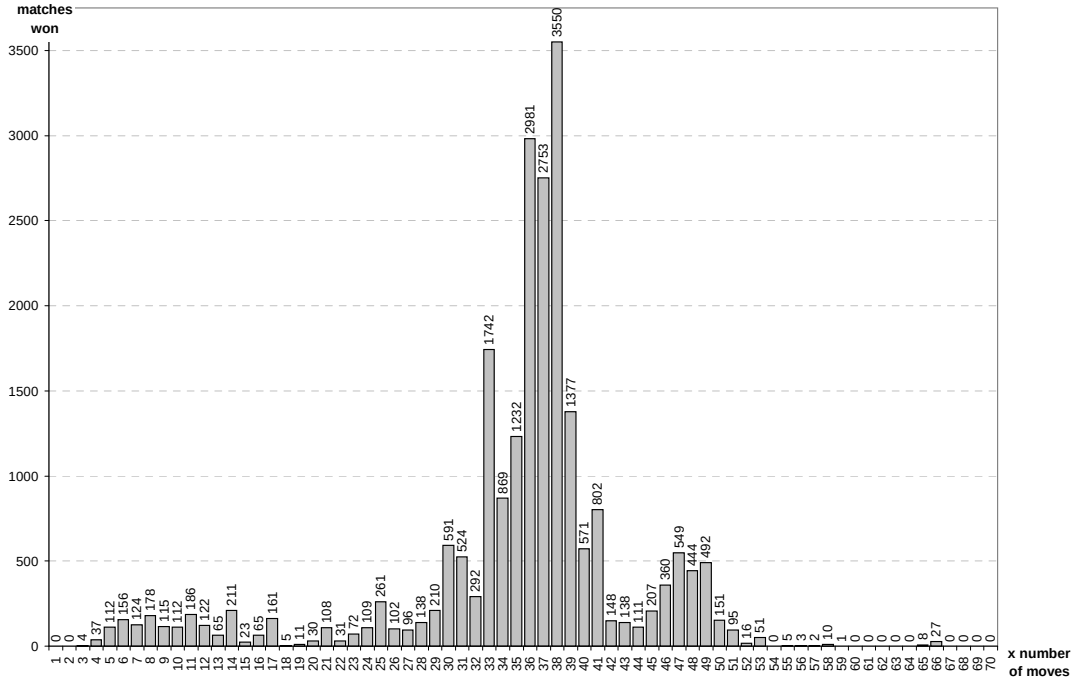


Figura 4.45: Istogramma dettagliato di tutti i finali di partita, con algoritmo basato su Boyce

Prima formulazione dell'algoritmo di ricerca

Inizialmente ho eseguito un test su 26536 posizioni iniziali, scelte casualmente dalle 175168 posizioni legali del finale di re e torre contro re.

Ogni posizione iniziale ha una scacchiera di riferimento per il bianco con incertezza massima, ovvero, dal punto di vista del Bianco, il re nero può essere posizionato ovunque nella scacchiera. La strategia del Nero consiste nel giocare sempre la mossa che gli permette di accentrarsi nella scacchiera e, in particolare, allontanarsi dagli angoli.

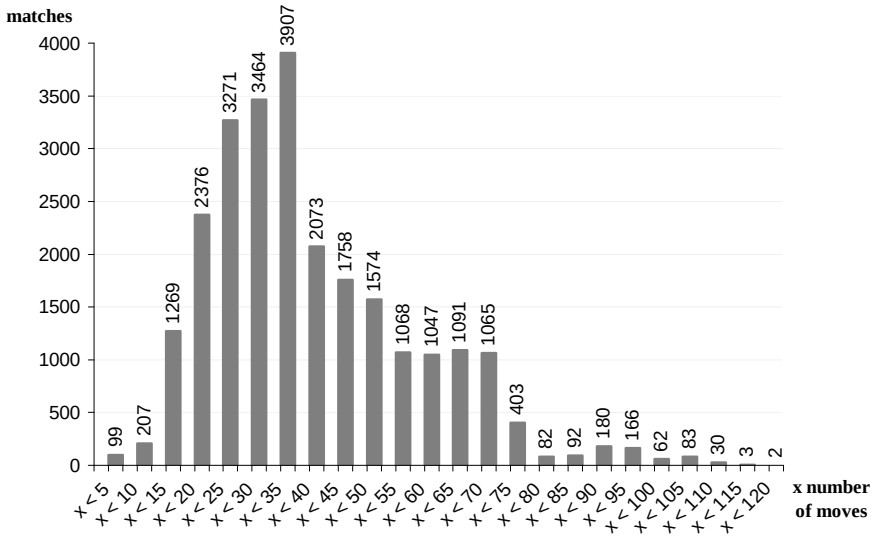


Figura 4.46: Test con posizioni iniziali casuali

Il test mostra che il 95.6% di partite sono vinte dal Bianco, mentre il 4.4% termina senza la vittoria. In particolare il 75.9% di queste è un gioco terminato per ciclo, il 24% per patta e il 24% per stallo. Il numero medio di mosse impiegato per dare il matto è 36 e la partita peggiore ha impiegato 117 mosse.

Nell'istogramma in figura 4.46 viene mostrata la durata delle partite vinte, ogni 5 mosse.

Per avere un confronto esauriente, ho eseguito un ulteriore test utilizzando la prima versione dell'algoritmo di ricerca e la prima stesura della funzione di valutazione. Questa prova è stata compiuta "dal punto di vista dell'arbitro", ovvero se, durante una partita, il gioco cominciava o attraversava posizioni precedentemente giocate, l'arbitro interveniva considerando il gioco vinto o perso a seconda dell'esito dalla posizione conosciuta.

In questo test 99.5% delle partite è vinto. Questo risultato mostra come la prima stesura (pubblicata in [3]) fosse buona, ma non perfetta. Si tratta in fatti di un test che aiuta molto il Bianco e ne da una sovrastima delle capacità di gioco. Ciononostante, non tutte le partite sono vinte.

L'istogramma in figura 4.47 mostra l'insieme delle partite vinte ogni 5 mosse e il numero di mosse necessarie per la vittoria.

Formulazione definitiva dell'algoritmo di ricerca

Le prove effettuate sulla versione definitiva dell'algoritmo di ricerca hanno mostrato l'efficacia dei miglioramenti apportati nella funzione di valutazione

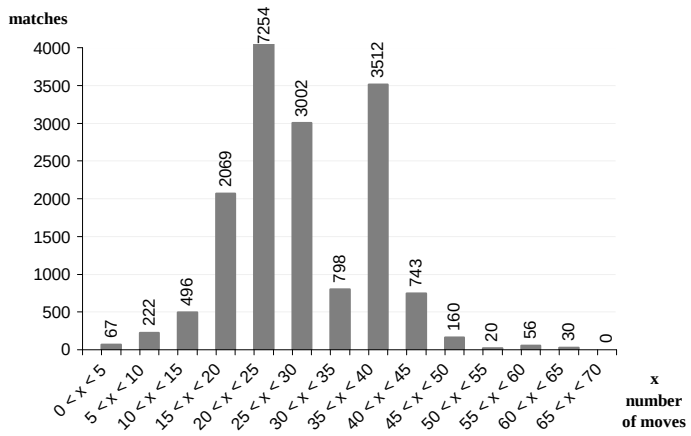


Figura 4.47: Test con suggerimenti dell'arbitro

e nell'evitare lo stallo ricercando lo scacco matto.

Il test è stato effettuato facendo giocare tutte le possibili posizioni iniziali del finale di torre e re contro re, con incertezza massima sulla scacchiera di riferimento iniziale del Bianco. La strategia del nero tende ad accentrare il re allontanandolo dagli angoli; a questa è stato aggiunto un fattore casuale sulle mosse più promettenti per il nero per generalizzarne l'esito.

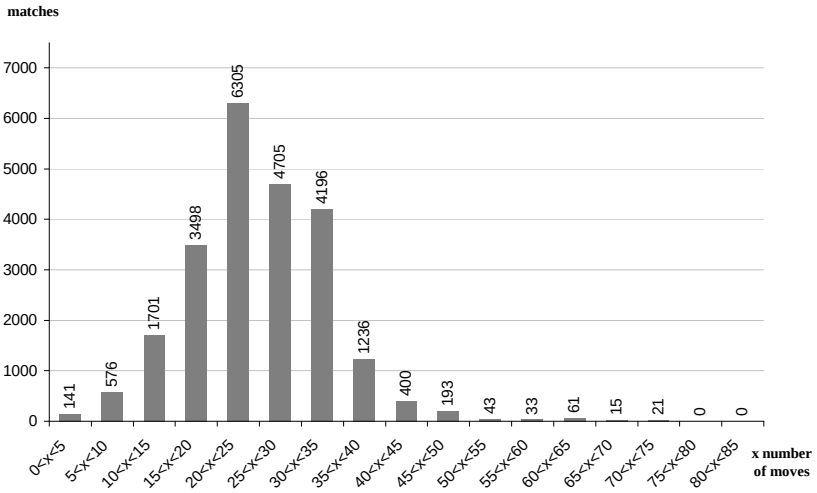


Figura 4.48: Istogramma di tutti i finali di partita, usando l'algoritmo di ricerca definitivo

Come nel caso del test sulla procedura di Boyce, il numero totale di partite è risultato essere pari a 23424. La profondità di ricerca sull'albero delle mosse è stata posta pari a 2. In nessuna partita il giocatore artificiale è incorso in cicli, come era accaduto invece nella prima versione, e non vi è stata alcuna partita terminata per stallo. Questa è una prova del fatto che

l'algoritmo termini sempre. Su 23424 partite, 23124 sono state vinte con una media di 24 mosse per partita, ove la peggiore ha impiegato 74 mosse; 300 sono terminate per patta in 1 mossa, poiché la torre, cercando di avvicinarsi al re bianco, ha mosso in una casa adiacente al re nero ed è stata catturata. Come detto precedentemente, questa cattura risulta inevitabile nella prima mossa, se la torre parte in una posizione scoperta.

Il 98, 72% di partite è quindi vinto, mentre il 1, 28% è inevitabilmente patta. Quest'analisi mostra quindi che l'algoritmo è sempre vincente, qualsiasi sia la posizione iniziale. In figura 4.48 è riportato l'istogramma che mostra tutte le partite del test e le mosse impiegate in ciascuna partita, raggruppate ogni 5.

In figura 4.49, lo stesso test è rappresentato in un istogramma più dettagliato in cui le partite sono classificate in base alle mosse impiegate, senza raggruppamenti.

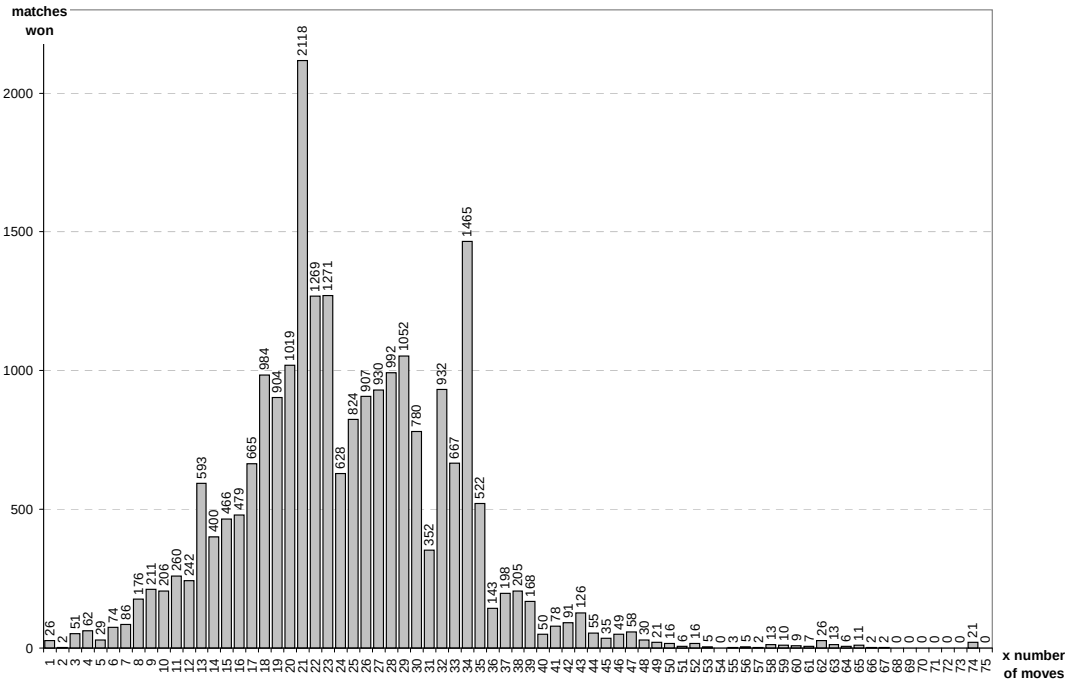


Figura 4.49: Istogramma dettagliato di tutti i finali di partita, usando l'algoritmo di ricerca definitivo

Capitolo 5

Altri finali

5.1 Il finale di Donna

Come per il gioco degli Scacchi, anche in Kriegspiel il finale di Donna risulta essere il più semplice. Negli Scacchi il numero di mosse necessario per vincere la partita è pari a 10 [25]; in Kriegspiel il programma del finale di Torre applicato al finale di Donna ha ottenuto la vittoria al più in 48 mosse, con una media di 19.

Per prima cosa è bene osservare come in Kriegspiel le metaposizioni in cui si rischia lo stallo sono molto frequenti, tanto che un pezzo potente come la Donna vi si imbatte decine di volte in una partita. Nella sezione 4.5.8 è stata descritta l'implementazione di una funzione atta ad evitare stati di questo tipo. Inizialmente infatti non avevo dotato il giocatore artificiale di alcuna funzione con questo scopo ed esso si era rivelato discreto nel caso del finale di Torre. Nel passaggio al caso della Donna, però, il numero di stalli, eseguendo un test di partite con posizione iniziale di incertezza massima, era piuttosto alto. Con l'introduzione della funzione $playerMaybeStale()$ il numero di stalli è stato azzerato. Si consideri la figura 5.1, essa mostra una metaposizione che ha una probabilità pari a 1/32 di essere in stallo, infatti la scacchiera dell'arbitro potrebbe essere quella di destra e l'obiettivo del Bianco sarebbe perso. Sebbene sembri una bassa probabilità essa si verifica di frequente, pertanto le metaposizioni di questo tipo vanno penalizzate durante la ricerca. La funzione $playerMaybeStale()$ permette appunto di scartare completamente quelle mosse che conducono in stati di questo tipo.

Utilizzando la medesima funzione di valutazione applicata al caso del finale di Torre, sul totale di 19376 partite, 19241 sono state vinte, 135 sono terminate per patta alla prima mossa. Questo indica, come si poteva supporre, che l'euristica da utilizzare per la Donna non deve essere troppo

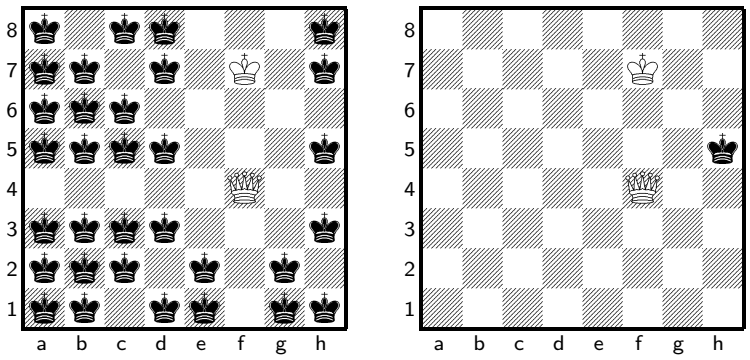


Figura 5.1: Metaposizione in potenziale stallo

differente da quella per la Torre. Il 0.7% dei casi però, sottolinea un difetto che il programma non deve avere nel giocare questo finale, che deve essere sempre vinto. Il motivo è imputabile al fatto che si tenta di avvicinare al Re la Donna, così come si faceva con la Torre.

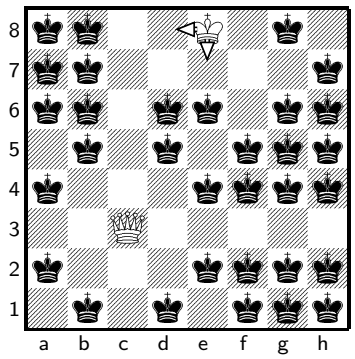


Figura 5.2: La regina non può muovere

Si consideri la miniatura in figura 5.2. In questa metaposizione la Regina non può muovere, poiché ogni sua pseudomossa la conduce in uno stato di possibile cattura; per avvicinarla al Re è necessario privilegiare le mosse che portano il Re stesso sulla stessa riga, o colonna, o diagonale della Regina; quindi muovere la Regina per seconda.

La funzione di valutazione è stata quindi modificata decrementando il valore della metaposizione nel caso in cui la Regina risulti sotto attacco. È stato inoltre aggiunto un parametro di penalizzazione pari al numero di re avversari presente sulla scacchiera di riferimento, in modo da incentivare il tentativo di diminuzione di incertezza. Infine è stata utilizzata una funzione simmetrica alla $evalArea()$ proposta nel caso della Torre, che divide la posi-

zione in quattro parti descritte dalle diagonali degli alfieri. Il codice di questa funzione, che prende il nome di *evalRhombArea*(), è riportato in tabella 5.1.

```
evalRhombArea (int q, int k, int c) {
    int i, a0, a1, a2, a3, diags, diagd, res;

    diags=DIAGS(s);
    diagd=DIAGD(s);
    a0=a1=a2=a3=0;

    for(i=0; i <128; ++i) {
        if(player with color (c ^ 1)'s king on i) {
            if(DIAGD(i)<diagd && DIAGS(i)<diags) a3++;
            if(DIAGD(i)>diagd && DIAGS(i)<diags) a0++;
            if(DIAGD(i)<diagd && DIAGS(i)>diags) a2++;
            if(DIAGD(i)>diagd && DIAGS(i)>diags) a1++;
        }
    }
    i=0;
    if(a0!=0) i++;
    if(a1!=0) i++;
    if(a2!=0) i++;
    if(a3!=0) i++;

    return i*(a0+a1+a2+a3);
}
```

Tabella 5.1: Funzione *evalRhombArea*()

Le diagonali vengono divise in due gruppi come mostrato in figura 5.3. Quelle, la cui direzione in figura è orientata dal basso verso l'alto a sinistra, sono calcolate tramite la macro $DIAGS(x)$, che accetta in input un intero $x = 1, .., 128$ raffigurante la casa della scacchiera e calcola il numero corrispondente alla diagonale a cui la casa appartiene. Sulla destra, rispettivamente è riportata una miniatura con la numerazione delle diagonali destre, calcolate tramite $DIAGD(x)$.

L'obiettivo, per il giocatore bianco è allora quello di diminuire l'area del triangolo all'interno del quale è presente il re nero. Come era stato nel caso della funzione *EvalArea*() descritta in 4.3, la funzione *evalRhombArea*() si occupa di questo aspetto tentando, da un lato, di diminuire l'area dei quattro triangoli, dall'altro di diminuire l'incertezza su quale sia il triangolo in cui

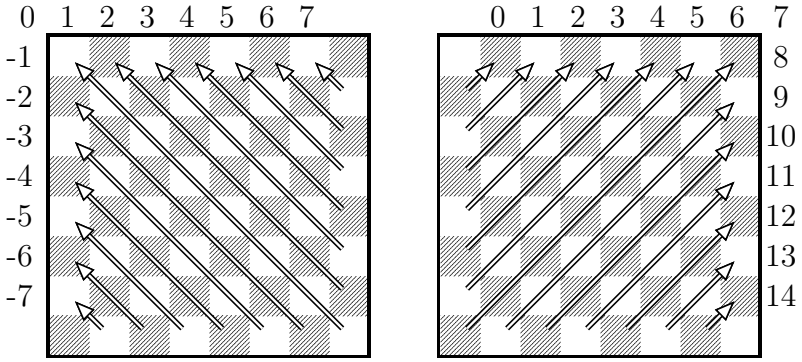


Figura 5.3: Conteggio delle diagonali per la funzione *evalRhombArea*()

è effettivamente presente il re nero. Definiamo la funzione *evalRhombArea* come

$$evalRhombArea(S) = c \cdot (a_0 + a_1 + a_2 + a_3) \tag{5.1}$$

ove $c \in \{1, 2, 3, 4\}$ è il valore che tiene traccia del numero di triangoli in cui è possibile che si trovi il re nero, e a_i $(i = 0, ..3)$ indicano il numero di posizioni possibili per il re avversario in ciascun triangolo. In figura 5.4 viene proposto il calcolo delle aree nel caso pessimo in cui si ha l'incertezza massima sul re nero pari a 132. In figura sono inoltre mostrate le posizioni delle tre aree utilizzate nel calcolo.

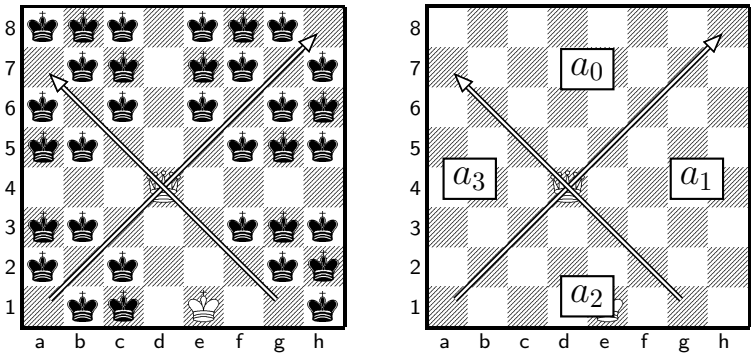


Figura 5.4: Significato grafico della funzione *evalRhombArea*()

In figura 5.5 è riportato l'istogramma che rappresenta il numero di mosse impiegato per ciascuna partita vinta, effettuando un test analogo a quello eseguito per il finale di Torre, con algoritmo di ricerca a profondità 2.

5.1.1 Alcune partite

Si consideri la miniatura in figura 5.6 sulla sinistra. In qualsiasi posizione sia il re nero, il giocatore artificiale ottiene correttamente lo scacco matto con

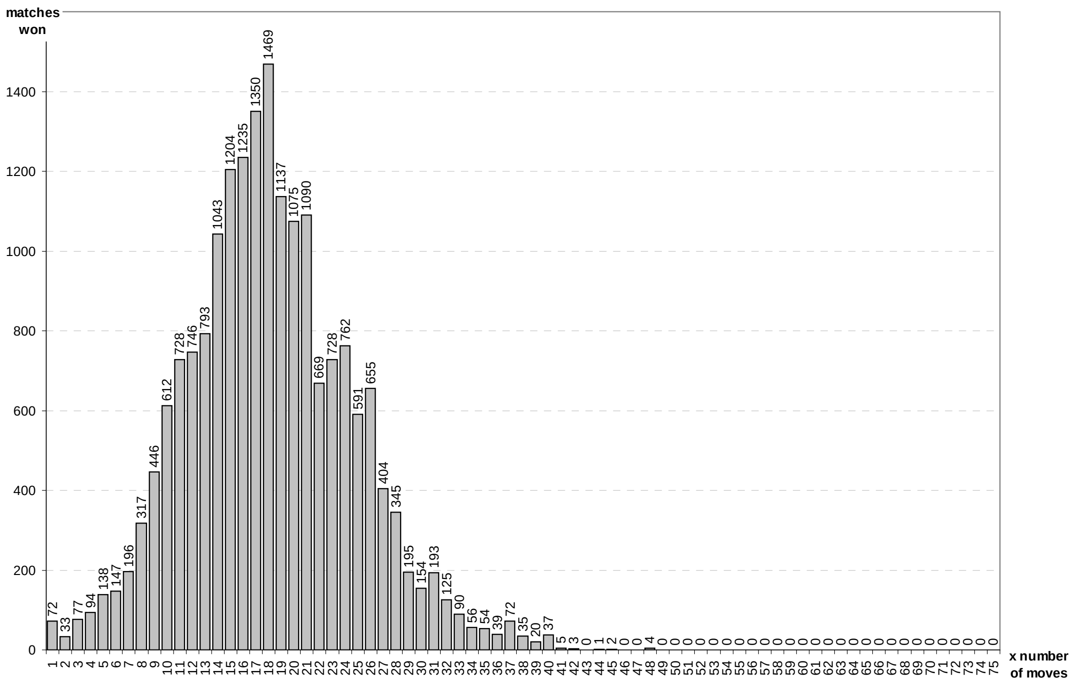


Figura 5.5: Istogramma dettagliato del finale di Donna

una mossa, giocando ♕a4#.

Si consideri invece la miniatura in figura 5.6 sulla destra. Posizionando il re nero in a8, la partita giocata dal programma prosegue come segue:
1. ♔c7; ...♔a7
2. ♔b6I,♕d3; ...♔a8
3. ♕a6#.

Posizionando il re nero in a7, la partita giocata dal programma prosegue come segue:
1. ♔c7; ...♔a6
2. ♔b6I,♕d3+; ...♔a5
3. ♕b3; ...♔a6
4. ♕a4#.

Posizionando il re nero in b8, la partita giocata dal programma prosegue come segue:
1. ♔c7I,♔c6; ...♔a8
2. ♕h6; ...♔a7
3. ♔c7; ...♔a8
4. ♕a6#.

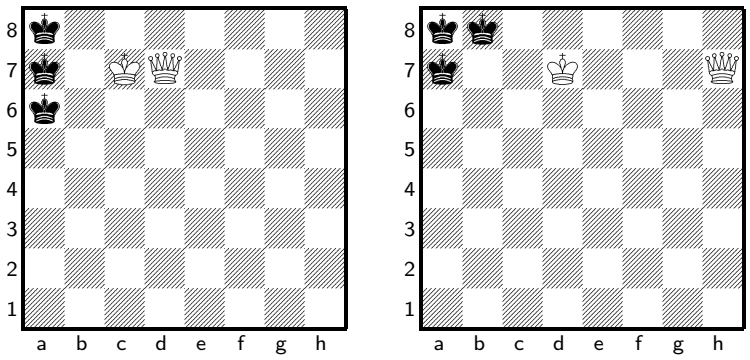


Figura 5.6: Alcuni esempi

5.2 Il finale con due Alfieri

Il finale di gioco che affrontiamo in questa sezione, vede il Bianco con il Re e due Alfieri contro il Re Nero da solo. Come mostrato in [15], a partire dalla posizione indicata dalla miniatura in figura 5.7, non esiste una strategia per il Bianco che gli permetta di vincere con probabilità uno, avendo incertezza massima sulla posizione del Re avversario. Ciò che si può dire è che dato $\epsilon > 0$ esiste una strategia per il Bianco che vince con probabilità almeno $1-\epsilon$, a prescindere dalla posizione iniziale e dalla strategia del Nero. Usando la terminologia vista nella sezione 2.3, si dice che esiste una strategia ϵ-ottimale per il bianco. La ragione per cui non esiste una strategia ottima per il Bianco è una conseguenza del fatto che il Bianco non può raggiungere una posizione in cui necessita di difendere gli alfieri solo da un lato. In particolare, egli non può raggiungere il bordo della scacchiera senza rischiare di perdere un alfiere o di permettere un possibile stallo. Se egli raggiunge il lato della scacchiera con successo, può sfruttare una strategia che dà scacco sicuramente in un numero finito di mosse senza usare metodi probabilistici, ampiamente approfondita in [15]. Dalla posizione in figura 5.7 il Bianco può muovere verso il bordo senza alcun rischio, giocando ♗c4,♔d5,♗d4,♔c5; in questo modo raggiunge la posizione mostrata in figura 5.8.

Per portare il re e gli alfieri verso il bordo e mantenere la loro copertura, il Bianco deve giocare ♗b5, rischiando lo stallo in a5. Quindi nel tentativo di raggiungere il lato della scacchiera, il Bianco deve scegliere probabilisticamente tra ♔d5,♔c5 e ♔b4,♗c3, dando rispettivamente peso $1-\epsilon$ e ϵ. Il Bianco ripete la prima sequenza di mosse indefinitamente fino a quando, ad un certo tempo casuale, egli gioca la seconda sequenza. Egli ha successo se in ogni momento riceve la risposta "mossa illegale" dall'arbitro o se, quando muove ♔b4 l'alfiere in d4 non viene immediatamente catturato dal re nero. È facile vedere che la probabilità con cui il Bianco ha successo è pari a $1-\epsilon$,

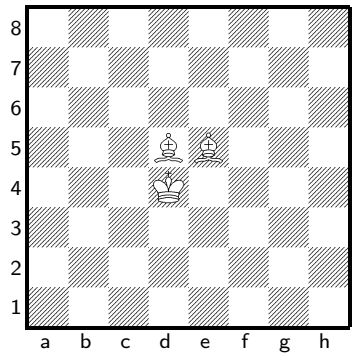


Figura 5.7: Ferguson, 1995

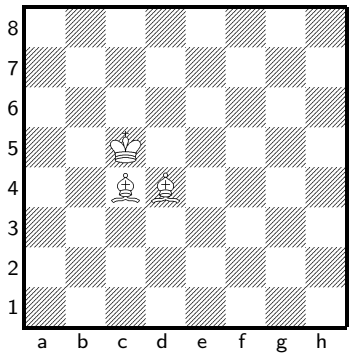


Figura 5.8: Posizione critica

a prescindere dalla strategia del Nero.

Una volta raggiunto il lato della scacchiera, il Bianco potrà dare il matto in un numero finito di mosse in modo non probabilistico. La procedura proposta da Ferguson in [15] prevede di partire dalla posizione in figura 5.9 o simili, per riflessione e simmetria. Dopo aver giocato ♔b4, ♗c3, il Bianco gioca semplicemente ♗b3 per ricondursi alla posizione iniziale.

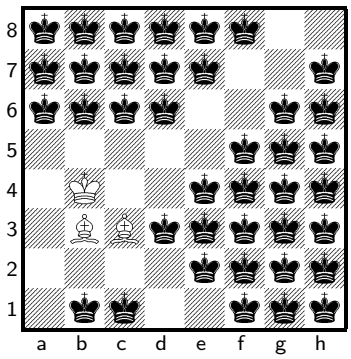


Figura 5.9: Posizione iniziale

Si osservi che, nella posizione di figura 5.9, gli alfieri non possono essere attaccati né dalla sinistra, né dall'alto, né dal basso. Il Bianco può quindi procedere a scandagliare la scacchiera alla ricerca del re nero, facendo attenzione a non mettere in pericolo gli alfieri dal lato destro. Egli può procedere come segue:

♔c4, ♔d3, ♔e3, ♔f3, ♔g2, ♔f3, ♔g4, ♔f5, ♔g6

Allorquando l'arbitro risponde "mossa illegale", il Bianco intrappola il re nero nella parte destra bassa o nella parte destra alta della scacchiera; altrimenti il re nero può essere posizionato nella parte sinistra alta o in quella sinistra in

basso. L'intera procedura, che comprende numerosi passi, è mostrata in [15]. Essa garantisce lo scacco matto in al più 32 mosse, a partire dalla posizione iniziale di figura 5.9.

5.2.1 L'algoritmo

La procedura descritta in [15], che muove da quanto mostrato nella sezione precedente, è analoga alla procedura di Boyce nel caso della Torre, poiché indica una sequenza di passi da seguire a seconda della posizione in cui ci si trova. Può essere interessante però, analizzare il comportamento di un algoritmo universale che funzioni correttamente anche per questo finale, pertanto si è tentato di sfruttare l'algoritmo utilizzato già per il finale di Torre e il finale di Donna. Mantenendo la stessa funzione di valutazione, i risultati non sono stati buoni, in quanto il giocatore artificiale mancava del progresso necessario per far terminare la partita. Ho quindi provveduto a lavorare sulla funzione di valutazione, formulandone una atta a giudicare le posizioni che le vengono sottoposte specifiche del finale con due alfieri.

Innanzitutto la matrice numerica, indicata nel finale di Torre dalla funzione f_5, è stata modificata per l'uso specifico degli alfieri. Essa è indicata in figura 5.10. Durante la fase di valutazione, viene eseguita una scansione della scacchiera di riferimento del Bianco: se una casa ha probabilità maggiore di zero di ospitare un re nero, allora la valutazione della metaposizione viene decrementata del valore pari all'entrata nella matrice *boardAnalisiBishop*.

$$\begin{pmatrix} 0 & -10 & -50 & -100 & -100 & -50 & -10 & 0 \\ -10 & -10 & -40 & -40 & -40 & -40 & -10 & -10 \\ -50 & -40 & -40 & -40 & -40 & -40 & -40 & -50 \\ -100 & -40 & -40 & -50 & -50 & -40 & -40 & -100 \\ -100 & -40 & -40 & -50 & -50 & -40 & -40 & -100 \\ -50 & -40 & -40 & -40 & -40 & -40 & -40 & -50 \\ -10 & -10 & -40 & -40 & -40 & -40 & -10 & -10 \\ 0 & -10 & -50 & -100 & -100 & -50 & -10 & 0 \end{pmatrix}$$

Figura 5.10: La matrice numerica *boardAnalisiBishop*

Se la somma dei valori attivi nella matrice è inferiore a -600, significa che il Bianco ha un'informazione molto limitata circa la posizione del re nero, pertanto si decrementa il valore della metaposizione utilizzando la funzione *evalRhombArea*() vista per il finale di Donna e moltiplicandone il risultato per 4. Questo viene eseguito per entrambi gli alfieri. Altrimenti,

se cioè l'informazione sulle posizioni avversarie sono discrete e la somma dei valori della matrice è maggiore di -600, si fa ancora ricorso alla funzione $evalRhombArea()$, ma, questa volta, se ne divide il risultato per 6. In altre parole, se l'incertezza è alta si dà molta importanza al risultato della funzione $evalRhombArea()$, altrimenti se ne considera il risultato, ma in modo marginale.

Si noti che i fattori che vengono applicati al risultato sono stati scelti tramite tentativi, che hanno portato ha delineare l'impiego migliore delle funzioni che compongono la valutazione euristica della metaposizione. Il ragionamento seguito ha una base comune con quello descritto per il finale di Pedone nella sezione 3.2.8 in cui si è proceduto a scrivere una funzione in grado di rappresentare il giudizio considerato corretto e si è passati a raffinarla laddove mostrava difetti.

Si è quindi utilizzata una funzione analoga alla $endingRook(c, rook, king)$ descritta in 4.5.9, chiamata $BKborder(c, bishop, king)$. Essa considera il rettangolo formato dal re e da un alfiere, ma a differenza della $endingRook()$ considera sottoinsiemi della scacchiera formati da intere colonne o da intere righe. Se un re avversario è presente all'interno di uno dei due rettangoli, uno formato dalle righe, l'altro formato dalle colonne tra l'alfiere ed il re, allora si giudica negativamente la posizione, decrementandone il valore di un valori pari a 25.

Poiché considera un solo alfiere, essa viene applicata due volte. In figura 5.11 sono riportate alcune miniature d'esempio che descrivono il giudizio restituito dalla funzione $BKborder()$.

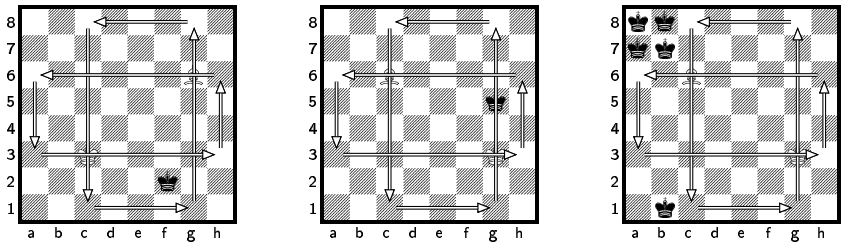


Figura 5.11: Esempio di quadranti della funzione *BKborder*

Se i due alfieri sono adiacenti il valore della metaposizione viene aumentato di 2, mentre se il re è nella stessa colonna o nella stessa riga degli alfieri il valore viene aumentato di 1.

Quindi si prediligono alcune posizioni del re, aumentando il valore della metaposizione di 30 nel caso in cui il re Bianco occupi le case c7, b6, f7, g6, b3, c2, g3 e f2, se è opportunamente supportato dagli alfieri. Si consideri senza

perdere di generalità che il re bianco sia in c7, gli altri casi possono essere considerati come le simmetrie e riflessioni di quello preso in considerazione. Se si ha il re Bianco in c7, allora se uno dei due alfieri si trova in c4 la posizione viene considerata buona, poiché l'alfiere supporta il re, e si aumenta di 30 il valore. A questo punto, si predilige la mossa che costringe all'angolo il re avversario e che vede quindi la presenza del primo alfiere in c4 e del secondo alfiere in c5. In questo modo lo scacco segue in una mossa. Quanto detto è espresso graficamente in figura 5.12. Sulla sinistra è riportata una miniatura con tutte le posizioni potenzialmente buone per il Bianco, in cui sono evidenziate quelle prese in considerazione nell'esempio. Se si ha il re Bianco in c7, come mostrato nella miniatura centrale, si può considerare buona la posizione che vede un alfiere in c4, in grado di bloccare il re nero all'angolo; quindi, come mostrato nella miniatura di destra, si può muovere il secondo alfiere in c5, costringendo il Nero a ♚a8 e quindi procedere con ♗c5#.

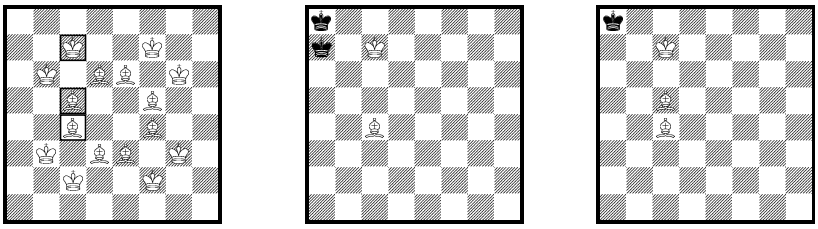


Figura 5.12: Esempio di posizioni buone per il finale con due alfieri

Infine, si sono penalizzate le mosse che non avvicinano gli alfieri al re, in situazioni di alta incertezza circa la posizione del re avversario, quando cioè la somma degli elementi della matrice *boardAnalisiBishop* corrispondenti alle case possibili occupate dal re nero è inferiore a -750.

È bene notare che la funzione di valutazione non considera l'aspetto probabilistico evidenziato da Ferguson, circa l'avvicinamento al bordo dei due alfieri. Teoricamente questo aspetto è importante e dovrebbe essere inserito. Come discusso nella sezione 3.2.8 però, l'uso di una funzione generatrice di numeri casuali all'interno di una funzione di valutazione impedisce l'uso di alcune euristiche di potatura durante la ricerca sull'albero. Questo avviene ad esempio per l'uso della tabella di trasposizione, che rischierebbe di tenere conto di valutazioni precedentemente effettuate, perdendo in questo modo la componente stocastica.

D'altra parte, non usare una tabella di trasposizione rende la ricerca della mossa più complessa e richiede una potenza di calcolo di cui non disponia-

mo. Non è quindi possibile introdurre nella nostra implementazione questo aspetto.

Poiché si fa uso della funzione *playerMaybeStale*() descritta in 4.5.8, che evita qualsiasi metaposizione in cui è possibile il verificarsi dello stallo, siamo certi che il giocatore eviterà sempre mosse quali, ad esempio, ♗b5 partendo dalla configurazione di figura 5.8. Poiché inoltre, nell'algoritmo di ricerca è inserito un controllo sulle ultime mosse giocate, per evitare che si ripetano mosse recenti, si è certi che il programma sceglierà via via mosse meno promettenti. Questo comporta, con un certo margine di rischio, che date due mosse equivalenti venga scelta la prima in ordine di calcolo, ma, successivamente, la seconda se ci si riconduce nella medesima posizione. Chiaramente, non si tratta di un metodo corretto, ma di un compromesso che risulta abbastanza efficace, come vedremo nella sezione dedicata ai test.

5.2.2 Il test

Come per gli altri finali, si è provveduto ad eseguire un test facendo giocare il programma a partire da metaposizioni con incertezza massima sulla posizione del re avversario. Poiché il numero di posizioni iniziali può essere approssimato a $64 \cdot 32 \cdot 32 \cdot 30 \simeq 1900000$, considerando pari a 30 il numero di case in cui può essere posizionato il re nero dopo aver disposto il re bianco e i due alfieri, non è stato possibile effettuare il test su tutte le configurazioni iniziali. Si è quindi provveduto ad eseguire il test generando casualmente le metaposizioni di partenza. Su 1942 partite giocate, 1790 sono state vinte dal Bianco con una media di 34 mosse; 70 sono terminate per mancanza di progresso ed è stata quindi interrotta la partita, 82 sono terminate per la cattura di un alfiere e quindi per patta. Come detto, la valutazione compiuta dal programma è imprecisa, non potendo tener conto dell'alea necessaria ad affrontare alcune situazioni. Per questo, il programma gioca via via mosse imprecise quando non riesce a raggiungere il suo obiettivo, che consiste nell'intrappolare il re avversario ad un angolo della scacchiera. Le metaposizioni in possibile stallo vengono sempre evitate, quindi non si affrontano mai rischi, che invece andrebbero affrontati con una certa probabilità. Questo spiega il motivo per cui il 3,6% di partite è stato terminato in modo anomalo, in quanto il giocatore artificiale ripeteva mosse senza guadagnare progresso. Il 4,22% di partite invece, ha visto il Bianco giocare mosse incaute e perdere un alfiere. Questo è dovuto alla procedura che evita di giocare mosse recenti e che comporta scelte via via più scadenti, fino a mosse rischiose. Il restante 92,17% come detto è vinto, mentre non si sono verificati casi di stallo.

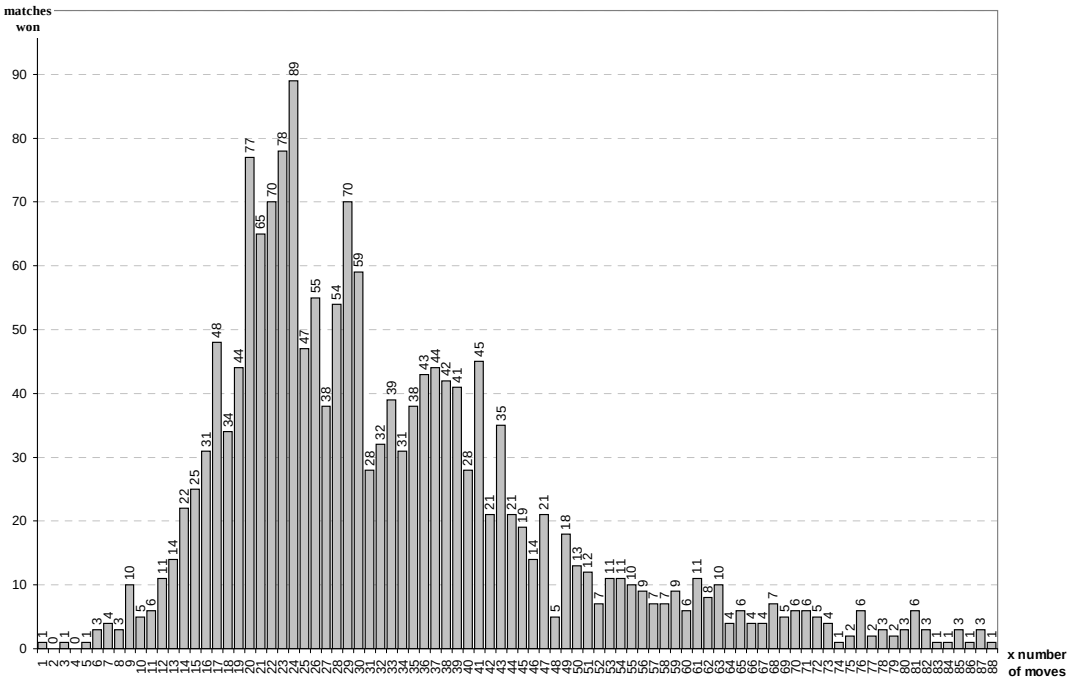


Figura 5.13: Istogramma dettagliato del finale con due alfieri

5.2.3 Alcune partite

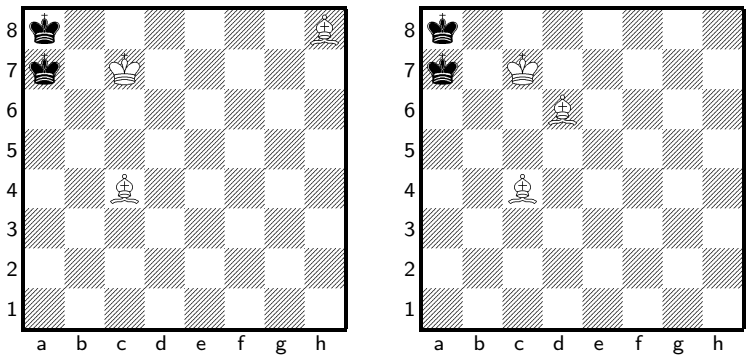


Figura 5.14: Alcuni esempi

Si consideri la miniatura in figura 5.14 sulla sinistra. Posizionando il re nero in a8, la partita giocata dal programma prosegue come segue:
1. ♔b6; . . . ♔b8
2. ♗e6; . . . ♔a8
3. ♗f6; . . . ♔b8

4. ♗e5+; ... ♔a8
5. ♗d5#.

Posizionando il re nero in a7:
1. ♔b6I, ♗d4+; ... ♔a8
2. ♗d5#.

Si consideri la miniatura di destra in figura 5.14. Posizionando il re nero in a8, la partita viene giocata dal programma come segue:
1. ♗b4; ... ♔a7
2. ♔b6I, ♗c4+; ... ♔a8
3. ♗d5#;

Posizionando il re nero in a7:
1. ♗b4; ... ♔a8
2. ♔b6; ... ♔b8
3. ♗e6; ... ♔a8
4. ♗e7; ... ♔b8
5. ♗d6+; ... ♔a8
6. ♗d5#.

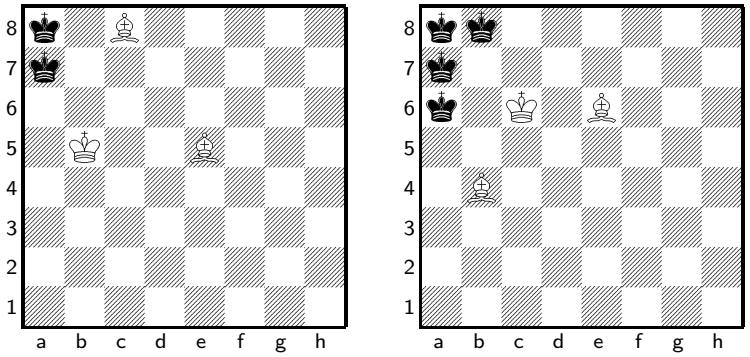


Figura 5.15: Alcuni esempi

Si consideri la miniatura di sinistra in figura 5.15. Posizionando il re nero in a8, la partita viene giocata dal programma come segue:
1. ♔c6; ... ♔a7
2. ♔c7; ... ♔a8
3. ♔c6; ... ♔a7
4. ♔c7; ... ♔a8
5. ♗f6; ... ♔a7
6. ♔b6I, ♗d4+; ... ♔a8
7. ♗b7#.

Posizionando il re nero in a7, la partita viene giocata dal programma come segue:
1. ♔c6; ... ♔a8
2. ♔c7; ... ♔a7
3. ♔c6; ... ♔a8
4. ♔c7; ... ♔a7
5. ♗f6; ... ♔a8
6. ♔b6; ... ♔b8
7. ♗e6; ... ♔a8
8. ♗e7; ... ♔b8
9. ♗d6+; ... ♔a8
10. ♗d5#.

Si può notare che, in principio, il Bianco gioca per due volte le stesse mosse. In questo caso la scelta viene influenzata dal vincolo per cui il programma scarta mosse recentemente giocate, pertanto al 5^o tratto scarta ♔c6 e gioca la mossa subito minore nella graduatoria delle buone mosse, che risulta essere ♗f6.

Chiaramente questo è un difetto del giocatore artificiale, e potrebbe essere un punto su cui lavorare per migliorarne le prestazioni. Comunque il gioco mostrato in questi matti elementari è buono e non si discosta troppo dalla soluzione per passi mostrata in [15]. Per il finale appena visto, Ferguson mostra che il matto si raggiunge al più in 7 mosse, mentre il programma ne impiega 10. Si consideri ora la miniatura di destra in figura 5.15. Ferguson mostra che il matto si ottiene con al più 7 mosse, mentre il giocatore artificiale ne impiega al massimo 6.

Posizionando il re nero in b8, la partita viene giocata dal programma come segue:
1. ♔b6; ... ♔a8
2. ♔c7; ... ♔a7
3. ♗b4; ... ♔a8
4. ♗e7; ... ♔a7
5. ♗c5+; ... ♔a8
6. ♗d5#.

Posizionando il re nero in a8, la partita viene giocata dal programma come segue:
1. ♔b6; ... ♔b8
2. ♔c7I,♗d6+; ... ♔a8
3. ♗d5#.

Posizionando il re nero in a7, la partita viene giocata dal programma come segue:

1. ♔b6I, ♗c4; . . . ♔a8
2. ♔c7; . . . ♔a7
3. ♗c4+; . . . ♔a8
4. ♗d5#.

Posizionando il re nero in a6, la partita viene giocata dal programma come segue:
1. ♔b6I, ♗c4+; . . . ♔a7
2. ♔c7; . . . ♔a8
3. ♗e7; . . . ♔a7
4. ♗c5+; . . . ♔a8
5. ♗d5#.

5.3 Il finale di Pedone

Questo finale è stato affrontato con razionalità procedurale in [8] e [24]. Ora se ne propone una soluzione tramite algoritmo di ricerca sull'albero delle mosse, così come è stato fatto per i finali di Torre, di Donna ed il finale con due Alfieri.

L'implementazione della funzione di valutazione nel caso del finale di Pedone, già descritto nelle sezioni 4.3 e 3.2.8, è interessante, poiché, essendo un finale semplice, permette l'applicazione di una funzione generatrice di numeri casuali. Questo vieta di usare la tabella di trasposizione e le chiavi di Zobrist, ma poiché il finale richiede una visita veloce dell'albero, avendo il Bianco un numero molto limitato di mosse, è possibile attuare una ricerca completa.

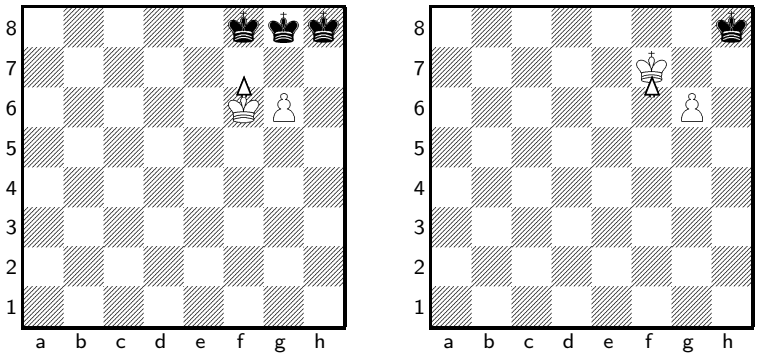


Figura 5.16: Il bianco non può tentare ♔f7

Si consideri che nel caso in cui il pedone sia presente nella prima o nella seconda colonna (o simmetricamente nella settima e nella ottava), la partita

termina per patta, in quanto la strategia probabilistica descritta da Magari non può essere applicata. Nel caso descritto in figura 5.16 infatti, se il Bianco tenta ♔f7 e la risposta dell'arbitro è silente, la metaposizione raggiunta è in stallo e la partita termina. Pertanto il Bianco non può tentare ♔f7 altrimenti la strategia ottima per il Nero porta allo stallo. Il Bianco deve quindi andare nella settima traversa sempre con il re in h6 e giocare ♔h7. Questo modifica le strategie di entrambi i giocatori. Se il Bianco spinge il pedone in settima e il re nero si trova in ♚h8 la partita termina per stallo, se invece il re nero si trova in ♚g8 il Bianco vince. In questa situazione il Nero è avvantaggiato, infatti, mentre il Bianco non può sapere dove si trovi l'avversario (se in h8 o in g8), il Nero può essere messo a conoscenza della posizione del re bianco dalla risposta dell'arbitro dopo la mossa ♔h7. In generale però i giocatori possono allungare le proprie manovre, rendendo impossibile l'intuizione all'avversario delle proprie mosse. Quindi si può concludere che, quando il Bianco tenta la spinta del pedone in settima, con probabilità di 1/2 vince la partita, e con probabilità di 1/2 perde per stallo. In questo caso non è quindi possibile alcuna strategia ϵ-ottimale.

In tabella 5.2 è mostrata la funzione di valutazione per il finale di Pedone, che rispecchia le direttive proposte nella sezione 3.2.8. Inizialmente il risultato viene posto a -500 ($res = -500$), poiché giunto alla settima traversa il programma deve considerare vantaggiosa la mossa che porta il pedone in promozione. Si considera quindi che lo stato con un pedone sia inferiore di 500 ad uno stato con la regina. Quindi si penalizza la situazione in cui il re non sia adiacente al pedone; si somma al valore il numero della riga in cui vi è il pedone moltiplicato per due, in modo da permetterne l'avanzamento; si prediligono le posizioni in cui il re sta in una riga superiore a quella del pedone.

Se il pedone è in settima traversa si proibisce che il re stia in una riga inferiore, altrimenti se anche il re è in settima oppure se il re è in ottava si predilige la mossa, aumentando il valore di 100.

Se il pedone è in sesta traversa si fa uso di una funzione generatrice di numeri casuali per scegliere tra le mosse di spinta del re sinistre o destre. Si assegna loro un valore pari a $5 + x$ ove $x = \{0, ..., 49\}$ è un valore casuale.

Infine si aumenta il valore di 25 nel caso in cui il re sia posizionato nella casa precedente a quella del pedone.

Il test effettuato per il finale di Pedone, il cui esito è rappresentato dall'istogramma in figura 5.17, è analogo a quello svolto per i finali affrontati precedentemente. Sono state giocate 1073 partite a partire da una metaposizione casuale con massima incertezza sul re nero, posizionando il re bianco nella quarta riga e il pedone bianco nella terza, evitando di porlo nella prima, seconda, settima ed ottava colonna. La funzione di ricerca per la mossa

```
evalPawn (int pawn, int king, int c) {
    int res;

    res=−500;
    if(king is not adjacent to the pawn)
      res−=200;
    else
      res−=distance between king and pawn;

    res+=ROW(pawn)*2;
    res+=ROW(king)-ROW(pawn);
    if(ROW(pawn)==seventh row)
      if(ROW(pawn)>ROW(king))
        return −100000;
      else
        res+=100;

    if(ROW(pawn)==sixth row) {
      if(king is ont the left of the pawn)
        res+=5+rand()%50;
      if(king is ont the right of the pawn)
        res+=5+rand()%50;
    }

    if(ROW(king)==ROW(pawn)−1)
      if(COL(king)==COL(pawn))
        res+=25;

    return res;
}
```

Tabella 5.2: Funzione $evalPawn()$

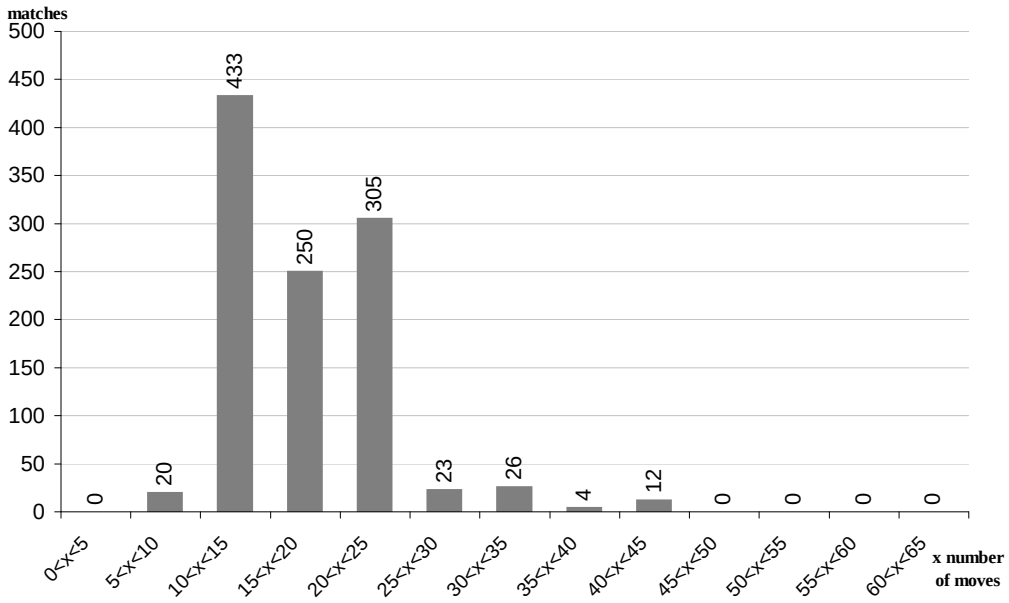


Figura 5.17: Istogramma delle prove effettuate nel finale di Pedone

del nero, che per i precedenti finali sceglieva le mosse in grado di accentrare nella scacchiera il re nero, ora sceglie le mosse che mantengono il re nero sulla ottava riga. Inoltre si è dato l'aiuto al Nero del suggerimento dell'arbitro che gli comunica la colonna in cui è posizionato il pedone, in questo modo il re nero muove sulla ottava riga collimando la propria posizione con la colonna del pedone.

Il 100% delle partite sono state vinte con una media di 16 mosse. Nessuna partita è terminata con oltre 40 mosse, quindi, anche con la regola delle 50 mosse, il Bianco avrebbe sempre vinto.

Capitolo 6

Conclusioni

Il percorso di lavoro seguito ha visto una prima fase di analisi della struttura del gioco e, in particolare, della struttura dei finali, in cui il Nero presenta un solo pezzo: il re. Si è partiti dal finale più studiato e approfondito in letteratura, il finale di Torre, e si è implementato un programma in grado di eseguire le procedure proposte. In particolare si è generalizzato quanto mostrato da Boyce. Essendo il numero di posizioni iniziali, nel finale di Torre, pari a circa 23000, è stato possibile eseguire un test che facesse giocare tale implementazione e si è così ottenuto un punto di riferimento molto utile per poter giudicare l'algoritmo di gioco basato su visita dell'albero delle mosse e valutazione euristica, che è stato il frutto principale di questa tesi.

Anche nell'implementazione dell'algoritmo di Boyce si è fatto uso di un algoritmo di ricerca, per poter permettere al giocatore artificiale di raggiungere una delle 4 posizioni iniziali della procedura per passi. Tale procedura ha quindi vinto con una media di 35 mosse. Si può pensare che questo valore dipenda in parte dall'algoritmo di ricerca iniziale che non sfrutta buone posizioni di gioco, avendo come obiettivo il raggiungimento di una delle posizioni iniziali della procedura di Boyce.

La soluzione da me proposta sfrutta invece un algoritmo di ricerca, che esegue una visita più dettagliata, considerando come fine ultimo quello del raggiungimento di posizioni, o meglio, metaposizioni vinte per il Bianco. Esso ha ottenuto la vittoria in media con 24 mosse. Questo risultato sottolinea la generalità del programma presentato, che analizza la metaposizione in cui si trova e, in base ai calcoli ricavati dalla visita delle pseudomosse possibili e dalle funzioni euristiche di valutazione statiche e ricorsive, prende la decisione considerata migliore.

Nella figura 6.1 è mostrato il confronto tra i test ottenuti con l'uso delle due implementazioni descritte. Tale test esaustivo prova la terminazione dell'algoritmo di ricerca, che come si vede raggiunge il massimo numero di

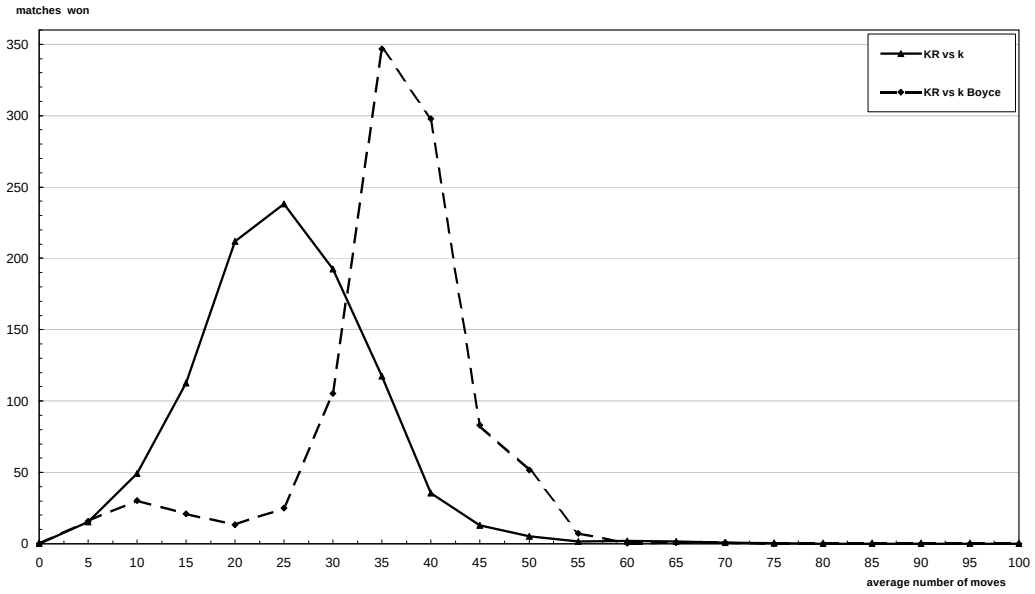


Figura 6.1: Confronto tra i test ottenuti con la procedura di Boyce e l'algoritmo di ricerca definitivo

partite vinte con una media di circa 25 mosse. La rappresentazione grafica della procedura di Boyce viene scalata verso destra e descrive una curva più accentuata, mostrando il picco massimo di partite vinte ad una media di 35 mosse circa.

È bene notare che il contro da pagare nell'implementazione tramite algoritmo di ricerca è dovuto al tempo impiegato dalla ricerca delle mosse, che cresce all'aumentare della profondità dell'analisi. D'altra parte la procedura ispirata alle direttive di Boyce ha complessità lineare.

Un ulteriore passo potrebbe consistere nel fare giocare il programma in modo che sfrutti l'algoritmo di ricerca e, nel caso in cui esso si imbatta in una metaposizione che rispetta le caratteristiche di una posizione iniziale nella procedura di Boyce, prosegua con l'esecuzione della procedura per passi ricavata dai suggerimenti di Boyce, corretti ed estremamente efficaci.

Avendo costruito e testato un motore di gioco in grado di muoversi abilmente in ogni situazione del finale di Torre, è stata mia intenzione cercare di ampliarlo ad altri finali possibili, e renderlo quindi ancor più generale.

Facile è stato il passaggio al finale di Donna, avendo essa molto potere e mosse in parte simili a quelle della torre. In questo frangente si è notato come le modifiche necessarie affinché il programma giocasse questo finale dipendessero esclusivamente dalla funzione di valutazione. Modificando questa

infatti, si ottenevano giudizi più o meno corretti sulle mosse e quindi decisioni più o meno efficaci.

I casi di stallo e le mosse per dare sicuramente lo scacco matto erano già considerati nel motore principale, e non è stata necessaria alcuna aggiunta. Ciò che si è fatto è stato mettere mano alle sottofunzioni giudicatrici delle metaposizioni.

Il risultato è stato buono per la donna e per i finali congiunti, ossia con due torri contro il re nero, oppure con la donna e la torre contro il re nero. In questo caso, come si osserva dal grafico in figura 6.2, come si sarebbe potuto pensare, il finale con torre e donna contro re è vinto con una media di 10 mosse, mentre il finale con due torri contro re vince con 14 mosse in media. Il grafico mostra le diverse velocità di convergenza con cui terminano le partite, quindi l'abilità del giocatore artificiale in tali situazioni.

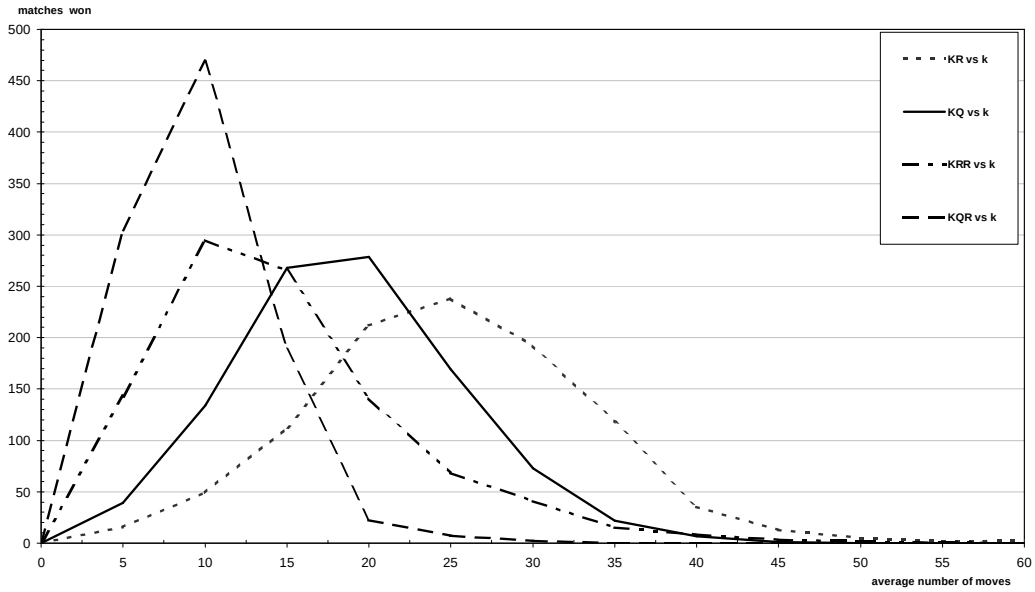


Figura 6.2: Confronto tra i test ottenuti con algoritmo di ricerca definitivo, sui finali di Torre, Donna, e misti

Diverso è stato l'esito considerando i finali in cui è richiesto l'uso di strategie probabilistiche e quindi è richiesto l'inserimento di un fattore stocastico nella funzione di valutazione.

Il grosso problema, in questo caso, risiede nel fatto che non è possibile utilizzare euristiche per la potatura dell'albero delle mosse di gioco, durante la ricerca. Ad esempio, l'uso della tecnica delle chiavi di Zobrist viene impedito dalla natura probabilistica che assume la valutazione. Se mantenessimo

l'ausilio della tabella di trasposizione, si rischierebbe di dare voti uguali a mosse che, grazie alla natura aleatoria introdotta, non hanno stesso valore.

Per questa ragione, un finale, come quello dei due alfieri, viene giocato dal programma in modo non completamente corretto. In esso non è stato possibile eliminare le euristiche sopraccitate per via della complessità e del numero delle mosse da analizzare, e si è tentato di eliminare la natura stocastica del problema, affidandosi al meccanismo che evita di giocare mosse recentemente giocate, già implementato nel motore principale, in modo da assicurarsi che due mosse con egual valutazione siano giocate una dietro l'altra. Chiaramente questo è un aspetto che merita ulteriori approfondimenti, la cui soluzione risulta necessaria nel caso si voglia implementare un algoritmo in grado di giocare l'intero gioco del Kriegspiel.

È stato invece possibile sfruttare l'uso di un euristica che valuta metaposizioni assegnando loro un certo punteggio casuale, in modo da accondiscendere alla natura stocastica del problema, nel caso del finale di Pedone. In questa situazione la struttura semplice del gioco ha permesso l'eliminazione delle tabelle di trasposizioni dalla ricerca nell'albero. Il risultato ottenuto è stato buono, come mostrato in figura 6.3 in cui sono messi a confronto i due test, nel caso del finale di Pedone e nel caso del finale con due alfieri.

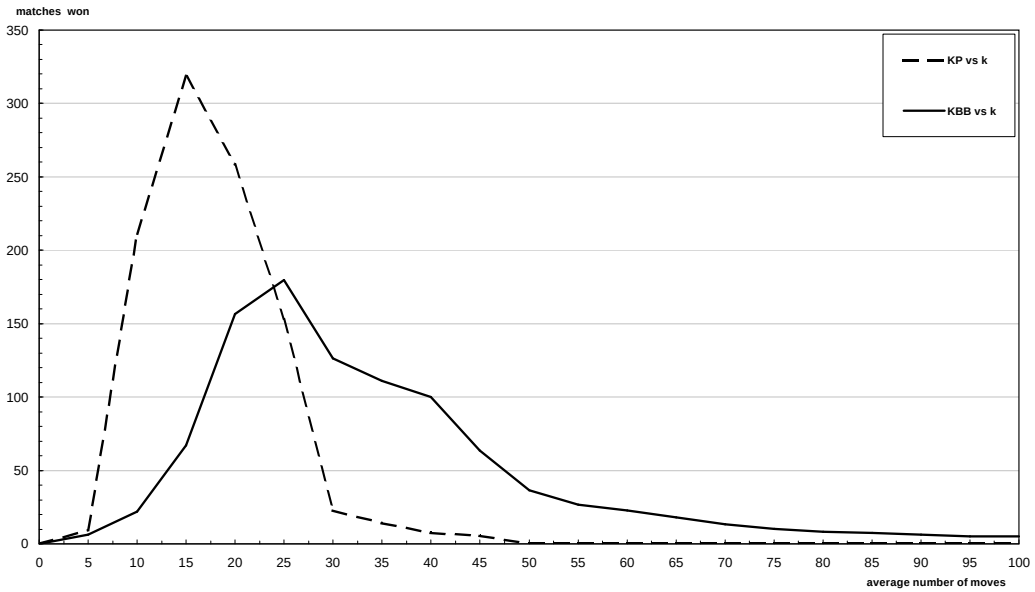


Figura 6.3: Confronto tra i test ottenuti con algoritmo di ricerca definitivo, sui finali di Pedone e con due alfieri

Il tentativo di estendere l'algoritmo, descritto per il finale con due alfieri, al caso del finale con un alfiere ed il cavallo ha fatto emergere alcune proble-

matiche circa il raggiungimento del "progresso" in fase di gioco, ossia circa la fase di avanzamento del Bianco nel ridurre l'incertezza sull'avversario. Il finale di alfiere e cavallo, affrontato già in [14], ove viene indicata una procedura probabilistica simile a quella per i due alfieri, si rivela più complesso, poiché la profondità di ricerca sull'albero delle mosse, pari a 2 o 3 livelli, nel caso dei movimenti del cavallo non risulta sufficiente; in effetti il cavallo muove compiendo "salti" ed il passaggio a celle adiacenti non è diretto. La funzione di valutazione utilizzata nel caso degli alfieri deve quindi essere modificata considerando il caso specifico in cui un alfiere viene accoppiato ad un cavallo contro il singolo re nero.

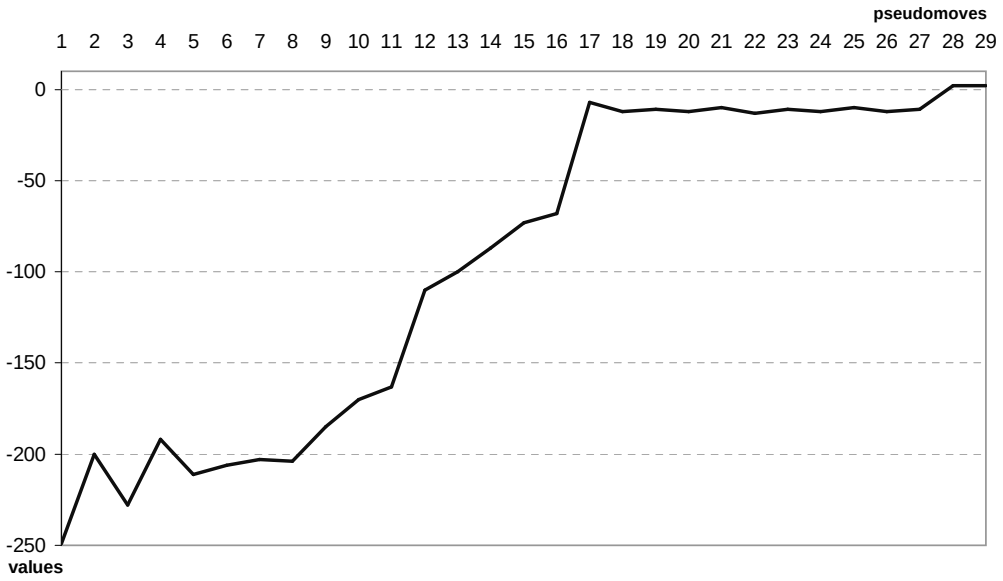


Figura 6.4: Andamento delle valutazioni assegnate alle metaposizioni percorse durante una partita nel finale di Donna

Per analizzare la capacità di "progresso" del giocatore artificiale è utile mostrare graficamente l'andamento dei valori restituiti dalla funzione di valutazione per ogni metaposizione raggiunta dopo i tentativi pseudolegali del Bianco. In figura 6.4 è mostrato un grafico che descrive lo sviluppo di una partita nel finale di Donna iniziando dalla posizione che vede i pezzi localizzati come segue: ♔a1, ♕a8, ♚d1. Essa è stata vinta in 24 mosse, ma ha visto un numero di pseudomosse pari a 29. Il grafico, quindi, mostra, al crescere delle pseudomosse, il valore delle posizioni raggiunte. Si vede facilmente come tale valore aumenti gradualmente durante lo sviluppo della partita. Sebbene in alcuni punti il valore diminuisca, si può notare come, approssimativamente, l'andamento complessivo presenti una tendenza crescente.

Allo stesso modo, in figura 6.5 è riportato un grafico analogo nel caso del

finale con due alfieri, partendo dalla posizione in cui si ha ♔c2, ♗a2, ♗g7, ♚c7. La caratteristica crescente è ben rilevabile, sebbene il numero di punti in cui tale tendenza viene meno sia più alto. Si può notare come il numero di tentativi pseudolegali necessario alla vittoria sia sensibilmente maggiore, è pari infatti a 76; conseguentemente la velocità di convergenza a posizioni buone per lo scacco è minore.

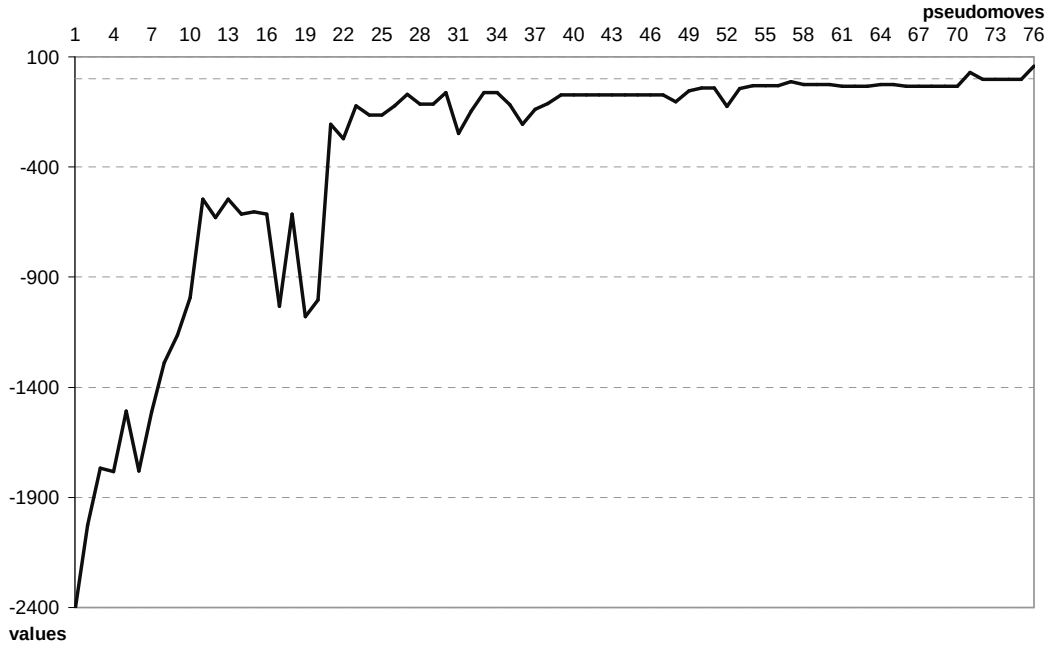


Figura 6.5: Andamento delle valutazioni assegnate alle metaposizioni percorse durante una partita nel finale con due alfieri

Utilizzando la stessa funzione di valutazione adottata per i due alfieri nel finale di cavallo, come detto, i risultati non sono stati ugualmente soddisfacenti. Il 50% circa delle partite viene perso, perché un pezzo viene mangiato o per mancanza di "progresso"; il restante 50% viene vinto, ma non si ha "progresso" durante lo sviluppo del gioco, bensì il giocatore artificiale incorre in situazioni di gioco favorevoli, sfruttandole nel migliore dei modi. Questo fa pensare che aggiustando adeguatamente la funzione di valutazione anche il finale con alfiere e cavallo possa essere risolto, ma è richiesto diversificare le valutazioni degli alfieri a seconda del finale in cui vengono coinvolti.

In figura 6.6 è mostrato il grafico riferito ad un finale con alfiere e cavallo vinto dal bianco in 111 tentativi, partendo dalla posizione in cui si ha ♔f2, ♗g4, ♘d3, ♚f8. È facile osservare come, nelle prime 20 pseuodomosse, non ci sia alcuna tendenza crescente, quindi alcun "progresso", mentre si

verifichi un salto considerevole circa al ventiseiesimo tentativo, in cui probabilmente il re bianco ha mosso ricevendo risposta illegale. Se ciò non fosse accaduto, probabilmente la partita sarebbe continuata invano. Dopo questo passaggio però, il valore delle posizioni percorse resta invariato per circa 80 pseudomosse, ancora una volta non si ha alcuna tendenza crescente.

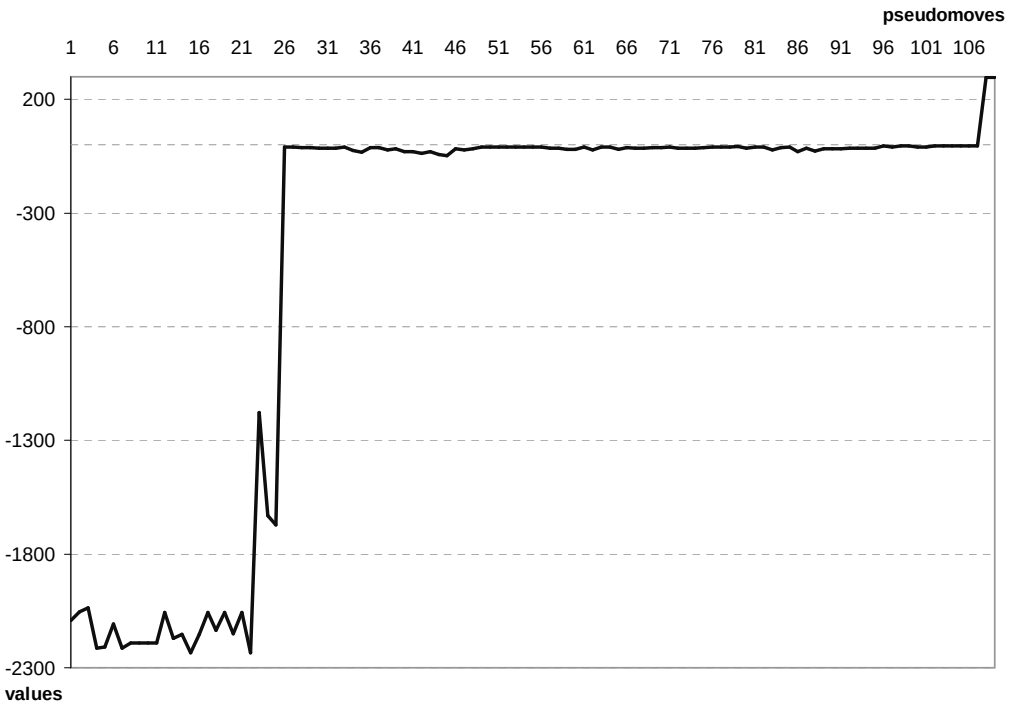


Figura 6.6: Andamento delle valutazioni assegnate alle metaposizioni percorse durante una partita nel finale con alfiere e cavallo

6.1 Estensione ad altre situazioni di gioco

Il programma implementato e le discussioni teoriche presentati, possono essere estesi a situazioni di gioco più complesse. Ad esempio è possibile considerare finali di partita in cui il giocatore Nero dispone non solo del re, ma anche di qualche pezzo importante.

Potrebbe essere interessante formulare una funzione di valutazione per un finale in cui il Bianco dispone di due torri e il Nero solo di una torre.

La prima difficoltà consiste nel costruire efficientemente un gestore di metaposizioni che riesca a calcolare nel più breve tempo possibile le varie metaposizioni raggiunte dalle metamosse avversarie. Questo compito è più complesso in questo caso, che nel finale con un solo re avversario. Se, ad

esempio, la torre bianca muove e riceve la risposta "scacco" dall'arbitro, ciò significa che o lei, o l'altra torre, ha dato scacco. Se invece, la risposta è "silente", non possiamo considerare che nella riga o nella colonna della torre non vi sia il re nero, in quanto esso potrebbe essere coperto dalla torre nera.

Chiaramente questa osservazione è vera anche in altri finali, in cui il re nero dispone di almeno un pezzo ulteriore oltre al re.

La seconda difficoltà deriva proprio da questa considerazione, ovvero che l'incertezza dopo una mossa non diminuisce, se questa risulta "silente". In queste circostanze avranno molta importanza le mosse del re Bianco nel sondare la scacchiera, ma sarà necessario una difesa più stretta dei propri pezzi, che non saranno più completamente al sicuro, se difesi solo dal re.

Una volta risolto il caso di finali composti da più pezzi, sarà ipotizzabile estendere l'implementazione vista all'intero gioco. In questo caso la potatura euristica delle mosse errate dall'albero avrà un ruolo cruciale, in quanto la complessità numerica del problema diverrà imponente.

6.2 Lavori futuri

Sarà approfondita l'analisi di valutazioni aleatorie per mosse equiprobabili, in grado di gestire situazioni in cui è necessario l'uso di strategie stocastiche da parte dei giocatori. Questo comporterà lo studio approfondito dell'utilizzo di euristiche di potatura dell'albero delle mosse di gioco.

Quindi, viste le considerazioni fatte nella sezione precedente, possibili lavori futuri consisteranno nell'estendere il giocatore artificiale fino ad ora implementato per i casi di finali composti. In particolare si potrà procedere prendendo in considerazione dapprima finali più semplici, come torre contro alfiere (KR vs kb), torre contro cavallo (KR vs kn), regina contro torre (KQ vs kr); in seguito si continuerà con finali a più pezzi, come quello con due torri (KRR vs kr), con torre e cavallo contro torre (KRN vs kr), con due alfieri contro cavallo (KBB vs kn) e altri.

Successivamente si potranno analizzare situazioni di gioco più articolate, fino ad estensioni al gioco completo.

Ringraziamenti

Un sentito e sincero ringraziamento va al Prof. Paolo Ciancarini per il suo aiuto, la sua disponibilità e la sua competenza, che tanto entusiasmo mi hanno trasmesso e tanto mi hanno incoraggiato nello svolgimento del mio lavoro.

Bibliografia

[1] M. Bain. *Learning Logical Exceptions in Chess*. PhD thesis, Dept. of Statistics and Modelling Science, University of Strathclyde, Glasgow, Scotland, 1994.

[2] A. Bell. *The Machine Plays Chess?*, chapter 2. Pergamon, 1978.

[3] A. Bolognesi and P. Ciancarini. Computer Programming of Kriegspiel Ending: the case of KRK. In *Proc. 10th Conf. on Advances in Computer Games, to appear*, 2003.

[4] Jim Boyce. A Kriegspiel Endgame. In D. Klarner, editor, *The Mathematical Gardner*, pages 28–36. Prindle, Weber & Smith, 1981.

[5] I. Bratko. *Prolog Programming for Artificial Intelligence.* Addison Wesley, Wokingham, 1986.

[6] A. Bud, D. Albrecht, A. Nicholson, and I. Zukerman. Information-theoretic Advisors in Invisible Chess. In *Proc. Artificial Intelligence and Statistics 2001 (AISTATS 2001)*, pages 157–162, Florida, USA, 2001. Morgan Kaufman Publishers.

[7] A. Bud, D. Albrecht, A. Nicholson, and I. Zukerman. Playing "Invisible Chess" with Information-Theoretic Advisors. In *Proc. 2001 AAAI Spring Symposium on Game Theoretic and Decision Theoretic Agents*, pages 6–15, California, USA, 2001. American Association for Artificial Intelligence.

[8] P. Ciancarini, F. Dalla Libera, and F. Maran. Decision Making under Uncertainty: A Rational Approach to Kriegspiel. In J. van den Herik and J. Uiterwijk, editors, *Advances in Computer Chess 8*, pages 277–298. University of Limburg, Maastricht, The Netherlands, 1997.

[9] Paolo Ciancarini. La Scacchiera Invisibile: Introduzione al Kriegspiel, manuscript. 2003.

[10] M.R.B. Clarke. A Quantitative Study of King and Pawn against King. In M.R.B. Clarke, editor, *Proc. First Conf. on Advances in Computer Chess*, pages 108–118, Oxford, UK, 1977. Edinbourgh University Press.

[11] Roberto Convenevole and Francesco Bottone. *La storia di Risiko e l'anello mancante*. Novecento Giochi, Roma, 2002.

[12] N. Megiddo D. Koller and B. von Stengel. Fast algorithms for finding randomized strategies in game trees. *Proceedings of the 26th Annual ACM Symposium on the Theory of Computing*, pages 750–759, 1994.

[13] Thomas S. Ferguson. *Game Theory. Class notes for Math 167*. 2000.

[14] Tom Ferguson. Mate with Bishop and Knight in Kriegspiel. *Theoretical Computer Science*, 96:389–403, 1992.

[15] Tom Ferguson. Mate with two Bishops in Kriegspiel. Technical report, UCLA, 1995.

[16] D. Koller and A.J. Pfeffer. Generating and Solving Imperfect Information Games. *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1185–1192, August 1995.

[17] L. Lambertini. Teoria dei Giochi: Storia e Metodologia. 2000.

[18] M. Leoncini. Elementi di strategia negli scacchi. `http://digilander.libero.it/marioleoncini/strategia/ /copertina.htm`, 1992.

[19] M. Leoncini and R. Magari. *Manuale di Scacchi Eterodossi*. Tipografia Senese, Siena, 1980.

[20] David K. Levine. Economic and Game Theory. What is Game Theory?. `http://levine.sscnet.ucla.edu/general/whatis.htm`, 2002.

[21] David Li. *Kriegspiel. Chess under Uncertainty*. Premier Publishing, 1994.

[22] R. Lucchetti. *Di duelli, scacchi e dilemmi. La teoria matematica dei giochi*. Paravia Scriptorium, Torino, 2001.

[23] Roberto Magari. Scacchi e probabilità. In *Atti del Convegno: Matematica e scacchi. L'uso del Gioco degli Scacchi nella didattica della Matematica*, pages 59–66, Forlì, 1992.

[24] F. Maran. Razionalità Sostanziale e Procedurale in una Variante Eterodossa del Gioco degli Scacchi. Il Kriegspiel, 1993.

[25] T.A. Marsland. Computer chess and search. In S. Shapiro, editor, *Encyclopedia of Artificial Intelligence*. John Wiley, 1992.

[26] A. Matros. The invisible Chessboard or Meeting with Lloyd Shapley. *64-Chess Review, #10*, pages 58–59, 1999.

[27] D. Michie. King and Rook Against King: Historical Background and a Problem on the Infinite Board. In M.R.B. Clarke, editor, *Proc. First Conf. on Advances in Computer Chess*, pages 30–59, Oxford, UK, 1977. Edinbourgh University Press.

[28] B. Moreland. Computer Chess. `http://www.seanet.com/~brucemo/chess.htm`, 2002.

[29] Roger Myerson. Learning Game Theory from John Harsanyi, 2000.

[30] Fioravante Patrone. Corso online di Teoria dei Giochi 2000/01, 2001.

[31] D. Pritchard. *The Encyclopedia of Chess Variants*. Games & Puzzles Publications, 1994.

[32] S.J. Russel and P. Norvig. *Artificial Intelligence: A Modern Approach*, chapter 5. Prentice Hall, 1995.

[33] M. Sakuta. *Deterministic Solving of Problems with Uncertainty*. PhD thesis, Shizuoka University, Japan, 2001.

[34] M. Sakuta and H. Iida. Solving Kriegspiel-like Problems: Exploiting a Transposition Table. *ICCA Journal*, 23(4):218–229, 2000.

[35] Ulrich Schwalbe and Paul Walker. Zermelo and the Early History of Game Theory. 1997.

[36] Branislav L. Slantchev. *Games of Incomplete Information, I: Static Games and Bayesian Nash Equilibrium*. 2002.

[37] H. D. Swart. Any in action!, 2003.

[38] Paul Walker. An Outline of the History of Game Theory, 1995.

[39] J. D. Williams. Kriegsspiel rules at RAND. (Unpublished manuscript)., 1950.

[40] J. D. Williams. *The Compleat Strategyst*. 2nd Edition, McGraw-Hill, NewYork, 1996.

[41] A.L. Zobrist. A new hashing method with application for game playing. *ICCA Journal*, 13(2):69–73, 1970. Reprinted (1990).

Indice analitico