Università
Ca'Foscari
Venezia

Facolta' di Economia e Commercio

Corso di laurea in Economia Aziendale – Indirizzo quantitativo

Razionalitá sostanziale e procedurale in una variante eterodossa del gioco degli scacchi: il Kriegspiel

Relatori: Prof. Francescp Dalla Libera
Prof. Paolo Ciancarini

Laurando: Fabio Maran

Anno accademico: 1992/93

Alla memoria di mio Padre

Inesauribile fonte di incoraggiamento

Indice

Capitolo 3.

Kriegspiel e razionalità procedurale

Capitolo 4

Considerazioni conclusive

Appendice 1.

Descrizione e regole del Kriegspiel

Appendice 2.

Teoria dei giochi e definizioni di base

Introduzione

Gli scacchi eterodossi sono una famiglia di giochi strettamente imparentati con i più noti scacchi "tradizionali". Essi derivano da modifiche più o meno ampie alle regole degli scacchi ortodossi. Si contano centinaia di varianti eterodosse, alcune ottenute semplicemente variando le regole di movimento dei pezzi, oppure le dimensioni o le proprietà della scacchiera; altre varianti invece sconvolgono completamente la natura del gioco, tra queste abbiamo il kriegspiel : in esso i giocatori non sono a conoscenza della disposizione dei pezzi avversari, ma cercano di inferirla nel corso della partita per mezzo di informazioni parziali che vengono date da un arbitro.
In questa dissertazione verrà eseguito uno studio razionale del gioco, di cui molto poco si sa.
Il problema di eseguire uno studio di questo tipo trova un primo ostacolo nella definizione di razionalità che vogliamo utilizzare; a tale proposito vengono utilizzati due differenti concetti che si rifanno ad una dicotomia introdotta dal premio Nobel per l'economia H.A. Simon: la razionalità sostanziale e la razionalità procedurale.
Il percorso seguito in questa ricerca si rifà a quello gia noto eseguito per uno studio razionale del gioco degli scacchi.
Infatti, dal punto di vista "sostanzialmente razionale" la teoria classica dei giochi ha da tempo esaurito i suoi interessi verso gli scacchi "ortodossi", classificandoli tra la categoria dei giochi ad informazione perfetta e stabilendo che questo gioco è dotato di uno o più punti di equilibrio (Teorema di Zermelo), anche se è impossibile stabilire quale sia la strategia

corrispondente a tali punti dato l'immenso ordine di grandezza dell'albero di gioco.
Simon in un recente articolo[1] *sottolinea come l'interesse verso gli scacchi da parte della teoria dei giochi possa essere rivalutato, studiando gli scacchi in un ottica "proceduralmente razionale" e coglie l'occasione per proporre una teoria dei giochi con razionalità procedurale, progetto in cui parte della ricerca è interessata. (Binmore, Rubinstein, Kreps ecc.).*
Come punto di partenza vi è lo studio degli algoritmi scacchistici in grado di condurre autonomamente una partita a formulando le strategie che ritengono più opportune. Da tempo l' Intelligenza Artificiale si e interessata a questa problematica: I programmi che giocano a scacchi possono essere visti come dei modelli di razionalità procedurale.
A differenza del gioco degli scacchi, nel kriegspiel l'approccio con razionalità sostanziale è sembrato più fruttuoso. La teoria classica dei giochi non si è limitata ad un apporto esclusivamente descrittivo (che è pur sempre indispensabile in quanto fornisce un linguaggio in grado di comprendere meglio le caratteristiche del problema che si affronta), ma ha fornito degli utili risultati applicando due modelli studiati negli anni '50 (Giochi stocastici e giochi ricorsivi) per la soluzione di alcuni semplici problemi finali.
I risultati che illustreremo nei capitoli successivi rappresentano un parziale successo della filosofia di approccio con razionalità sostanziale, la quale però si trova ancor più inadeguata che nel gioco degli scacchi, nel momento

[1] *Simon & Schaeffer "The game of chess" in Aumann Hart "Handbook of game theory" North Holland(1992).*

in cui vengono posti problemi più complessi che si presentano nel kriegspiel.
Assume allora maggior rilievo uno studio del gioco con razionalità procedurale, che ha come sbocco la modellizzazione di un giocatore artificiale in grado di affrontare con successo un particolare problema finale.
Dall' esperienza nella programmazione di questo giocatore (del tutto nuova) emerge come la conoscenza sia un elemento insostituibile che caratterizza la validità dell'approccio proceduralmente razionale. La conoscenza viene acquisita in diversi modi e come si vedrà, buona parte di essa può essere ottenuta dall'approccio con razionalità sostanziale, che dunque non va accantonato solamente perchè in grado di dare soluzioni normative a problemi troppo semplici.
In particolare il presente lavoro è così strutturato :
Nel capitolo 1 saranno introdotte le nozioni di razionalità, con particolare riguardo per la distinzione tra razionalità sostanziale e procedurale. Verrà quindi ripercorso il cammino seguito nella programmazione di giocatori artificiali di scacchi.
Il capitolo 2 entra nel vivo dell'argomento cercando di classificare il gioco all'interno delle categorie indicate dalla teoria dei giochi, verranno descritti in particolare i modelli dei giochi stocastici e ricorsivi e alcuni problemi risolti efficientemente proprio per mezzo di questi modelli.
Il capitolo 3 illustra i risultati di un approccio con razionalità procedurale, in particolare il risulato concreto di questo è un programma sviluppato in

ambiente PROLOG in grado di gestire autonomamente il finale re + pedone contro re[2].

Nelle appendici 1 e 2 vengono rispettivamente riportate la descrizione dettagliata del kriegspiel e le nozioni di base di teoria dei giochi utilizzate nel corso della dissertazione.

[2] *Il programma è contenuto nel dischetto allegato.*

Capitolo 1

Nozioni di razionalità

1. Razionalità e modelli di comportamento razionale.

In una formulazione generale di H.A. Simon (*Simon 1964*), il termine "razionalità" denota un tipo di comportamento il quale :

(A) è indicato al raggiungimento di obiettivi dati ;
(B) entro i limiti imposti da certi vincoli e condizioni.

Per comprendere come possano esistere diverse concezioni di razionalità occorre specificare quali siano alcune possibili interpretazioni di (A) e (B).

(A).1 L'obiettivo può essere quello di massimizzare il valore atteso di una funzione di utilità. Tale obiettivo è uno dei postulati fondamentali dell' analisi economica tradizionale; in tal modo il "consumatore razionale" massimizza la propria utilità attesa, mentre “l' imprenditore razionale" massimizza il proprio profitto. In questo contesto si parlerà di "ottimizzazione".

(A).2 L' obiettivo può essere quello di "soddisfare" alcuni criteri (quello che gli psicologi definiscono "ottenimento del livello di aspirazione").

(B).1 I vincoli e le condizioni possono essere delle caratteristiche oggettive esterne all'organismo che deve decidere , in tal caso si parlerà di "razionalità oggettiva" .

(B).2 I vincoli e le condizioni possono essere delle percezioni soggettive dell' individuo ed allora si parlerà di "razionalità soggettiva" o "limitata".

Tale definizione ha il pregio di essere abbastanza elastica da comprendere le varie accezioni di razionalità che sono state adottate da varie discipline.

I modelli di comportamento razionale sono costituiti da alcuni o da tutti i seguenti elementi (*Simon 1955*):

1) Un insieme di alternative di comportamento (o di decisione), rappresentate in modo matematico da un insieme (di punti) $\mathbf{A}$. Volendo può essere considerato solo $\mathbf{A}' \subset \mathbf{A}$, cioè un sottoinsieme di alternative prese in considerazione dal soggetto che deve decidere.

   Ogni singola alternativa di comportamento verrà indicata con $a \in \mathbf{A}$.
2) Un insieme $\boldsymbol{S}$ di risultati delle scelte. In cui ogni possibile risultato viene indicato con $s \in \mathbf{S}$
3) Una funzione di utilità $u(s)$ che rappresenta il valore che il soggetto dà ai risultati delle scelte.
4) Informazioni su quale sarà il risultato s in corrispondenza di ogni particolare scelta di comportamento $a \in \mathbf{A}$ (o $\mathbf{A}'$). Tali informazioni vengono rappresentate da una funzione che associa ad ogni alternativa di comportamento $a \in \mathbf{A}$ (o $\mathbf{A}'$) un risultato s_a. L'informazione potrà

essere incompleta, nel senso che non è possibile associare ad ogni alternativa di comportamento un unico risultato , in tal caso si dovrà assegnare ad ogni scelta $a \in \mathbf{A}$ (o $\mathbf{A}'$) un sottoinsieme $\mathbf{S}a \subset \mathbf{S}$ di possibili risultati.

5) Una distribuzione di probabilità che, qualora l'informazione sia incompleta, associa ad ogni $s \in \mathbf{S}a$ una probabilità $\mathbf{P}a(s)$, che indica la probabilità che si verifichi il risultato s qualora sia stata scelta la alternativa a.

Premesso questo, possiamo definire delle procedure di scelta razionale strutturate secondo i paradigmi della teoria economica usuale e la teoria classica dei giochi[3] (*Arrow 1951*).

- *Regola della certezza.* Data l'informazione che ogni $a \in \mathbf{A}$ (o $\mathbf{A}'$) ha come risultato uno specifico $s_a \in \mathbf{S}$, scegliere l'alternativa di comportamento che ha come risultato l' utilità maggiore.
  Sceglierò un a' tale che $u(s_{a'}) = \underset{a \in \mathbf{A}}{\mathrm{Max}}\ u(s_a)$
- *Regola probabilistica.* Massimizzare il valore atteso $E(s)$ per la distribuzione di probabilità $\mathbf{P}a(s)$ che si suppone nota.
  Sceglierò un a' tale che $u(a') = \underset{a \in \mathbf{A}}{\mathrm{Max}} \sum_{s \in \mathrm{Sa}} u(s)\ \mathbf{P}a(s)$
- *Regola del maxmin.* Supponendo che qualunque sia l'alternativa di scelta ne seguirà il peggior risultato possibile (si realizzerà il più piccolo $u(s)$ per $s \in \mathbf{S}a$). Si scelga quell'alternativa per cui questo guadagno peggiore è il più grande possibile.

[3] La definizione di teoria classica dei giochi che postula un comportamento illimitatamente razionale viene suggerita dal Simon (*Simon & Schaeffer 1992*), per distinguerla da un possibile nuovo approccio che postuli un comportamento *limitatamente razionale* da parte dei giocatori.

Sceglierò un a' tale che $u(a') = \underset{a\in\mathbf{A}}{\text{Max}} \; \underset{s\in\mathbf{S}a}{\text{Min}} \; u(s)$

E' immediatamente chiaro che i paradigmi classici della scelta razionale richiedano, nella maggior parte delle situazioni reali, capacità computazionali improponibili al soggetto che sceglie.
Il decisore deve essere in grado di associare ad ogni alternativa un guadagno definito, o addirittura una distribuzione di probabilità sui possibili guadagni. Inoltre i guadagni devono essere completamente confrontabili, nel senso che è sempre possibile stabilire se un risultato è migliore uguale o peggiore di ogni altro.
H.A. Simon (*Simon 1955*) introduce delle modifiche alle assunzioni dei modelli classici di razionalità, che egli stesso definisce *semplificazioni essenziali*, il risultato di tali rettifiche dà origine ad un "*modello comportamentale di scelta razionale*", che ha il pregio di descrivere con maggiore corrispondenza alla realtà, situazioni in cui il "decisore razionale" deve affrontare problemi complessi.
Le modifiche in questione sono :

<u>Semplificazione della funzione di utilità.</u>
E' possibile ipotizzare che l'individuo assegni ad ogni risultato, uno dei due valori (1;0), interpretandoli come "soddisfacente" e "non soddisfacente". Lo spartiacque tra questi due valori, è quello che gli psicologi definiscono: "livello di aspirazione".
Questo sistema di valutazione diventa verosimile nel momento in cui il soggetto che deve scegliere non è in grado di valutare tutte le alternative contemporaneamente, ma si limita ad esplorarle sequenzialmente; in tal

modo il processo di decisione razionale ha termine nel momento in cui il decisore trova un' alternativa soddisfacente. Su queste basi, Simon propone una nuova procedura di scelta razionale caratterizzata dai seguenti passi :

i) Si cerchi un insieme di risultati possibili (un sottoinsieme $\mathbf{S}' \in \mathbf{S}$) tale che il pay-off sia soddisfacente ($u(s) = 1$) per ogni $s \in \mathbf{S}'$.
ii) Si cerchi un alternativa comportamentale soddisfacente $a \in \mathbf{A}'$ i cui possibili risultati appartengano tutti ad $\mathbf{S}'$ (tale che ad a corrisponda un insieme $\mathbf{S}a \subseteq \mathbf{S}'$)[4].

E' importante sottolineare che la procedura non garantisce nè l'esistenza nè l'unicità di un a soddisfacente. Qualora ad esempio, la ricerca al punto ii) non porti ad un' alternativa comportamentale soddisfacente è proponibile una modifica del livello di aspirazione, che si abbasserà qualora la ricerca di una soluzione soddisfacente si rivelasse ardua e viceversa. Alternativamente alla modifica del livello di aspirazione si ha l'espansione dell'insieme delle possibili alternative prese in considerazione (**A**').

<u>Raccolta di informazioni :</u>

E' verosimile ipotizzare che il decisore non sappia con certezza quale sia la corrispondenza di **A** con i sottoinsiemi di **S**, cioè i possibili risultati scaturibili da ogni alternativa. Nella peggiore delle ipotesi dalla scelta *a* potrebbero scaturire tutti i risultati dell' insieme **S**.

Possono allora essere introdotti nel modello dei momenti di raccolta delle informazioni che danno luogo alla formazione di una corrispondenza più precisa tra i vari elementi di **A** e i sottoinsiemi di **S**. Se il processo di

[4] In altri termini: ad *a* devono corrispondere solamente risultati soddisfacenti.

raccolta delle informazioni ha un costo, sarà opportuno calcolare fino a che punto convenga perfezionare la funzione di corrispondenza tra gli insiemi.
Secondo Simon *(Simon 1955)* il processo decisionale appena descritto è uno dei metodi principali impiegati dagli scacchisti per scegliere una mossa a metà e alla fine della partita[5].
Infatti: sia **A** l'insieme delle mosse disponibile per il bianco alla ventesima mossa, sia **S** l'insieme di posizioni che potrebbero essere raggiunte, diciamo, alla trentesima mossa. Sia **S'** un possibile sottoinsieme di **S** composto da posizioni chiaramente vincenti[6]. Da una conoscenza inizialmente molto approssimativa della relazione che lega ogni $a \in \mathbf{A}$ ad **S**, il bianco sceglie sperimentalmente una mossa *a* che (se il nero gioca in un certo modo) conduce all'insieme di risultati **S'**. In questo contesto, il processo di raccolta delle informazioni è costituito dal perfezionamento della relazione che lega ogni $a \in \mathbf{A}$ ad **S** : quanto più la relazione è precisa maggiori sono le probabilità di giocare una mossa oggettivamente vincente.

Ordinamento parziale dei pay-off :

La teoria classica richiede una funzione scalare di *pay-off*, cioè un'ordinamento completo dei *pay-off (Simon 1955)*.
Simon mette in evidenza che in realtà, nella valutazione degli esiti di una scelta, potremmo trovarci di fronte ad una funzione vettoriale $\mathbf{V}(s)$, dove **V** ha come componenti $v_1, v_2, ..., v_n$.

---

[5] La fase di apertura viene affrontata con un metodo diverso, prevalentemente mnemonico, poichè le prime mosse di una partita a scacchi sono state studiate e catalogate.
[6] Il fatto che **S'** sia costituito da posizioni chiaramente vincenti, dipende dal livello di aspirazione del giocatore, infatti se il livello di aspirazione fosse più basso, il giocatore potrebbe anche "accontentarsi" si un **S'** costituito da posizioni almeno pari

Ecco degli esempi in cui è necessario introdurre una funzione di *pay-off* vettoriale.

La decisione viene presa da un gruppo di *n* persone, ciascuna delle quali esprime un proprio giudizio ; nel complesso la scelta sarà valutata da un vettore con *n* elementi.

L' individuo deve considerare aspetti di un pay-off non paragonabili tra loro, ad esempio nella scelta di un automobile deve confrontare diverse caratteristiche tecniche come la velocità, il comfort, il prezzo ecc.. che non possono essere ricondotte ad un unico valore.

In corrispondenza della decisione *a* esiste un insieme di *n* alternative possibili, ***S****a*, in questo caso si potrà sostituire all'insieme ***S****a* una sola conseguenza, configurata però come un vettore di *pay-off* con *n* elementi.

Se utilizziamo la nozione semplificata di utilità potremmo considerare come soddisfacente il vettore di *pay-off* per il quale $v_i = 1$ per ogni i (oppure $v_i \geq K_i$ per ogni i dove K_i sono i livelli di aspirazione per ogni elemento del vettore).

Nei paragrafi successivi, vedremo come le *semplificazioni essenziali* introdotte da Simon nei modelli di scelta razionale vengano adottate nel contesto decisionale del gioco degli scacchi.

2. Razionalità sostanziale e procedurale

La distinzione tra "*Razionalità sostanziale*" e "*Razionalita procedurale*" è stata introdotta da Simon *(Simon 1976,1978)*.

Un comportamento si dice *sostanzialmente razionale* quando è teso alla ricerca della migliore azione in assoluto da intraprendere per risolvere un problema. E' un dato di fatto che la teoria economica tradizionale si sia interessata quasi esclusivamente a *quali* decisioni debbano essere prese piuttosto di *come* debbano essere prese. In particolare, queste sono le assunzioni fondamentali dell'analisi economica classica:

1) L'attore economico ha uno specifico obiettivo (massimizzare l'utilità o il profitto)
2) L'attore economico è "*sostanzialmente razionale*".

Partendo da queste due assunzioni, l'analisi economica viene condotta per mezzo dei tradizionali strumenti matematici, che sono il calcolo differenziale, la programmazione lineare ecc.

La *razionalità procedurale* è invece interessata alle procedure che permettono di trovare una soluzione accettabile ad un dato problema, tenendo conto non solo dell'obiettivo e dei vincoli oggettivi, ma anche della conoscenza e dei limiti delle capacità computazionali del soggetto decisore. Simon afferma che un comportamento è *proceduralmente razionale* quando è il risultato di una deliberazione appropriata.

Quanto più la complessità del problema posto al decisore mette in evidenza i limiti computazionali di quest'ultimo (sia esso una macchina o una mente umana), emerge la necessità di utilizzare un approccio procedurale che cerchi la razionalità nel modo in cui si affronta il problema, e non nella sua soluzione: l'utopia di un ottimizzazione viene abbandonata verso la ricerca di una soluzione soddisfacente.

Varie discipline dal secondo dopoguerra a questa parte hanno dedicato il loro interesse alla razionalità procedurale, in particolare la *Ricerca operativa*, l' *Intelligenza artificiale*, le *Scienze cognitive*.

Nella ricerca operativa, ad esempio nel momento in cui ci si trova ad affrontare un problema NP-completo di dimensioni tali per cui la ricerca di una soluzione ottimale richiederebbe tempi astronomici anche per un potentissimo calcolatore , l'interesse si sposta verso una procedura che dia una soluzione *soddisfacente* in un tempo ragionevole; in questi contesti assume un certo rilievo la *Teoria della complessità computazionale*, una disciplina recente che si pone l'obiettivo di misurare l'efficienza computazionale degli algoritmi, quindi la loro attitudine a ricercare una soluzione in modo proceduralmente razionale.

Intelligenza Artificiale e *Simulazione cognitiva* sono due discipline molto simili che molto spesso si intrecciano, si può addirittura affermare che non esiste un chiaro confine tra le due. L' intelligenza artificiale cerca di *far eseguire ai calcolatori compiti, in cui per il momento gli esseri umani sono più abili*[7]. Anche la simulazione cognitiva cerca di far svolgere a delle macchine compiti complessi, la differenza tra le due discipline è che l' intelligenza artificiale non usa necessariamente tecniche affini ai processi realmente utilizzati dalla mente umana, mentre invece la simulazione

---

[7] La definizione è introdotta da E. Rich, *Artificial Intelligence* - Mc Graw Hill (1983).

cognitiva mantiene un legame più stretto con la psicologia e con lo studio dei processi cognitivi umani.

La stessa osservazione che è stata fatta per la teoria economica tradizionale, cioè di essersi dedicata quasi esclusivamente alla razionalità sostanziale, può essere fatta anche per la teoria dei giochi che ha concentrato la propria attenzione sul quesito "*Cosa decidono i giocatori*", piuttosto che "*Come i giocatori decidono cosa fare*" *(Binmore 1988)*. La teoria dei giochi si è quasi esclusivamente occupata del comportamento interattivo dell' "*Homo rationalis*" inteso come fautore di una razionalità sostanziale. Alcune recenti direzioni di ricerca *(Kreps 1990)* cercano di introdurre i concetti di razionalità limitata e procedurale nella teoria dei giochi. In questo contesto il gioco degli scacchi, come vedremo nei paragrafi successivi, assume un certo interesse[8].

Nel corso di questo lavoro verrà introdotto il gioco Kriegspiel, si tenterà di approcciare ad esso facendo uso dei paradigmi della razionalità sostanziale e della razionalità procedurale.

[8] "*The only nontrivial theory of chess is a theory of procedural rationality in choosing moves*". (*Simon & Schaeffer* 1992).

3. Pregi e limiti della metafora scacchistica per rappresentare un contesto decisionale reale.

Il gioco degli scacchi rappresenta un fertile terreno nel quale possono essere messi in discussione i paradigmi della razionalità sostanziale, mentre viene messo in risalto un tipo di approccio con razionalità limitata, che cerca di dare delle indicazioni descrittive e normative su *come* l' individuo decida *cosa* fare.

La *teoria dei giochi classica*, la quale postula un comportamento illimitatamente razionale da parte dell'individuo, non ha mai approfondito le problematiche del gioco degli scacchi , esso infatti appartiene alla categoria dei giochi ad informazione perfetta, i quali possiedono sempre una soluzione di equilibrio in una strategia pura *(Teorema di Zermelo)* . Il gioco sarebbe allora banale, in quanto ipotizzando un individuo in grado di valutare i risultati di ogni propria strategia, attraverso un procedimento di *backward induction*, egli giocherebbe sempre la mossa migliore in senso assoluto.

H.A. Simon fu uno dei primi studiosi a rendersi conto dell'importanza del "*microcosmo scacchistico*", per modellizzare situazioni in cui un approccio con razionalità in senso tradizionale, non avrebbe dato alcuna indicazione di tipo pratico, e inoltre non sarebbe nemmeno stata in grado di spiegare l'effettivo comportamento umano di fronte al problema.

La critica parte dalla constatazione di impossibilità da parte del soggetto decisore di considerare contemporaneamente tutte le conseguenze che derivano da ogni sua potenziale scelta ; detto in altri termini, l'albero degli scacchi, secondo stime di Simon, è costituito da un numero di nodi terminali compreso tra 10^{40} e 10^{120}, un ordine di grandezza intrattabile anche per un moderno elaboratore.

Il "deficit computazionale" da parte del decisore, origina un particolare tipo di incertezza, la quale però va ben distinta dall'incertezza in senso tradizionale, che è determinata da una mancanza di conoscenza circa le conseguenze rilevanti di ogni scelta, o dei contesti (stati della natura) in cui sono inserite.

Situazioni in un ambiente di tipo scacchistico sono state definite "complicate" *(Faccipieri 1989)*, poichè un aumento delle capacità computazionali del soggetto decisore, sarebbe in grado di risolvere questo tipo di problema.

Von Neumann e Morgenstern sostenevano infatti che qualunque gioco ad informazione perfetta sarebbe banale con una capacità computazionale illimitata da parte dei giocatori.

Alla luce di queste considerazioni, sembra che la realtà sia caratterizzata da due tipi di incertezza, l' una strettamente correlata alla complicazione insita nei calcoli e nelle procedure necessarie per giungere alla soluzione del problema (situazione complicata).

La seconda, invece, che deriva dall'incapacità da parte del decisore di individuare lo *stato della natura* che corrisponde al suo contesto decisionale (situazione complessa).

Il gioco degli scacchi mette in risalto il primo tipo di incertezza, e gli algoritmi che sono stati progettati per affrontarla costituiscono un ottimo esempio di modellizzazione di razionalità limitata o procedurale, proprio a riguardo di questo, in un recente articolo scritto da Simon in collaborazione con Schaeffer *(Simon & Schaeffer 1992)*, gli autori dopo aver descritto una rassegna dell'evoluzione degli algoritmi per computer che giocano a scacchi, affermano : "*What is emerging, therefore, from research on games like chess, is a computational theory of games : a theory of what is reasonable to do when it is impossible to determine what is best, a theory of bounded rationality".*

Osserveremo nei capitoli successivi come il microcosmo del Kriegspiel sembri essere più adatto di quello scacchistico per modellizzare un contesto di decisione reale, in quanto a differenza degli scacchi il Kriegspiel comprende entrembe le categorie di incertezza sopre delineate: la complessità e la complicazione.

4. Scacchi e scienze cognitive

Le ricerche psicologiche sul modo in cui il giocatore di scacchi organizza il proprio pensiero al fine di decidere una mossa, costituiscono il punto di partenza di un approccio con razionalità procedurale al gioco degli scacchi.

Il primo studioso ad interessarsi di ciò fu lo psicologo olandese *A. De Groot (1948)*[9] il quale condusse un'esprimento che consisteva nel far decidere ad alcuni forti giocatori e ad altri meno forti la mossa più opportuna, ragionando per così dire "a voce alta".

Il risultato fu sorprendente in quanto tra gli scacchisti sottoposti al test era praticamente impossibile distinguere un giocatore forte da uno debole, entrambi, infatti, valutavano una media di 44 posizioni, visitando 16 rami con una profondità da 1 a 4 mosse. Dalla ricerca emergevano due importanti risultati : in primo luogo la capacità computazionale dello scacchista non è l'elemento principale che discrimina un buon giocatore da una *schiappa*, inoltre la capacità computazionale umana è di gran lunga inferiore a quella di un qualunque computer[10].

Un secondo importante risultato della ricerca psicologica stabiliva che quando un giocatore esperto osserva una posizione sulla scacchiera, egli non vede solamente dei pezzi sulle caselle come farebbe un comune principiante, ma cerca di far confluire la posizione che sta analizzando, in contesti a lui noti e studiati in precedenza (*Chunks*), i quali suggeriscono eventuali pericoli o opportunità insite nella posizione raggiunta.

[9] Per le citazioni bibliografiche di questo paragrafo cfr. *Moates 1980* e *Simon* 1973

[10] Attualmente le macchine più potenti espolorano fino a due milioni di posizioni al secondo.

Alcune stime (*Simon, Gilmartin 1973*) affermano che un forte giocatore professionista può accumulare nella sua memoria fino a 50.000 *contesti* o *chunks* , frutto di almeno un decennio di studio. Tale affermazione venne dimostrata da un noto esperimento in ambito scacchistico, (*De Groot 1965*) nel quale vennero mostrate a degli individui, scacchisti e non, delle posizioni con un numero di circa 25 pezzi sulla scacchiera per un lasso di tempo da 5 a 10 secondi. Mentre i giocatori riuscivano a ricordare la collocazione del 90% dei pezzi, i non giocatori al più riuscivano a ricordare al più 5 o 6 pezzi.
Questo dimostra che i giocatori esperti e quindi dotati di un notevole bagaglio di conoscenze erano in grado di memorizzare immediatamente una posizione perchè la associavano ad una di quelle immagazzinate nella loro memoria.
In un successivo esperimento (*Chase & Simon 1973*), vennero attuate alcune modifiche; le posizioni mostrate non erano verosimili, cioè tratte da partite giocate nella realtà, ma erano completamente casuali : quindi non erano presenti nel "bagaglio culturale" del giocatore esperto. In questo caso sia gli scacchisti che i non giocatori ricordavano la posizione di solo 5 o 6 pezzi.
Gli studi condotti in questo ambito, suggeriscono che il decisore faccia fronte all'incertezza computazionale facendo ricorso a modelli impliciti acquisiti nel corso della propria esperienza, piuttosto che aumentando le capacità di calcolo.

5.1 Scacchi e intelligenza artificiale

Nel 1950 C.Shannon *(Shannon 1950)* ideò un primo algoritmo in grado di giocare a scacchi, i cui principi rappresentano la struttura portante dei programmi susseguitisi nei successivi 40 anni. L'algoritmo di Shannon rappresenta un compromesso tra la procedura *minmax* suggerita dalla teoria dei giochi e l'effettivo approccio umano al gioco il quale lungi dal poter effettuare una visita esaustiva dei nodi dell'albero del gioco, deve far ricorso ad altri espedienti, primo fra i quali la conoscenza di euristiche che sintetizzano la teoria e la pratica del gioco.

Sulla base della conoscenza teorica, Shannon proponeva la costruzione di una *funzione di valutazione* in grado di dare un punteggio ad ogni nodo terminale dell' albero del gioco che poteva essere visitato nei limiti della capacità computazionale della macchina o dell'uomo che avesse voluto giocare tentando di razionalizzare il proprio pensiero seguendo le indicazioni dell'algoritmo. Operando in questo modo si sarebbe potuta effettuare una drastica semplificazione dell'albero di gioco, considerando solamente il sottoalbero (*search-tree*) ottenuto dalla "potatura" (*pruning*) di tutti i rami che eccedevano una prefissata profondità[11].

[11] Data la bassa capacità computazionale delle macchine di quel tempo, il limite massimo entro il quale il sottoalbero poteva essere espanso, consisteva di 4 semimosse, il che significa : Tutte le prime mosse del bianco, tutte le prime repliche del nero, tutte le seconde mosse del bianco, e tutte le seconde repliche del nero.

Tutti i nodi terminali dell' sottoalbero di ricerca venivano dotati di un punteggio ottenuto con la *funzione di valutazione*.

A questo punto entrava in gioco la procedura minmax, la quale sceglie la mossa che massimizza il punteggio di un giocatore, tenendo conto che anche l' avversario giocherà la mossa a lui più favorevole (backward induction).

La figura 1.1 illustra la procedura minmax, nell'ipotesi che la funzione di valutazione abbia solo tre possibili risultati (1, 0, -1).

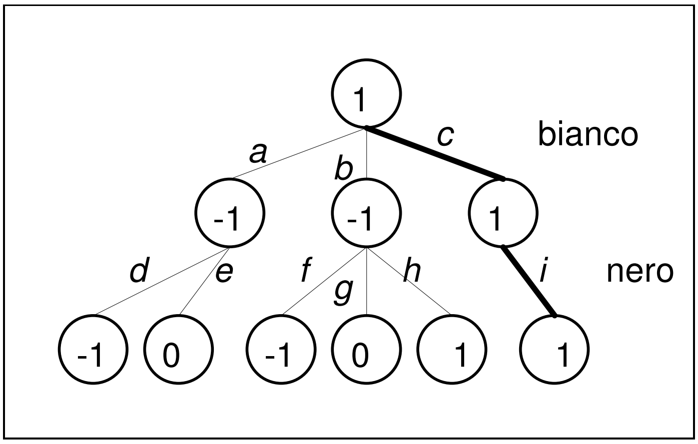


Fig 1.1

Il bianco sceglie l'alternativa *c* in quanto il suo avversario avrà un'unica replica possibile (*i*) che garantirà al bianco un pay-off di 1.

Se il bianco avesse scelto *a* oppure *b*, il nero scegliendo le alternative a lui favorevoli, avrebbe imposto al bianco un pay-off di -1 in entrambi i casi.

Chiaramente se esistesse una funzione in grado di dare una corretta valutazione della posizione, (nel senso che: scegliere il nodo con il punteggio più alto porta necessariamente alla vittoria) avremmo già risolto il problema del gioco degli scacchi, e probabilmente di tutti i giochi ad informazione perfetta; in realtà tale funzione di valutazione rappresenta la pietra filosofale dei progettisti di questo tipo di algoritmo. Non esiste difatti

una *formula magica*, che mi permetta di affermare che un nodo non terminale dell'albero mi conduca alla vittoria, e che quindi debba essere preferito ad un altro.
In verità il valore assegnato ai nodi rappresenta un' approssimazione molto rozza delle reali prospettive di una posizione: molte volte tale metodo funziona e l'algoritmo sceglie effettivamente quella che anche gli esperti giudicano una buona mossa, capita spesso, però, che l' algoritmo scelga delle mosse palesemente errate.
La *funzione di valutazione* deve essere in grado di assegnare un punteggio numerico ad ogni nodo terminale del sottoalbero preso in considerazione dall'algoritmo. Nel momento in cui si stima una posizione di parità, il valore della funzione sarà $f(Pos) = 0$. Un valore positivo deve indicare una posizione vantaggiosa per il bianco, mentre un valore negativo deve indicare superiorità da parte del nero.
Le euristiche che costituiscono la funzione di valutazione possono considerare i più svariati aspetti di una posizione, esse vengono valutate separatamente e poi sommate tra di loro. Per chiarire meglio cosa si intenda per "*aspetti di una posizione*" si consideri il seguente esempio.
Supponiamo di trovarci di fronte ad una posizione scacchistica e di volerla valutare euristicamente. La prima cosa che generalmente un giocatore di scacchi esegue è la valutazione delle forze in campo, di conseguenza sarà considerato avvantaggiato il giocatore che dispone di più pezzi sulla scacchiera. Quello appena citato è un aspetto della posizione, ma non è sicuramente l'unico, infatti, ad esempio, la minoranza di forze in campo può essere bilanciata da un migliore piazzamento dei pezzi, che ad esempio

possono minacciare seriamente la difesa del re nemico. Abbiamo così indicato due possibili aspetti di una posizione.

La struttura tipica di una funzione di valutazione di una posizione è :

$$f(Pos) = \sum_{i=1}^{n} A_i(Pos)$$

dove i termini A_i rappresentano le varie euristiche utilizzate.

Volendo si può variare l'importanza relativa delle euristiche, attribuendo loro un peso, in questo caso avremo:

$$f(Pos) = \sum_{i=1}^{n} p_i \ A_i(Pos)$$

in cui p_i è un peso relativo assegnato alle valutazioni euristiche.

Ecco alcuni esempi di euristiche utilizzate nella maggior parte degli algoritmi:

Bilanciamento del materiale.

L'euristica relativa a questo importante aspetto di una posizione è :

$$mat(Pos) = Pb(Pos) - Pn(Pos)$$

$Pb(Pos)$ è dato dalla somma dei valori dei pezzi bianchi presenti nella posizione, analogamente $Pn(Pos)$ è la speculare valutazione per il nero.

La seguente tabella indica il valore relativo dei pezzi, tali valori in linea di massima sono presenti in tutti i più recenti algoritmi, nonchè accettati come parametro rozzo di valutazione anche dai più forti giocatori umani.

pedone = 1	cavallo = 3
alfiere = 3.25	torre = 5
regina = 9	re = 999

Se costruissimo un'algoritmo sulle basi di quello di Shannon, con una funzione di valutazione basata esclusivamente sul bilanciamento del materiale, il giocatore artificiale che ne scaturirebbe, sarebbe ingordo e prevedibile. Infatti, privilegiando le mosse che gli consentono di *guadagnare materiale*[12] , la macchina corre il rischio di trascurare questioni di vitale importanza, come ad esempio, la salvaguardia del re.

Mobilità dei pezzi.

La teoria del gioco afferma che un fattore determinante per la vittoria è la mobilità delle proprie forze in campo, detto in altri termini, la possibilità di controllare più caselle possibili con il raggio d'azione dei propri pezzi, specularmente, una posizione sarà tanto più vantaggiosa, quanto più si riduca la mobilità dei pezzi avversari. Il fattore mobilità viene così espresso:

$$MOB(Pos) = MOBb(Pos) - MOBn(Pos)$$

Dove $MOBb(Pos)$ è il numero delle mosse legali a disposizione del bianco in una certa posizione.

Oltre alle due euristiche sopra descritte ne esistono diverse altre, per una dettagliata descrizione delle quali si rimanda alla bibliografia.

E' comunque stato sperimentato *(Schaeffer 1986)* che quanto più è raffinata la funzione di valutazione, nel senso che è composta da un numero maggiore di euristiche, tanto migliore è la qualità di gioco espressa dall'algoritmo che le utilizza.

La strategia di espansione dell'albero di ricerca proposta da Shannon, prende il nome di "espansione cieca" o per "forza bruta", la quale va distinta dalla "espansione euristica"[13]

[12] Guadagnare materiale, nel gergo scacchistico significa catturare un pezzo nemico senza che l'avversario possa fare altrettanto.

[13] Gia Shannon aveva intuito i due possibili differenti approcci, definendo il primo "type A strategy" e il secondo come "type B strategy".

L'espansione cieca effettua un'esplorazione esaustiva di tutti i nodi dell'albero di gioco che sono situati ad un livello di profondità prestabilito. E'stato stimato che ad ogni nuova mossa si presentano in media 38 possibili alternative : stabilendo una profondità di analisi di sole 2 semimosse, con una ricerca esaustiva si dovrebbero visitare circa 1500 nodi. E' importante sottolineare che nell'espansione cieca, la *funzione di valutazione* viene applicata solo ai nodi terminali dell'albero di ricerca.

L'espansione euristica invece rinuncia ad un esplorazione esaustiva dei nodi a favore di una espansione selettiva di quei nodi che presentano maggiori prospettive di successo, sulla base della *funzione di valutazione*.

Sacrificando l'esaustività della ricerca dei nodi ad un dato livello, si può permettere all'algoritmo di effettuare ricerche ad una profondità di gran lunga superiore. E' opportuno osservare che con questo metodo, la funzione di valutazione viene applicata a tutti i nodi dell'albero di ricerca, in quanto è proprio essa che permette di decidere quali nodi tagliare e su quali nodi proseguire l'esplorazione.

Il primo programma ad utilizzare questo tipo di ricerca fu il *Bernstein program*, costruito nel 1957 da alcuni programmatori dell' IBM, il quale ad ogni livello di profondità dell'albero, considerava solamente le 7 alternative con un punteggio più elevato.

L'espansione euristica fu la preferita dai ricercatori americani durante gli anni '60 , in quanto sembrava essere la più adatta a imitare i processi cognitivi della mente umana.

Contemporaneamente, i ricercatori sovietici avevano adottato la strategia di espansione cieca. Negli stessi anni ebbe così luogo una affascinante disputa tra le due differenti scuole, e nei vari match che si seguirono, ebbe decisamente la meglio la scuola sovietica.[14]

I successi degli algoritmi che effettuavano una visita esaustiva dei nodi dell' albero, stimolarono la ricerca in questa direzione , in tal modo ci si allontanò dai propositi iniziali dei primi ricercatori, vale a dire quelli di riprodurre artificialmente i meccanismi umani nella formulazione delle decisioni.

In tutti gli anni '70 e fino a metà degli anni '80, non ci furono grandi progressi dal punto di vista degli algoritmi, la struttura dei programmi proseguiva sul filone "brute force", con l'ausilio dell' algoritmo *Alfa-Beta Pruning* ed altri raffinamenti dell'algoritmo minmax i quali aumentano l'efficienza della ricerca esaustiva, riducendo sensibilmente il numero di nodi da visitare[15]. Ad esempio, solo con l'ausilio dell'algoritmo *Alfa-Beta Pruning*, il numero di posizioni da valutare, nella migliore ipotesi, da N diventa in media $\sqrt{N}$.

Il progresso nella qualità del gioco delle macchine, era dovuto all'aumento delle capacità computazionali dei calcolatori, i quali erano in grado di esplorare esaustivamente l'albero ad una profondità di analisi sempre più elevata.

Con la seconda metà degli anni '80, riprende il raffinamento degli algoritmi, i quali proseguono l'espansione euristica solo dopo aver eseguito un'espansione cieca fino ad un livello prefissato. Inoltre vengono introdotti

[14] Nel 1966 il programma Kotok/McCarthy sviluppato alla Stanford University venne sonoramente sconfitto in un match da un programma sviluppato da Arlazarov,Adelson-Velsky,Bitman all'università di Mosca.

[15] Alcuni di questi : "Dead position heuristic" introdotta da Turing (1953), "Transposition tables" (Greenblatt 1967), "Killer heuristic" (Gillogly 1972). Per una descrizione di questi si veda *Levy (1988).*

altre tecniche di ricerca che si affiancano alla struttura portante del programma[16].

I migliori programmi attuali seguono tutti la strada della ricerca selettiva all'interno dell'albero di gioco, la potenza computazionale della macchina rimane comunque un elemento fondamentale. La ricerca odierna è orientata al progetto di nuove architetture hardware a "parallelismo massiccio" in grado di visitare un numero sempre maggiore di nodi.

La punta di diamante di questo filone è rappresentata dal progetto "*Deep thougt*" ideato da un gruppo di ricerca della Carnegie Mellon University.

La macchina affianca ad un algoritmo di ricerca selettiva, una potente struttura hardware che utilizza tecnologie di integrazione su grandissima scala in grado di esplorare più di due milioni di nodi al secondo. La nuova versione di *Deep thougt* è in fase di progettazione ed è sovvenzionata dall' IBM. Si stima che un ulteriore potenziamento dell' hardware permetterà di esplorare fino a 10^9 nodi al secondo.

Il raffinamento degli algoritmi di ricerca selettiva ha permesso di raggiungere buoni risultati anche ai programmi che "girano" sui PC. Il gap qualitativo tra i giocatori artificiali implementati su mainframe e quelli implementati su PC si è ridotto sensibilmente in questi ultimi anni (*Milocco 1993*).

L'ultimo campionato mondiale per giocatori artificiali su PC (luglio 1993) è stato vinto dal programma *Socrates II* che gira su PC 486 .[17]

---

[16] Alcuni di questi : Conspiracy numbers algorithm (Mc Allester 1988), Min-Max approximation (Rivest 1988), Singular extensions (Anantharaman et al. 1988), Equi-potential search (Anantharaman 1990). Citati in *Simon & Schaeffer 1992*.

[17] Crf. ACMmembernet Vol 37, N.7 luglio 1993.

Anche se il livello qualitativo del gioco espresso da queste macchine è aumentato, fino ad oggi la qualità di gioco espressa dai giocatori umani al massimo livello risulta essere superiore[18].
La fig 2.1 illustra il progresso nella qualità del gioco espressa dai giocatori artificiali negli ultimi anni. L'indice di riferimento è costituito dall' *indice E.L.O.*, che è un punteggio che viene assegnato ad ogni giocatore di torneo.
I calcoli per ricavare l' *indice E.L.O* sono piuttosto complessi, comunque basta accennare che il punteggio è ottenuto sulla base dei risultati di un giocatore, ed è tanto più elevato quanto più il giocatore vince.
Poichè da molti anni i giocatori artificiali partecipano a tornei in cui sono presenti giocatori "umani", ne è stato possibile valutare l' *E.L.O.*

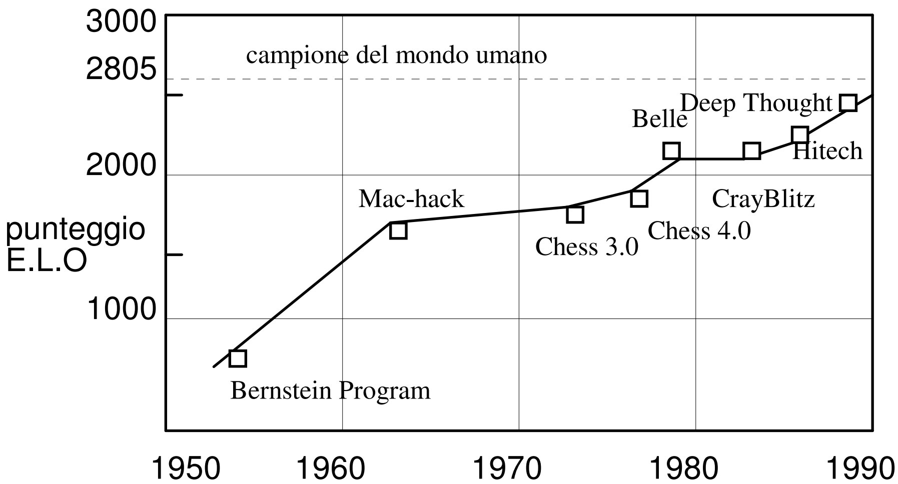


Fig. 2.1

Per avere dei punti di riferimento si sappia che il campione del mondo di scacchi G.Kasparov ha un indice di circa 2800 punti ; i forti giocatori

[18]Ne è una prova la facilità con cui il campione del mondo G.Kasparov ha sconfitto Deep thougt in un match (New York 1989).

professionisti dai 2500 punti in su, mentre la maggior parte dei giocatori dilettanti ha un indice che va dai 1500 ai 2000 punti.

5.2 Il programma Newell Shaw Simon

Gli algoritmi impostati sui suggerimenti di Shannon hanno rappresentato il filone dominante della progettazione di programmi che giocano a scacchi. Il potenziamento dell' hardware delle macchine ha permesso di espandere in profondità gli alberi di ricerca migliorando la qualità del gioco e consentendo di ottenere notevoli risultati agonistici.
Se però ricordiamo quanto emerge dalle ricerche condotte in ambito psicologico, l'approccio al gioco seguito dall'uomo è completamente diverso da quello seguito dalle macchine sopra descritte; infatti mentre l' umano, per decidere che mossa fare, raramente giunge a considerare un centinaio di

possibili posizioni, la macchina considera un numero di posizioni che si aggira sull'ordine di 10^9.

Considerando questi due differenti approcci, non si può dire che l'uno sia più o meno razionale dell'altro; entrambi sono *proceduralmente razionali* in quanto cercano di ottenere un obiettivo (giocare una buona mossa) tenendo conto dei limiti computazionali e della conoscenza.

L' algoritmo progettato nel 1958 da Newell, Shaw e H.A. Simon (NSS) merita una particolare considerazione per l'originalità con cui affronta il problema. Il NSS si discosta ampiamente dai paradigmi dominanti nella progettazione di macchine che giocano a scacchi, in quanto sostituisce la *funzione di valutazione* con un meccanismo di *soddisfazione di sotto-obiettivi*.

Il programma non viene certamente ricordato per i suoi risultati agonistici anche perchè l'uso di un linguaggio sperimentale utilizzato nella implementazione penalizzò notevolmente l'efficienza del programma, che venne usato per giocare una sola partita[19]. Quello che invece si intende sottolineare è come in questo programma siano presenti tutte i punti essenziali del *Behavioral model of rational choice* che caratterizzano l'innovazione del pensiero di Simon. (Razionalità limitata, soddisfacentismo, semplificazione della funzione di utilità, scomposizione del problema complesso in una gerarchia di sottoproblemi).

Il programma di Newell Shaw e Simon, (NSS) è preceduto da approfonditi studi sui processi cognitivi umani e sui processi decisivi, non è un caso che proprio nel 1955 veniva pubblicato il modello di Simon.

[19] NSS-Simon (Pittsburgh, 1960)

Gli autori sottolineano come sia utile cercare di conoscere e di riprodurre meccanicamente quello che l'uomo fa naturalmente nel momento in cui prende una decisione in un contesto di razionalità limitata: *gli scacchi sono un "arena naturale" per sperimentare l'efficacia di un processo di decisione artificiale, in quanto la validità delle scelte intraprese può essere inequivocabilmente valutata sulla base dei risultati ottenuti (vittoria, pareggio, sconfitta).(Simon 1958).*

Struttura fondamentale del programma

Come abbiamo accennato in precedenza, una caratteristica fondamentale del NSS è quella di scomporre un obiettivo complesso (scegliere una buona mossa), in una gerarchia di *sotto-problemi* più semplici. Per *sotto-problema* si intende il compito di stabilire se una data mossa possa reputarsi soddisfacente da un certo punto di vista. Il fatto di considerare una posizione da un certo punto di vista significa esaminare solo alcune caratteristiche della posizione, tralasciandone altre.

Ad esempio: supponiamo che in una certa posizione il bianco abbia la possibilità di catturare la regina avversaria con un proprio pedone; se consideriamo la mossa esclusivamente dal punto di vista del *bilanciamento del materiale* essa sarebbe da considerarsi ottima; se invece cambiassimo punto di vista, tenendo conto della *salvaguardia del re* potrebbe essere che la mossa venga reputata pessima perchè alla mossa successiva il nero potrebbe ad esempio dare scaccomatto al bianco.

La tecnica di considerare differenti aspetti di una posizione viene utilizzata anche dalla *funzione di valutazione*, con la differenza che in essa tutti i punteggi ottenuti dai singoli moduli vengono sintetizzati in un unico valore[20].

Il programma NSS è invece costituito da un insieme di moduli, ciascuno dei quali valuta le possibili mosse secondo il *proprio punto di vista*; vi è poi un'ulteriore routine che raccoglie le valutazioni provenienti dai singoli moduli, e sceglie la mossa migliore.

Le analogie tra il programma e il *behavioural model of rational choice* sono evidenti in due punti :

1. Nel modo con cui i singoli moduli esprimono il loro giudizio su una certa mossa.
2. Nel modo in cui la routine principale sceglie la mossa.

Il primo punto è caratterizzato dalla *semplificazione della funzione di utilità*, nel senso che i moduli non danno un punteggio alla mossa, come avviene nella funzione di valutazione, ma esprimono solamente un giudizio di accettabilità o meno, secondo i loro criteri di valutazione. Un modulo, allora, si potrà esprimere solo attraverso un valore che sarà 1 se reputa la mossa soddisfacente e 0 se reputa la mossa insoddisfacente.

La routine che deve scegliere una mossa *a* tra l'insieme **A'** di alternative prese in considerazione dovrà *tirare le somme* sui giudizi espressi dai vari moduli. In altri termini dovrà confrontare tra loro una serie di vettori del tipo indicato in fig 3.1.

[20] Cfr. Par. 5.1 pag 18.

a)	b)	c)	d)	e)
1	**0**	**0**	**1**	**1**

a) Salvaguardia del re
b) Bilanciamento del materiale
c) Controllo del centro
d) Sviluppo
e) Promozione del pedone

Fig 3.1

Nell'esecuzione di questo compito ritroviamo un'ulteriore elemento presente nel modello di Simon, quello che viene definito *ordinamento parziale dei pay-offs*.

Ogni mossa candidata ad essere scelta viene valutata da un vettore: sorge allora il problema di dover confrontare tra loro grandezze non ordinabili, in quanto raramente disporrò di una mossa in grado di soddisfare tutti i moduli del programma.

Il programma NSS risolveva il problema in questione *gerarchizzando i sotto-obiettivi*, nel senso che venivano privilegiate le mosse che *soddisfavano* gli obiettivi prioritari.[21]

La gerarchia di sotto-obiettivi non è statica ma si evolve nel corso della partita, ad esempio in fase di *apertura*[22] vengono privilegiati i giudizi

[21] In alrti termini le mosse venivano confrontate con il criterio lessicografico.

[22] La fase di apertura comprende le prime mosse della partita, in questa fase riveste particolare importanza lo sviluppo, che in gergo scacchistico significa mettere in gioco i pezzi dalla loro posizione di partenza verso il centro della scacchiera. Anche il controllo del centro è di vitale importanza, e consiste nell'occupare le caselle e4,e5,d4,d5 con i propri pedoni.

espressi dai moduli *sviluppo* e *controllo del centro*, mentre nel *mediogioco*[23] diventa prioritaria la *salvaguardia del re*.

Nel momento in cui il programma non è in grado di trovare una mossa che *soddisfi* almeno gli obiettivi prioritari viene messa in atto una tra le due possibili azioni:

- Abbassamento del livello di aspirazione.
- Ulteriore raccolta di informazioni.

Con l'*abbassamento del livello di aspirazione* ogni singolo modulo reputerà soddisfacente un insieme più ampio delle mosse prese in considerazione.

L'ulteriore *raccolta di informazioni* avviene in contrapposizione all' abbassamento del livello di aspirazione, e consiste in un'ulteriore ricerca in profondità nell'albero del gioco al fine rivalutare il giudizio su una mossa precedentemente reputata inaccettabile.

5.3 L'approccio " knowledge based ".

[23]La fase di mediogioco si ha al termine della fase di apertura, è caratterizzata da posizioni molto complesse in cui la maggior parte dei pezzi è entrata in gioco e i giocatori preparano le loro strategie per sfoderare l'attacco all'avversario.

Confrontando l'approccio umano al gioco con quello degli algoritmi sopra descritti, si può immediatamente constatare quale sia la fondamentale differenza tra i due.
L'uomo ha una inferiore potenza computazionale[24], che viene compensata in modo efficiente da una base di conoscenza che gli permette di selezionare gli archi dell'albero su cui proseguire la ricerca.
Gli algoritmi di espansione euristica dell' albero di ricerca, avevano cercato di seguire questa strada, ma con scarso successo, in quanto la *funzione di valutazione*, per raffinata che sia, rimane sempre uno strumento assai rozzo se paragonato alla capacità umana di associare la posizione in esame, ad un database di conoscenza costituito da migliaia di *chunks*[25].
Sulle basi di queste constatazioni, si sviluppa a partire dagli anni '80 un filone di sviluppo di algoritmi "Knowledge based", che cercano di limitare l'espansione dell'albero di gioco, per mezzo della conoscenza. Sullo sviluppo di questi algoritmi, comunque, gia in precedenza si erano interessati altri esperti in scienze cognitive, tra i quali Newell e Simon ('MATER' 1972).
Un esempio successivo, illustrerà come tale approccio risulta valido nella soluzione di *problemi finali*[26].
I giocatori artificiali che scaturiscono da un approccio knowledge based, si possono definire dei veri e propri sistemi esperti in grado di utilizzare la loro base di conoscenza per pianificare la strategia da seguire nelle mosse successive.

---

24 In quanto raramente riesce ad esplorare più di 100 nodi del'ìalbero di gioco, nel momento in cui deve decidere quale mossa giocare.
25 Vedi par.4
26 Si dicono " finali" le posizioni con la presenza di pochi pezzi in campo; in questi contesti, la conoscenza risulta essere la componente principale nella scelta della strategia da adottare.

In sintesi un sistema esperto è composto da tre moduli i quali interagiscono tra loro. Essi sono illustrati in fig. 4.1 (*Bratko 1986*).

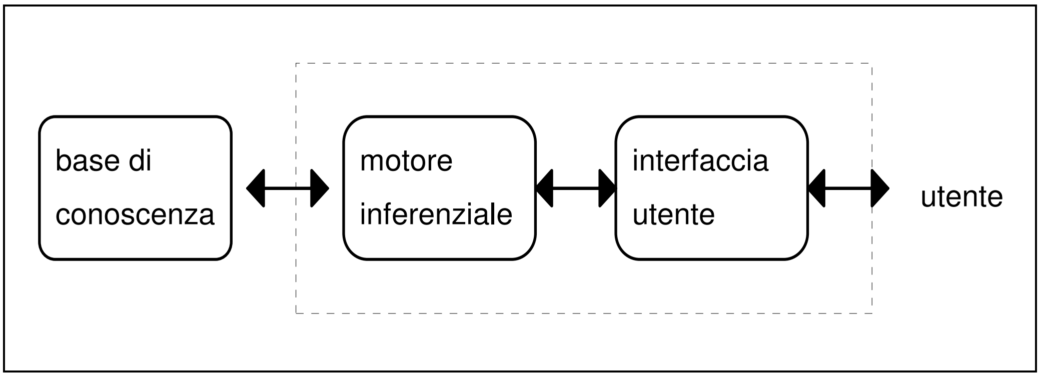


fig. 4.1

La "base di conoscenza" comprende le nozioni necessarie per affrontare il problema in questione. Tali nozioni possono essere costituite da regole che descrivono relazioni o caratteristiche del problema esposto dall'utente. Nel caso di una *base di conoscenza scacchistica* troveremo le nozioni necessarie a classificare la posizione in esame in categorie con caratteristiche affini. La stessa base di conoscenza, contiene suggerimenti, euristiche, e metodi che, una volta riconosciuto il tipo di posizione in esame, *pilotano* il sistema esperto nella ricerca della strategia soddisfacente.

Il "motore inferenziale" è la parte attiva del sistema esperto, in quanto è in grado di ricorrere alla base di conoscenze per risolvere il problema.Nel caso di un sistema esperto scacchistico, il *motore inferenziale* ha il compito di utilizzare la base di conoscenza per costruire degli alberi di forzatura, del cui significato ci occuperemo tra breve.

L' "interfaccia utente" provvede alla comunicazione tra l'utente e il sistema, facendo in modo che l'utente abbia una visione completa di come abbia luogo il processo di soluzione del problema da parte del motore inferenziale.
La particolare categoria di sistemi esperti progettata per risolvere problemi scacchistici è indicata col nome di "*Advice languages*"*(Bratko 1986)*.
Il concetto che sta alla base di un *advice language* è il "consiglio" (*piece of advice*). Il *consiglio* indica cosa si deve fare, o cosa si deve tentare di fare in una data posizione: esso viene rappresentato da un obiettivo che deve essere raggiunto e dai mezzi che devono essere impiegati per raggiungerlo.
Ogni *consiglio* è composto da :

1) Obiettivo primario : L'obiettivo che deve essere raggiunto.
2) Obiettivo minimo : Un obiettivo che deve sempre essere mantenuto durante il gioco che conduce al conseguimento dell' obiettivo primario.
3) Nostri vincoli di movimento : Una procedura che seleziona tra tutte le mosse possibili quelle che portano al conseguimento dell'obiettivo primario.
4) Loro vincoli di movimento : Una procedura che seleziona le mosse avversarie che possono compromettere il conseguimento dell'obiettivo in questione.

Se ad esempio dovessimo trattare una posizione finale in cui *noi* disponiamo di pedone e re mentre *loro* dispongono del solo re, *noi* cercheremo di vincere spingendo il pedone fino all'ultima traversa. Un possibile consiglio potrebbe essere configurato nel seguente modo.
1) Obiettivo primario : Spingere il pedone

2) Obiettivo minimo : Evitare che il re avversario catturi il pedone

3) Nostri vincoli di movimento : Tutte le possibili mosse di re e di pedone

4) Loro vincoli di movimento : Tutte le possibili mosse di re

Un *consiglio* si dice *soddisfabile* se è possibile forzare il raggiungimento dell'obiettivo primario da parte del giocatore, con la condizione che l'obiettivo minimo non sia mai violato.

Se è possibile soddisfare un *consiglio* ne deriverà un albero di forzatura (*Forcing-tree*), ovvero un albero nel quale viene indicata dettagliatamente la strategia che occorre per la realizzazione dell'obiettivo principale (Fig 5.1).

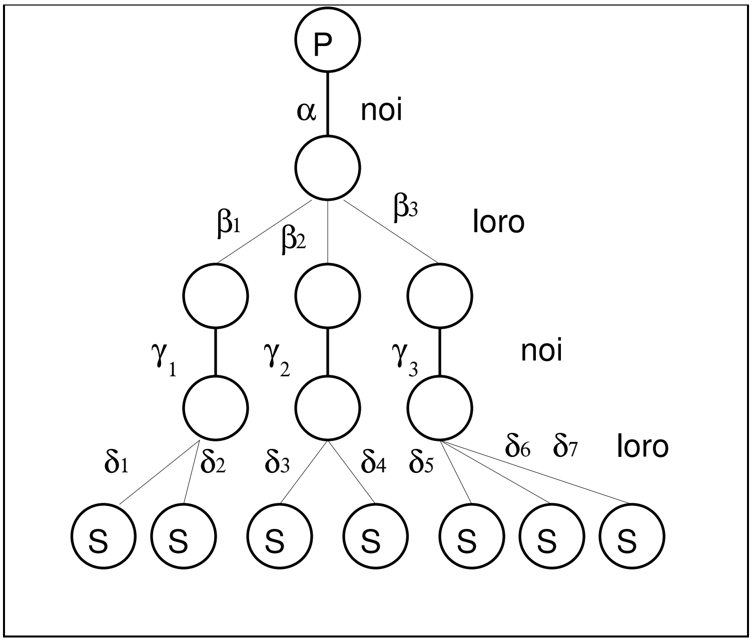


fig 5.1

Con riferimento alla figure 5.1, l' *albero di forzatura* descrive la seguente strategia:

gioca α.

se *loro* risponde β_1 gioca γ_1

se *loro* risponde β_2 gioca γ_2

se *loro* risponde β_3 gioca γ_3

Un *albero di forzatura* specifica cosa "*noi*" dobbiamo fare in relazione ad ogni "*loro*" risposta. Più precisamente, un albero di forzatura **T** relativo ad un consiglio **C** in una data posizione **P** è un sottoalbero dell'albero di gioco tale che

- La radice di **T** corrisponde a **P**.
- Tutti i nodi in **T** soddisfano l'obiettivo minimo.
- Tutti i nodi terminali di **T** soddisfano l'obiettivo primario e nessun nodo interno di **T** soddisfa l'obiettivo primario.
- Esiste un solo arco fuoriuscente dalle posizione nelle quali tocca a *noi* muovere, l'arco corrispondente a quella mossa deve soddisfare i "*nostri vincoli di movimento*".
- Gli archi fuoriuscenti dalle posizioni nelle quali tocca a *loro* muovere corrispondono a tutte le mosse che soddisfano i "*loro vincoli di movimento*".

Nell' "Advice language" ogni singolo *consiglio* viene integrato nel *knowledge base* per mezzo della seguente gerarchia.

Una lista di *consigli* compone una regola *if-then*.

Un' insieme di regole *if-then* costituisce una *tabella di consigli* (Advice table).

Una collezione di tabelle di consigli viene strutturata in modo che ad ognuna di esse venga assegnato uno specifico sub-problema. La collezione di tabelle di consigli, che assieme alla libreria dei predicati (del cui significato parleremo dopo) costituisce il *knowledge base* del nostro sistema esperto.

Per chiarire le idee ecco un'esempio di tabella di consigli specializzata nella soluzione di una posizione finale in cui "*noi*" disponiamo di torre e re e dobbiamo vincere contro l'avversario che dispone di solo un re.
Le regole *if-then* sono strutturate nel modo seguente :

NomeRegola : *if* Condizione *then* ListaConsigli.

Condizione è costituita da un insieme di predicati connessi da connettivi logici del tipo and , or, not.
ListaConsigli è una lista di *consigli* posti in ordine di priorità decrescente nel senso che si tenterà prima di soddisfare il primo *consiglio*, se questo non è possibile si passerà al secondo della lista e così via.
Ecco un esempio di regola *if-then* estratta dal knowledge base di un programma in grado di giocare correttamente un finale di torre e re contro re *(Van Emden 1982)*.

Regola_bordo : *if* loro_re_al_bordo *and* nostro_re_vicino
then [matto_in_2, restringi,avvicina,mantieni,dividi]

Il significato della regola è il seguente: Se nella posizione che stiamo esaminando, il re avversario è collocato al bordo della scacchiera, e il nostro re è vicino a lui, allora cerca di soddisfare in ordine di preferenza i seguenti consigli : cerca di dare scacco matto in due mosse, se questo non è possibile cerca di restringere il raggio d'azione del re avversario... ecc.

Ogni consiglio è strutturato nella seguente maniera:

consiglio(NomeConsiglio,
Obiettivo primario :
Obiettivo minimo :
Nostri vincoli di movimento :
Loro vincoli di movimento).

Gli obiettivi sono espressioni composte da predicati e connettivi logici del tipo and, or, not. I vincoli di movimento sono anche composti da predicati e dai connettivi *and* e *then_after* (che significa "dopo la replica dell'avversario").
Ecco come verrebbe indicato un consiglio relativo al problema che stiamo discutendo.

consiglio(matto_in_2,
matto :
non_perdere_torre :
(profondità = 0) *and* legale *then_after* (profondità = 2) and scacco :
(profondità = 1) *and* legale).

In questo *consiglio* l'obiettivo principale è costruire un albero di forzatura che porti allo scaccomatto in due mosse; l'obiettivo minimo è quello di non perdere la torre. I "nostri vincoli di movimento" dicono di analizzare tutte le nostre mosse possibili che partono dalla posizione iniziale (profondità = 0

and legale), e dopo la replica dell'avversario, solo le mosse che danno scacco (profondità=2 and scacco).
I "loro vincoli di movimento" dicono di analizzare tutte le possibili repliche dell'avversario (profondità=1 and legale).
Se il consiglio è soddisfabile, il sistema giocherà come indicato dall'albero di forzatura, mentre in caso opposto si passerà al successivo consiglio così come indicato dalla regola *if-then*.
In sintesi una *tabella di consigli* viene utilizzata ripetendo, fino alla fine del gioco, il seguente ciclo principale:

ripeti fino a scaccomatto :
  costruisci un albero di forzatura.
  gioca secondo le indicazioni dell'albero.

Un albero di forzatura viene costruito come segue:

- Considera la posizione corrente ed esamina le regole *if-then* della tabella dei consigli.
- Per ogni regola confronta la posizione con le condizioni richieste dalla regola stessa e fermati quando la posizione soddisfa le condizioni richieste dalla regola.
- Considera la lista di consigli fornita dalla regola e verifica uno ad uno i consigli finchè trovi quello soddisfabile che genera l'albero di forzatura.

Come abbiamo accennato in precedenza, un sistema esperto è composto da tre parti che interagiscono tra loro : la base di conoscenza, il motore inferenziale e l'interfaccia utente.

Nel nostro caso l' "Advice language" può essere diviso in tre moduli.

- Interprete "Advice language"
- Tabella dei consigli
- Libreria dei predicati.

L'interprete costituisce il motore inferenziale, mentre la tabella dei consigli e la libreria dei predicati costituiscono la base di conoscenza.
L'interfaccia utente in questo esempio si limita a ricevere le mosse giocate dall'avversario e a restituire le mosse che il sistema reputa valide: non abbiamo quindi una grande interazione sistema-utente per mezzo della quale il sistema spiega anche i motivi per cui è stata scelta una certa mossa.
La funzione principale dell'interprete Advice Language, è quella di costruire l'albero di forzatura che dà come risultato un albero come quello in fig 5.1.

6. Considerazioni conclusive.

Dopo aver introdotto alcune nozioni di razionalità, soffermandoci in particolare modo sul modello di Simon e le definizioni di razionalità procedurale e sostanziale, abbiamo visto come l'unico approccio al gioco degli scacchi che sia in grado di darci delle indicazioni normative, è un approccio con razionalità procedurale, che ha per sbocco naturale la creazione di un giocatore artificiale *(Binmore 1988)*.
Nei capitoli successivi analizzeremo un parente stretto del gioco degli scacchi, che è caratterizzato da forti componenti di incertezza e complessità : il kriegspiel.
In particolare nel capitolo 2 si tenterà un approccio al gioco di tipo sostanziale cercando di metterne in evidenza i pregi e i limiti. Nel capitolo 3 cambieremo ottica tentando un approccio di tipo procedurale con la modellizzazione di un giocatore artificiale.

BIBLIOGRAFIA DEL CAPITOLO 1

Anantharaman T.; F.Hsu, "Il calcolatore Grande Maestro di scacchi" : Le scienze n.268 Dic. (1990)

Arrow K.J. "Alternative Approaches to the theory of choise in Risk-Taking Situations", in "Econometrica", 19, pp. 404-437 (1951)

Aumann R.J. "What is game theory trying to accomplish ? " in Arrow-Honkapohja (1985) pp. 28-76

Binmore K. "Modeling rational players, part I " : Economics and Philosophy n.3 (1987) pag. 179-214.

Binmore K. "Modeling rational players, part II " : Economics and Philosophy n. 4 (1988) pag 9-55.

Bratko I. (1986) *Prolog programming for artificial intelligence*. Addison Wesley .

Ciancarini P. (1992) *I Giocatori artificiali*. Mursia

Faccipieri S. "L'analisi strategica" in Rispoli (a cura di) *L'impresa industriale* Il mulino (1989)

Kreps D. (1990) *Teoria dei giochi e modelli economici* (Trad. It.). Il mulino

Levy D. (1988) *Computer games Vol I* . Springer Verlag

Milocco F. "Progressi nei giocatori artificiali per PC" : "Scacco!" (Aprile 1993)

Moates D.R, Schumacher G.M. (1980) *An introduction to cognitive psychology* (Trad. It. Psicologia dei processi cognitivi) Il Mulino.

Newborn M. "Recent progress in computer chess" : Advances in computers Vol 18 pp.59-117 (1979).
Pearl J. *Heuristics: Intelligent Search Strategies for computer Problem Solving* . Addison Wesley (1984)

Shannon C. "A computer chess machine" : Scientific American feb. (1950)

Simon H.A. (1955) "A behavioural model of rational choice" : Quarterly journal of economics n.69 pag. 99-118

Simon H.A.; A.Newell; J.C.Shaw (1958) "Chess-playing programs and the problem of complexity" : IBM Journal of research & development Vol.2

Simon H.A. (1964) "Rationality" in : A dictionary of the social sciences

Simon H.A. (1972) "Theories of bounded rationality" : in Decision and organization. North Holland Publishing Company

Simon H.A. (1973), *Le scienze dell'artificiale*, Isedi, Milano.

Simon H.A. (1976) "From substantive to procedural rationality" : in Method and appraisal in economics" Cambridge Univ. Press

Simon H.A. (1978) "On how to decide what to do" : The Bell journal of economics 9 n.2

Simon H.A. (1978) "Rationality as Process and as Product of Thought" : American economic review 68 n.2

Simon H.A. (1979) "Rational decision making in business organizations" : American economic review n.4 [la metafora dell'impero coloniale]

Simon & J.Schaeffer (1992) "The game of chess" in : R.J.Aumann, S.Hart "Handbook of Game Theory"

Van Emden M. "Chess end-game advice : a case study in computer utilization of knowledge". in Machine Intelligence 10. Ellis Horwood (1982).

Capitolo 2

Kriegspiel e razionalità sostanziale

1. Illustrazione secondo la teoria dei giochi

Kriegspiel è un gioco su scacchiera che appartiene alla categoria delle varianti eterodosse del gioco degli scacchi; per molti aspetti ha le stesse regole degli scacchi: viene giocato da due persone su una scacchiera 8 x 8, i pezzi in gioco e il loro movimento sono gli stessi del gioco tradizionale. Vi è però una fondamentale differenza che lo rende completamente originale rispetto agli scacchi e le altre varianti eterodosse: i giocatori non sono a conoscenza delle mosse eseguite dal loro avversario[27], quindi agiscono in un contesto di parziale incertezza.

E' difficile classificare il kriegspiel secondo i modelli forniti dalla teoria dei giochi. Mentre sappiamo che gli scacchi sono un gioco ad *informazione perfetta*[28], prima di definire Kriegspiel un gioco ad *informazione imperfetta* o ad *informazione incompleta*, sembra opportuno fare alcune considerazioni.

Anzitutto Kriegspiel è un *gioco multistadio*[29]: il soggetto decisore sceglie di volta in volta quale mossa intraprendere alla luce delle informazioni di cui

---

27 La descrizione dettagliata delle regole del gioco è riportata in appendice.

28 Un richiamo alle definizioni in corsivo è riportato nell' appendice 2.

29 "Multistage game" secondo la definizione di Owen. (Owen 1982).

viene a disporre. Nei giochi multistadio assume un maggiore rilievo il concetto di *strategia comportamentale* che, rispetto ai concetti di *strategia pura* o *mista,* sembra essere più adeguato nell'approccio al gioco.

I fattori che condizionano l'appartenenza del kriegspiel alla categoria dei giochi ad informazione imperfetta o ad informazione incompleta sono essenzialmente due :

- La struttura della posizione iniziale
- La complessità della posizione.

Per descrivere il primo fattore bisogna premettere che una partita a kriegspiel può inizialmente presentarsi in due modi: la posizione iniziale può essere nota ad entrambi i giocatori, e quindi costituire un unico insieme informativo, oppure gia a partire dalla posizione iniziale possono esistere condizioni di incertezza da parte di uno o entrambi i giocatori, nel senso che ad uno o ad entrambi i giocatori non è noto con precisione dove siano collocate le forze avversarie.

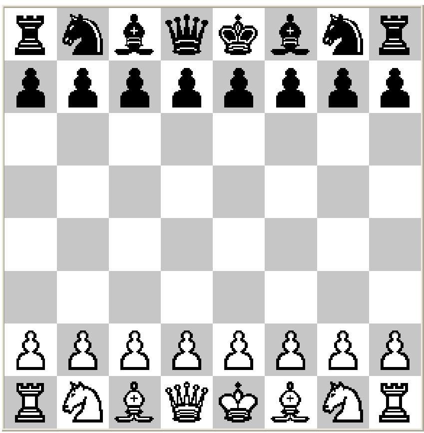


Fig 1.2

Se alla posizione di partenza (Fig 1.2), ipotizziamo che entrambi i giocatori abbiano una conoscenza perfetta della disposizione delle forze dell'avversario, al primo stadio[30] il gioco potrà dirsi ad informazione imperfetta. Vi sono infatti 20 alternative da parte del primo giocatore e 20 repliche da parte dell' avversario e se volessimo rappresentare il tutto in forma estesa otterremo un albero con un nodo ed un insieme informativo nel momento in cui il bianco effettua la prima mossa ; 20 nodi ed un insieme informativo alla prima replica del nero ; 400 nodi e 20 insiemi informativi alla seconda mossa del bianco.

Se il gioco terminasse al primo stadio, esso potrebbe sicuramente definirsi ad informazione imperfetta, assegnando un pay-off ad ogni nodo terminale dell'albero ottenuto.

Nella realtà il gioco continua oltre il primo stadio e la complessità della rappresentazione esploderebbe già alla seconda replica del nero; possiamo quindi constatare come il Kriegspiel sia un gioco nel quale oltre all' elemento di "complicazione" che avevamo definito nel capitolo precedente, ritroviamo anche un elemento di incertezza relativo alla mancanza di conoscenza delle azioni intraprese dall'avversario.

Dalle regole del gioco, inoltre, si può facilmente constatare come nella maggior parte dei casi, negli stadi successivi venga a crearsi una asimmetria nella distribuzione delle informazioni facendo sì che il gioco diventi ad "informazione incompleta"[31].

Ricordiamo infatti che nel momento in cui un giocatore riferisce la mossa all'arbitro, riceverà da quest'ultimo delle informazioni[32] . Per mezzo di

[30] Intendiamo "primo stadio" la prima mossa del bianco e la prima replica del nero; dopodichè il " secondo stadio" sarà rappresentato dalla seconda mossa del bianco e dalla seconda replica del nero, e così via.

[31] Un gioco in cui ciascun giocatore, all' atto della valutazione delle possibili strategie, dispone di informazioni private che non sono a conoscenza degli altri. (Myerson 1991)

[32] Ricordiamo che le informazioni possono essere: Mossa possibile, mossa impossibile, mossa con scacco.

queste informazioni e attraverso una fitta rete di deduzioni logiche che illustreremo in seguito, il giocatore sarà in grado di ottenere delle ulteriori informazioni ignote al suo avversario.

Nel caso della fig 1.2, anche se nella posizione iniziale vi è informazione perfetta, la posizione ha un elevato grado di complessità, nel senso che già a partire dal secondo stadio del gioco, oltre alla esplosione combinatoria delle possibili situazioni, possono crearsi delle asimmetrie nella distribuzione delle informazioni.

In altri casi le posizioni iniziali possono essere presentate in modo da configurare immediatamente il gioco nella categoria ad informazione incompleta.

Ad esempio nella fig. 2.2 è illustrato un problema con una posizione iniziale caratterizzata da asimmetria nella distribuzione delle informazioni.

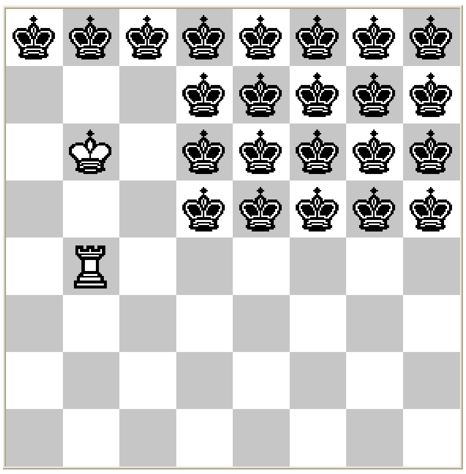


Fig. 2.2

Nella figura viene indicato il set di informazioni così come viene percepito da uno dei giocatori (es. il bianco), il quale pur essendo a conoscenza della posizione dei propri pezzi, ha solo una parziale informazione su dove sia

collocato il nemico, nel senso che l'area in cui il nemico potrebbe essere situato è limitata ad un sottoinsieme delle 64 caselle.
Il problema così posto presenterà delle interessanti varianti, se ipotizziamo che l'avversario abbia o meno un' informazione completa sulla disposizione in campo delle forze bianche.
Con riferimento alla fig.2.2, infatti possiamo allora ipotizzare due diversi scenari :

- L'avversario non sa dove sono collocati i nostri pezzi
- L'avversario sa dove sono i nostri pezzi[33]

Come conclusione di queste premesse, possiamo affermare che in linea di massima il gioco si può definire ad informazione incompleta, anche se, come vedremo negli esempi successivi, molti problemi, entro determinate condizioni, possono ricevere una collocazione più precisa nell'ambito della teoria dei giochi, configurandosi come giochi ricorsivi e stocastici.
Dopo aver eseguito questo tentativo di classificazione del gioco, viene spontaneo chedersi quali siano i risultati che la teoria dei giochi è in grado di darci, dal punto di vista descrittivo e normativo.
Dal punto di vista descrittivo, la teoria dei giochi classica ha senza dubbio il pregio, come sottolineava A.Naddeo (*Naddeo 1967*), di mettere a fuoco gli aspetti fondamentali di un gioco, al fine di comprenderne meglio l'essenza e razionalizzarne l'approccio da parte del giocatore; essa provvede innanzitutto all'astrazione in termini matematici delle caratteristiche salienti derivabili dalle regole di un gioco.

[33] E' interessante notare che anche in questo caso l' avversario non ha un' informazione completa, in quanto pur conoscendo la disposizione del bianco, non è in grado di sapere quali sono le informazioni del bianco sulla posizione del nero.

Più scoraggianti invece sono i risultati normativi della teoria, anche se, come vedremo in seguito, certi problemi del kriegspiel, offrono la possibilità alla teoria dei giochi di intervenire con i suoi strumenti raffinati. In questi casi, che purtroppo sono limitati a situazioni non complesse, le soluzioni che si ottengono sono "sostanzialmente razionali".

Paradossalmente il kriegspiel, gioco indubbiamente più complesso degli scacchi, offre più spazio alla razionalità sostanziale di quanto ne offrano gli scacchi.

Il Teorema di Zermelo, nato appunto da uno studio razionale del gioco degli scacchi, è cronologicamente uno dei primi risultati della teoria dei giochi: esso ha però tolto ogni interesse nell'approccio al gioco in un'ottica "sostanzialmente razionale".

Teorema 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

Il teorema quindi afferma che esiste una strategia pura da parte di uno dei due giocatori che garantisce la vittoria o almeno la patta.

Il teorema di Zermelo viene generalizzato per induzione a tutti i giochi affini agli scacchi.

Teorema (*Kuhn 1953*)

Ogni gioco finito ad informazione perfetta con n-persone ammette una soluzione di equilibrio con una strategia pura.

Dal teorema emerge che l'unico modo razionale di agire al fine di trovare un equilibrio è quello di procedere per *backward induction*, cercando la strategia o le strategie che individuano l'insieme degli equilibri di Nash.
In realtà, come abbiamo visto nel precedente capitolo, risulta impossibile cercare le strategia pure corrispondenti a tali equilibri, i quali, anche se in teoria esistono, non hanno alcuna rilevanza pratica (*Kreps 1990*).
Come avevamo accennato in precedenza, nel kriegspiel possono aver luogo delle situazioni in cui un approccio con razionalità sostanziale mi permette di ottenere delle soluzioni che corrispondono a strategie ottimali in senso assoluto.
Questi problemi presentano delle caratteristiche comuni :

- Hanno un livello di complessità gestibile. Sia dal punto di vista computazionale (la dimensione dell'albero di gioco è relativamente ridotta), che da quello dell'incertezza (gli insiemi di informazione non hanno mai una dimensione eccessiva).
- La loro formalizzazione nel linguaggio matematico della teoria dei giochi, richiede un'adeguata operazione di "interfacciamento". Come vedremo negli esempi successivi, il problema deve essere compreso profondamente, trascurando le alternative irrilevanti. In un certo senso, l'albero di gioco deve essere semplificato al massimo, ma in modo che la soluzione non perda di significato.

In particolare due modelli studiati dalla teoria dei giochi si sono dimostrati particolarmente efficaci nella ricerca della strategia ottimale in relazione a dei problemi finali[34] : si tratta dei *giochi stocastici* e dei *giochi ricorsivi*.
Nei successivi due paragrafi verranno illustrati questi modelli, in seguito esamineremo alcuni esempi che dimostrano l'efficacia della loro applicazione nell'ambito del kriegspiel.

[34] *Problemi finali* è un termine mutuato dal gergo scacchistico che indica le posizioni nelle quali vi è in gioco un numero limitati di pezzi.

2. Giochi stocastici

Si definisce stocastico un gioco composto da due o più matrici di pay-off del seguente tipo:

	β_1^1	β_2^1	
α_1^1	4 & (0.4S, 0.5Γ^1, 0.1Γ^2)	0 & (0.2S, 0.5Γ^1, 0.3Γ^2)	
α_2^1	2 & (0.6S, 0 Γ^1, 0.4Γ^2)	5 & (0.8S, 0.2Γ^1, 0 Γ^2)	(gioco componente Γ^1)

	β_1^2	β_2^2	
α_1^2	2 & (0.6S, 0.1Γ^1, 0.3Γ^2)	9 & (1 S, 0 Γ^1, 0 Γ^2)	
α_2^2	0 & (0.6S, 0.2Γ^1, 0.2Γ^2)	1 & (0.4S, 0.4Γ^1, 0.2Γ^2)	(gioco componente Γ^2)

Supponiamo che il gioco parta dal componente Γ^1 , inoltre il giocatore *1* scelga α_1^1 mentre il giocatore *2* sceglie β_1^1, il pay-off che ne deriva implica che il primo giocatore riceverà 4 dal secondo e inoltre verrà eseguita una lotteria il cui esito potrà essere: S = Fermare il gioco con probabilità 0.4, ripetere il gioco Γ^1 con p = 0.5, giocare il componente Γ^2 con p = 0.1.

Il gioco continuerà fino a quando la lotteria darà esito Stop e il pay-off globale dei giocatori sarà dato dalla somma dei singoli pay-off giocati ad ogni stadio.

Poichè ogni lotteria ha una probabilità di fermarsi che è maggiore di zero, ne segue che la probabilità che il gioco termini dopo un conveniente numero di iterazioni tende a 1.

Partendo dall'esempio di cui sopra è possibile una generalizzazione:

Un gioco stocastico è composto da:

n giochi componenti $\{\Gamma^1, \Gamma^2, ..., \Gamma^n\}$

in ogni gioco componente Γ^k il giocatore *1* ha a disposizione le seguenti strategie : $\alpha_1^k, \alpha_2^k, ..., \alpha_{mk}^k$, mentre il giocatore *2* : $\beta_1^k, \beta_2^k, ..., \beta_{nk}^k$.
Se il giocatore *1* sceglie α_i^k e il giocatore *2* sceglie β_j^k il pay-off sarà:

$$a_{ij}^k \;\&\; (p_{ij}^{k0}S, p_{ij}^{k1}\Gamma^1, p_{ij}^{k2}\Gamma^2, ..., p_{ij}^{kn}\Gamma^n)$$

dove

$$p_{ij}^{k0} > 0,\ p_{ij}^{kl} \geq 0 \text{ per } l = 1,2,..,n$$

e p_{ij} è una distribuzione di probabilità sull' insieme $\{S, \Gamma^1, \Gamma^2, ..., \Gamma^n\}$.

Se indichiamo con $\mathcal{M}$ l'insieme degli n giochi componenti $\{\Gamma^1, \Gamma^2, ..., \Gamma^n\}$ lo specifico gioco che inizia con Γ^k viene indicato con $\{\mathcal{M}, \Gamma^k\}$.
Shapley (1953) fu il primo a definire questo gioco, e ne propose la soluzione individuando un metodo per trovare, partendo da un qualunque gioco componente iniziale, il valore del gioco e la strategia mista ottima per i due giocatori.
Per dimostrare la sua tesi, Shapley aggiunse al gioco una clausola con la quale si stabilisce a priori che il gioco si potrà iterare non più di *n* volte.
Se il gioco termina prima della *n*-esima iterazione, allora nulla cambia rispetto alle regole originali.
Se il gioco giunge alla *n*-esima iterazione, invece di eseguire il gioco previsto, alla iterazione *n*+1, il giocatore *2* pagherà al giocatore *1* un importo prefissato $w_1^{(0)}$ se il gioco fosse continuato con Γ^1, oppure $w_2^{(0)}$ se la continuazione fosse stata Γ^2.
In pratica il gioco viene troncato allo stadio n, con un pay-off $(w_1^{(0)}, w_2^{(0)})$.

E' intuibile che al divergere di n, il gioco anche se modificato avrebbe un valore che non si discosta da quello del gioco originale. Inoltre i valori $(w_1^{(0)}, w_2^{(0)})$ non dovrebbero influenzare il valore del gioco troncato.
Alla n-esima iterazione il gioco figurerà come segue :

	β_1^1	β_2^1	
α_1^1	$4 + 0.5w_1^{(0)} + 0.1w_2^{(0)}$	$0 + 0.5w_1^{(0)} + 0.3w_2^{(0)}$	
α_2^1	$2 + 0 \quad + 0.4w_2^{(0)}$	$5 + 0.2w_1^{(0)} + 0 \quad w_2^{(0)}$	componente$\Gamma^1(w_1^{(0)}, w_2^{(0)})$

	β_1^2	β_2^2	
α_1^2	$2 + 0.1w_1^{(0)} + 0.3w_2^{(0)}$	$9 + 0 \quad w_1^{(0)} + 0 \quad w_2^{(0)}$	
α_2^2	$0 + 0.2w_1^{(0)} + 0.2w_2^{(0)}$	$1 + 0.4w_1^{(0)} + 0.2w_2^{(0)}$	componente$\Gamma^2(w_1^{(0)}, w_2^{(0)})$

Indicheremo con $w_1^{(1)}$ il valore del gioco $\Gamma^1(w_1^{(0)}, w_2^{(0)})$ quindi :

$$w_1^{(1)} = \text{val. } \Gamma^1(w_1^{(0)}, w_2^{(0)})$$

$$w_2^{(1)} = \text{val. } \Gamma^2(w_1^{(0)}, w_2^{(0)})$$

Procedendo per "backward induction" otteniamo che, per $0 \le s \le n-1$

$$w_k^{(s+1)} = \text{val. } \Gamma^k(w_1^{(s)}, w_2^{(s)})$$

Se n è lo stadio in cui viene troncato il gioco il valore del gioco che parte con il componente Γ^1 sarà dato da $w_1^{(n)} = \text{val. } \Gamma^1(w_1^{(n-1)}, w_2^{(n-1)})$.
Se indichiamo con T una *funzione di trasformazione* che associa ad ogni coppia di valori $(w_1^{(0)}, w_2^{(0)})$ i rispettivi $[\text{val. } \Gamma^1(w_1^{(0)}, w_2^{(0)}), \text{val. } \Gamma^2(w_1^{(0)}, w_2^{(0)})]$ possiamo definire per induzione :

$$T^1 (w_1^{(0)}, w_2^{(0)}) = (w_1^{(1)}, w_2^{(1)})$$

$$T^2 (w_1^{(0)}, w_2^{(0)}) = T(w_1^{(1)}, w_2^{(1)}) = (w_1^{(2)}, w_2^{(2)})$$

$$\dots \quad \dots \quad \dots \quad \dots$$

$$T^n (w_1^{(0)}, w_2^{(0)}) = (w_1^{(n)}, w_2^{(n)})$$

Shapley ha dimostrato che :

1. Al divergere di n , la *funzione di trasformazione* $T^n(w_1^{(0)},w_2^{(0)})$ ha un limite che è indipendente dalla coppia di valori $(w_1^{(0)},w_2^{(0)})$. Indichiamo tale coppia con (w_1^*,w_2^*) per cui :

$$\lim_{n \to \infty} T^n(w_1^{(0)},w_2^{(0)}) = (w_1^*,w_2^*)$$

2. (w_1^*,w_2^*) è l'unica coppia di valori tale che $T(w_1^{(*)},w_2^{(*)}) = (w_1^*,w_2^*)$ questo implica che (w_1^*,w_2^*) è l'unica soluzione del sistema

$$w_1 = \text{val. } \Gamma^1(w_1,w_2)$$

$$w_2 = \text{val. } \Gamma^2(w_1,w_2)$$

Generalizzando : in un gioco ad r componenti, $(w_1{}^*, w_2{}^*, ..., w_r{}^*)$ è l'unica soluzione del sistema di equazioni.

$$w_k = \text{val. } \Gamma^k(w_1, w_2, ..., w_r) \quad \text{per } k = 1,2,...,r.$$

3. Il giocatore "1" dispone di una strategia maximin per il gioco stocastico, la quale consiste nel giocare una strategia maximin per il gioco $\Gamma^k(w_1^*,w_2^*)$ ogni volta che il gioco inizia con il componente $\Gamma^{k.}$

Questa strategia ottima mista gli garantisce un pay-off di almeno w_k^* per il gioco definito come {♏,$\Gamma^{k.}$}.

Il ragionamento è speculare per il giocatore "2", il quale adotterà una strategia minmax.

Esempio :

I giocatori A e B dispongono in tutto di 5 monete.

A seconda della componente da cui partirà il gioco, avremo delle diverse situazioni di partenza, (Γ^1 = 1 moneta per A e 4 monete per B : Γ^2 = 2 monete per A e 3 monete per B , ecc.).

Γ^1 =

$3+\frac{1}{2}\Gamma^4$	-1
-1	$1+\frac{1}{2}\Gamma^2$

Γ^2 =

3	-2
-2	$1+\frac{1}{2}\Gamma^3$

Γ^3 =

2	$-2+\frac{1}{2}\Gamma^1$
$-2+\frac{1}{2}\Gamma^1$	$1+\frac{1}{2}\Gamma^4$

Γ^4 =

1	$-2+\frac{1}{2}\Gamma^2$
$-2+\frac{1}{2}\Gamma^2$	1

Ad ogni stadio del gioco il giocatore A sceglie "testa" o "croce", altrettanto farà il giocatore B all'insaputa della scelta dell' avversario.

Supponiamo che il gioco inizi con il componente Γ^1 : se le scelte coincidono e sono entrambe "testa" il giocatore B paga ad A 3 monete e verrà lanciata una moneta per stabilire se il gioco termina o prosegue. Se le scelte coincidono in "croce" allora B paga ad A una moneta e sempre con p = ½ viene stabilito se il gioco deve continuare.

In caso di non coincidenza delle scelte, A pagherà a B l'unica moneta che ha, e il gioco necessariamente termina poichè A rimane senza soldi.

Possiamo risolvere il gioco in due modi :

1. Individuando il punto fisso della funzione di trasformazione, attraverso un certo numero di reiterazioni, e fermando il processo quando i valori raggiungono una sufficiente approssimazione al limite.

Poichè il vettore iniziale $(w_1^{(0)},w_2^{(0)}\ w_3^{(0)},w_4^{(0)})$ è indipendente dal valore verso cui converge la reiterazione della funzione di trasformazione, possiamo iniziare la procedura assegnando al vettore $(w_1^{(0)},w_2^{(0)}\ w_3^{(0)},w_4^{(0)})$ dei valori arbitrari, ad esempio : (0, 0, 0, 0).

Quindi :

$T^1\ (w_1^{(0)},w_2^{(0)}\ w_3^{(0)},w_4^{(0)}) = \underline{\mathbf{w}}^{(1)} = (0.33, -0.13, -0.29, -0.50)$

$T^2\ (w_1^{(0)},w_2^{(0)}\ w_3^{(0)},w_4^{(0)}) = \underline{\mathbf{w}}^{(2)} = (0.26, -0.19, -0.29, -0.53)$

$T^3\ (w_1^{(0)},w_2^{(0)}\ w_3^{(0)},w_4^{(0)}) = \underline{\mathbf{w}}^{(3)} = (0.26, -0.19, -0.31, -0.55)$

$T^4\ (w_1^{(0)},w_2^{(0)}\ w_3^{(0)},w_4^{(0)}) = \underline{\mathbf{w}}^{(4)} = (0.26, -0.19, -0.32, -0.55)$

Dopo poche iterazioni abbiamo potuto stabilire i valori dei 4 giochi componenti con un approssimazione alla prima cifra decimale.

Il vettore $\underline{\mathbf{w}}^{(4)}$, ci permette anche di ottenere le strategie maxmin e minmax per i giocatori *A* e *B* individuando le strategie miste che risolvono i giochi :

$\Gamma^1 =$

2.72	-1
-1	0.91

$\Gamma^2 =$

3	-2
-2	0.84

$\Gamma^3 =$

2	-1.87
-1.87	0.72

$\Gamma^4 =$

1	-2.10
-2.10	1

Le soluzioni saranno :

$x_1 = (0.34, 0.66)$ $y_1 = (0.34, 0.66)$

$x_2 = (0.38, 0.62)$ $y_2 = (0.38, 0.62)$

$x_3 = (0.40, 0.60) \qquad y_3 = (0.40, 0.60)$

$x_4 = (0.50, 0.50) \qquad y_4 = (0.50, 0.50)$

Abbiamo ottenuto quella che viene definita *Strategia stazionaria ottimale* in un gioco stocastico.

2. Il secondo modo per risolvere il gioco è quello di risolvere il sistema di equazioni funzionali:

$$\begin{cases} w_1 = \operatorname{val}\begin{bmatrix} 3+\frac{1}{2}w_4 & -1 \\ -1 & 1+\frac{1}{2}w_2 \end{bmatrix} \\ w_2 = \operatorname{val}\begin{bmatrix} 3 & -2 \\ -2 & 1+\frac{1}{2}w_3 \end{bmatrix} \\ w_3 = \operatorname{val}\begin{bmatrix} 2 & -2+\frac{1}{2}w_1 \\ -2+\frac{1}{2}w_1 & 1+\frac{1}{2}w_4 \end{bmatrix} \\ w_4 = \operatorname{val}\begin{bmatrix} 1 & -2+\frac{1}{2}w_2 \\ -2+\frac{1}{2}w_2 & 1 \end{bmatrix} \end{cases}$$

Abbiamo utilizzato il termine *Strategia stazionaria ottimale* per distinguerla da altre due tipologie considerate in letteratura , in particolare :

Una strategia stazionaria assegna ad ogni gioco componente una strategia mista la quale resta invariata anche negli stadi successivi del gioco.

Esistono però anche nozioni più complesse di strategia:

Si dice *Strategia markoviana* una strategia la quale per mezzo di una funzione che ha per dominio l'insieme dei numeri naturali, assegna ad ogni stadio successivo del gioco una differente distribuzione di strategie miste sui giochi componenti.

Il tipo più complesso di strategia è la *Strategia di comportamento*, la quale per impostare la strategia mista da adottare nei vari giochi componenti,

prende in considerazione la "storia del gioco", cioè gli i giochi componenti che si sono susseguiti nei vari stadi e le scelte adottate dai giocatori.
Si può notare come le strategie Markoviane siano un sottoinsieme di quelle di comportamento, e a loro volta le strategie stazionarie siano un sottoinsieme delle Markoviane.

3. Giochi ricorsivi

Sono una categoria molto simile a quella dei giochi stocastici. Gli elementi che li contraddistinguono sono i seguenti :

1. Per i giochi ricorsivi, la probabilità che il gioco si fermi ad un certo stadio, non è necessariamente positiva.
2. Il pay-off viene eseguito solamente quando il gioco termina.

Nei giochi stocastici, il pay-off era indicato da :

$$a_{ij}^{k} \; \& \; (p_{ij}^{k0} S, p_{ij}^{k1} \Gamma^{1}, p_{ij}^{k2} \Gamma^{2}, \ldots, p_{ij}^{kn} \Gamma^{n})$$

dove

$$p_{ij}^{k0} > 0,\; p_{ij}^{kl} \geq 0 \text{ per } l = 1,2,..,n \text{ e } \sum_{l=1}^{n} p_{ij}^{kl} < 1$$

Nei giochi ricorsivi il pay-off è invece dato da :

$$p_{ij}^{k0} (a_{ij}^{k} \; \& \; S),\; p_{ij}^{k1} \Gamma^{1},\; p_{ij}^{k2} \Gamma^{2}, \ldots, p_{ij}^{kn} \Gamma^{n}$$

dove

$$p_{ij}^{k0} \geq 0,\; p_{ij}^{kl} \geq 0 \text{ per } l = 1,2,..,n \text{ e } \sum_{l=0}^{n} p_{ij}^{kl} = 1$$

La principale difficoltà nel trattare questi giochi, è dovuta al fatto che la soluzione non può essere ottenuta, in linea di massima, come soluzione di un gioco troncato dopo un numero elevato di stadi. Infatti, poichè esiste la possibilità che il gioco si prolunghi all'infinito, non esiste un numero *n* di iterazioni tale che gli effetti del troncamento sul gioco, siano trascurabili sulla computazione del valore del gioco stesso.

Il primo ad occuparsi di giochi ricorsivi fu Everett (1954) il quale dimostrò che ogni gioco ricorsivo ha una soluzione. Prima di riportare le conclusioni di Everett è opportuno illustrare alcuni esempi che mettono in luce le peculiarità dei giochi ricorsivi, permettendo di comprenderne l'essenza:

Esempio 1:

	β_1^1	β_2^1	
α_1^1	Γ^1	1	comp Γ^1

In questo gioco ricorsivo, il giocatore *1* ha a disposizione una sola strategia, risulta ovvio che al giocatore *2* non converrà mai scegliere β_2^1 in quanto gli procurerebbe una sicura perdita di *1*. Al giocatore *2* converrà invece scegliere sempre β_1^1 in modo da iterare all'infinito il gioco, senza dovere nulla al proprio avversario.
L'esempio, indica chiaramente come in un gioco ricorsivo esiste la possibilità che esso si ripeta all'infinito: quando questo avviene, diremo che il valore del gioco è nullo.

Esempio 2 :

Questo esempio illustra il motivo per cui la logica utilizzata per risolvere i giochi stocastici non si adatta, in generale, alla soluzione di un gioco ricorsivo.

	β_1^1	β_2^1	
α_1^1	Γ^1	1	
α_2^1	1	0	comp Γ^1

La soluzione proposta nell'analisi dei giochi stocastici, si ottiene trovando quel valore w tale che : $w = \text{val.}\begin{bmatrix} w & 1 \\ 1 & 0 \end{bmatrix}$

Risolvendo l'equazione funzionale si ottiene W*=1, e da cio si deduce che la strategia maxmin per il giocatore *1* , consisterebbe nel giocare la strategia pura α_1^1 . Ma questo indurrebbe l'avversario a giocare sistematicamente β_1^1 in modo da iterare il gioco all'infinito, evitando di pagare 1.

Quanto sopra induce il giocatore *1* ad accantonare l'idea di utilizzare una strategia pura, giocando invece una strategia mista del tipo $[(1-\varepsilon)\alpha_1^1, \varepsilon\alpha_2^1]$; in questo modo può garantirsi un valore atteso che si avvicina ad 1 quanto più ε è ridotto. Tale strategia viene definita come "*strategia* ε*-maxmin*" o "*ε-ottimale*".

Esempio 3

$$\begin{array}{c|c|c|}
 & \beta_1^1 & \beta_2^1 \\ \hline
\alpha_1^1 & \Gamma^1 & \Gamma^1 \\ \hline
\alpha_2^1 & \Gamma^2 & 20 \\ \hline
\alpha_3^1 & 20 & \Gamma^2 \\ \hline
\end{array} \quad \text{comp } \Gamma^1 \qquad\qquad \begin{array}{c|c|}
 & \beta_1^2 \\ \hline
\alpha_1^2 & -10 \\ \hline
\end{array} \quad \text{comp } \Gamma^2$$

Poichè è immediato constatare che il valore del secondo gioco componente è = (-10), ponendo nel primo gioco componente la condizione Γ^2 = -10, si possono facilmente individuare le strategie maxmin e minmax per i giocatori *1* e *2* , che sono rispettivamente $(0\,\alpha_1^1, 0.5\alpha_2^1, 0.5\alpha_3^1)$ e $(0.5\beta_1^1, 0.5\beta_2^1)$.
Il valore del gioco sarà +5.
Se invece cercassimo di ottenere la soluzione seguendo la strada suggerita dalla teoria dei giochi stocastici, cioè individuandola come punto fisso della funzione di trasformazione T $(w_1^{(0)},w_2^{(0)})$, ci troveremo di fronte ad una spiacevole sorpresa.
Infatti, risolvendo il seguente sistema:

$$w_1 = \text{val}\begin{bmatrix} w_1 & w_1 \\ w_2 & 20 \\ 20 & w_2 \end{bmatrix}, \qquad w_2 = \text{val}\left[-10\right] = -10$$

osserviamo che esso ha infinite soluzioni, del tipo $(w_1, -10)$ per ogni $w_1 \geq 5$.
Inoltre se volessimo osservare il processo di convergenza verso la soluzione del gioco, partendo da $(w_1^{(0)},w_2^{(0)})$ = (0, 0) otteremo che T $(w_1^{(0)},w_2^{(0)})$ = (10, -10), e iterando : $T^2 (w_1^{(0)},w_2^{(0)})$ = T (10, -10) = (10, -10) e così via.
Alla fine constatiamo che T^n (0, 0) converge sulla coppia di valori (10, -10) invece che sulla soluzione del gioco da noi sopra indicata (5, -10).

L'esempio serve a farci trarre delle importanti considerazioni sulle caratteristiche dei giochi ricorsivi :

1. Un punto fisso[35] della funzione di trasformazione T , non coincide necessariamente con il valore del gioco.
2. Non esiste necessariamente un unico punto fisso, quindi a differenza dei giochi stocastici, non è sempre vero che partendo da un qualunque vettore di valori, una sequenza di reiterazioni della funzione di trasformazione converga al valore del gioco.

Giunti a questo punto possiamo riportare le conclusioni di Everett.

1) Dato un gioco ricorsivo con i seguenti componenti $(\Gamma^1, \Gamma^2, ..., \Gamma^r)$.

Il vettore $(w^1, w^2, ..., w^r)$ si dirà *ottenibile* da parte del giocatore *1* se la funzione di trasformazione $T(w^1, w^2, ..., w^r) = (w^{1\prime}, w^{2\prime}, ..., w^{r\prime})$,

in cui : $w^{k\prime} = \text{val}\,\Gamma^k(w^1, w^2, ..., w^r)$ per $k = 1, 2, ..., r$

è tale che :

$$\left.\begin{array}{lll} w'_k > w_k & \text{ogni volta che} & w_k > 0 \\ w'_k \geq w_k & \text{ogni volta che} & w_k \leq 0 \end{array}\right. \quad \text{per } K = 1, 2, .., r$$

Se il vettore $(w^1, w^2, ..., w^r)$ è *ottenibile* e il gioco inizia con Γ^i , allora giocando una strategia maxmin in $\Gamma^k(w^1, w^2, ..., w^r)$ ogni volta che il componente k ha luogo (k = 1, 2, ..., r), il giocatore *1* può garantirsi un valore atteso di almeno w^i .

[35] Sia $F: R^n \to R^n$. Un punto fisso di F è un punto $x \in R^n$ tale che $F(x) = x$

Analogamente per il giocatore *2*, il vettore $(w^1, w^2, ..., w^r)$ si dirà *ottenibile* se

$$\left.\begin{array}{llll} w'_k & < w_k & \text{ogni volta che} & w_k < 0 \\ w'_k & \leq w_k & \text{ogni volta che} & w_k \geq 0 \end{array}\right. \quad \text{per } K = 1, 2, .., r$$

Se il vettore $(w^1, w^2, ..., w^r)$ è *ottenibile* e il gioco inizia con Γ^i, allora giocando una strategia minmax in $\Gamma^k(w^1, w^2, ..., w^r)$ ogni volta che il componente k ha luogo (k = 1, 2, ..., r), il giocatore *2* può garantirsi un valore atteso al più di w^i.

Il vettore $(w_1^{(0)}, w_2^{(0)}, ..., w_r^{(0)})$ si dice *vettore critico* se per ogni $\varepsilon > 0$ piccolo a piacere

a) Esiste un altro vettore *ottenibile* da parte del giocatore *1*, i cui componenti sono ad uno ad uno prossimi a $(w_1^{(0)}, w_2^{(0)}, ..., w_r^{(0)})$ a meno di un ε.
b) Esiste un altro vettore *ottenibile* da parte del giocatore *2*, i cui componenti sono ad uno ad uno prossimi a $(w_1^{(0)}, w_2^{(0)}, ..., w_r^{(0)})$ a meno di un ε.

Se esiste, il *vettore critico* è unico viene inteso come il valore del gioco ricorsivo.

2) Se il *vettore critico* esiste, allora è un punto fisso della funzione di trasformazione T, cioè $T(w_1^{(0)}, w_2^{(0)}, ..., w_r^{(0)}) = (w_1^{(0)}, w_2^{(0)}, ..., w_r^{(0)})$.

3) Ogni gioco ricorsivo possiede un vettore critico.

La strategia che corrisponde al vettore critico viene definita da Everett strategia *ε–ottimale.*

Esempio 4

(*Everett 1954*) Il colonnello Blotto è a capo di un avamposto costituito da 3 unità militari. Il suo compito è quello di conquistare un'appostamento nemico, distante 10 miglia, che è occupato da due unità.

Blotto otterrà un pay-off di +1 se riuscirà a conquistare la base nemica senza perdere la propria, mentre avrà un pay-off di -1 se perde il proprio avamposto, a prescindere dalla conquista o meno di quello nemico.

Gli attacchi si svolgono di notte e entrambi gli eserciti stabiliscono l'uno all'insaputa dell'altro quante unità lasciare a difesa della propria base, e quante mandare all'attacco. Un attacco contro il nemico ha successo solo se le truppe che attaccano dispongono di almeno un unità in più del difensore ; in caso opposto non succede nulla e il gioco si ripete nelle stesse condizioni.

Questa è la rappresentazione del gioco ricorsivo :

				NEMICO	
			Att. 0	Att. 1	Att. 2
			Dif 2	Dif 1	Dif 0
	Att.	Dif.			
BLOTTO	0	3	Γ	Γ	Γ
BLOTTO	1	2	Γ	Γ	1
BLOTTO	2	1	Γ	1	-1
BLOTTO	3	0	1	-1	-1

Il valore del gioco tende a +1, e la strategia ε-*ottimale* per Blotto (come ricavata nell'articolo sopra citato) è costituita dal vettore $(0,\ 1-\varepsilon-\varepsilon^2,\ \varepsilon,\ \varepsilon^2)$.
E' interessante osservare che quanto più piccolo risulta ε, tanto più risulta elevata la probabilità di vittoria per Blotto, e quanto più lunga sarà la durata attesa del gioco. In un certo senso viene premiata la "pazienza" del giocatore.
L'esempio risulta utile per meglio comprendere il significato di una strategia ε-*ottimale*.

4. Posizione critica nel finale R+C+A contro R in kriegspiel. Risolta come gioco stocastico.

In questo esempio, viene dimostrato come gli strumenti appena illustrati della teoria dei giochi, possano essere utilizzati per risolvere, con razionalità sostanziale, un sottoproblema nel Kriegspiel.

Recentemente il matematico californiano T.Ferguson (*Ferguson 1992*) ha dimostrato che nel kriegspiel, è possibile forzare la vittoria con probabilità 1 in un finale in cui re, alfiere e cavallo devono dare scacco matto al re avversario, che è l'unico pezzo nemico.

Tale problema era rimasto insoluto per molti anni, nonostante l'interesse che questo gioco suscita in ambito matematico: lo stesso Shapley, ideatore dei giochi stocastici, è autore di una raccolta di problemi di kriegspiel.

La dimostrazione di Ferguson è costruttiva, essa in molti dei suoi passi richiede che per forzare la vittoria, si debba utilizzare una strategia mista.

Esaminiamo ora un passo esemplificativo della dimostrazione :

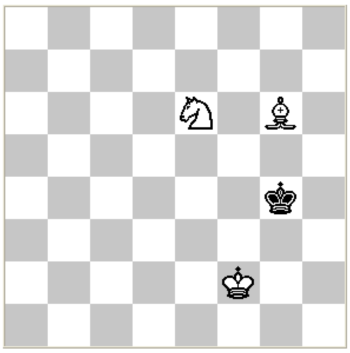


Fig 3.2

Data la posizione iniziale (Fig 3.2), nota ad entrambi i giocatori, e <u>con il tratto al nero</u>, dobbiamo cercare una strategia ottimale per il bianco, che sia in grado giungere nel modo più efficiente alla posizione indicata in figura 4.2 oppure in fig.5.2.

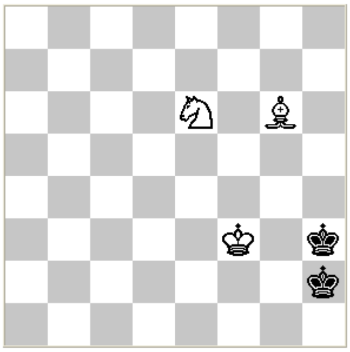


fig 4.2

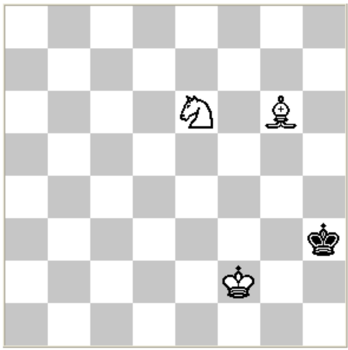


fig 5.2

Prendiamo per noto che, qualora riuscissimo a finire nella posizione 4.2 la vittoria sarebbe assicurata con 24 mosse, mentre nella posizione 5.2 basterebbero 18 mosse (*Ferguson 1992*).

Premesso questo, il nostro obiettivo sarà quello di minimizzare il numero di mosse necessarie per portarsi dallo stadio della fig 3.2 ad uno degli stadi successivi.

			°	°	°			8
		°			°	°	°	7
				C		A		6
		°			°	°	°	5
			°	°	°	X		4
			°	°	°	°		3
		°		°	R	°		2
	°			°	°	°		1
a	b	c	d	e	f	g	h	

Il simbolo ° indica le caselle controllate da uno dei pezzi bianchi. Il re nero non può andare in queste caselle perchè sarebbe sotto scacco. Si può facilmente vedere come le uniche mosse possibili siano Rh4 o Rh3.

fig 6.2

Poichè il tratto spetta al nero, esso può optare tra due alternative: muovere in h3 o in h4, tutte le altre mosse sono illegali, infatti con un minimo di attenzione si percepisce come il Re sia chiuso in una "gabbia immaginaria" (Fig 6.2).

Dopo la mossa del nero il bianco dispone di un set di ben 25 scelte alternative[36] e si trova in un contesto di incertezza per quanto riguarda la posizione del re avversario, che potrà essere in h3 o in h4 (Fig.7.2).

								8
								7
				C		A		6
								5
							X	4
							X	3
					R			2
								1
a	b	c	d	e	f	g	h	

Set informativo del bianco dopo la prima mossa del nero.
fig 7.2

A questo punto è opportuno semplificare l'albero di gioco, eliminando dallo stesso tutti gli archi corrispondenti a mosse che presentano una o più delle seguenti caratteristiche :

- Rischiano che un pezzo possa essere catturato. Ad esempio la mossa Ah5, permetterebbe al re nero, qualora fosse in h4, di catturare il pezzo bianco.
- Non portano alcuna nuova informazione al set informativo esistente. Ad esempio la mossa Rg3 è inutile, in quanto posso già dedurre che sarà illegale sia nell' ipotesi che il re nero sia in h3 che in h4.
- Aumentano l'incertezza sulla possibile collocazione del re. Ad esempio la mossa Re3 è da scartare, in quanto aprendo la "gabbia immaginaria" ,

[36] anche il nero teoricamente avrebbe avuto un set di 8 alternative, ma conoscendo la posizione iniziale, sapeva che 6 di queste sarebbero state illegali

porterebbe ad una situazione in cui allo stadio successivo l'incertezza sulla collocazione del re sarebbe di gran lunga maggiore. (Fig 8.2).

								8
								7
				C		A		6
								5
						X	X	4
				R		X	X	3
						X	X	2
								1
a	b	c	d	e	f	g	h	

Dopo la mossa Re3 e la replica del nero, il set informativo del bianco sarebbe molto più complesso e con una maggiore incertezza sulla collocazione del re nemico.
fig 8.2

Dopo aver eseguito questa semplificazione euristica, dal set di 25 scelte ne rimangono solamente 2, sono : Rg2 e Rf3

Vediamo cosa succede nell'ipotesi che vengano giocate:

Rg2

Ipotesi 1 : L' arbitro annuncia che la mossa è legale. Questo implica che il Re nero era in h4. Dato che una mossa legale va giocata obbligatoriamente, alla replica il Re nero andrà forzatamente in g4 (Fig 9.2).

Poichè non desidero che la complessità del gioco mi sfugga di mano, dopo la replica del nero, risponderò Rf2 e il problema si riproporrà come nella posizione di fig 3.2 con la differenza che avrò utilizzato due mosse in più per risolverlo.

			◦	◦	◦			8
		◦			◦	◦	◦	7
				C		A		6
		◦			◦	◦	◦	5
			◦	◦	◦		X	4
			◦		◦	◦	◦	3
		◦			◦	R	◦	2
	◦				◦	◦	◦	1
a	b	c	d	e	f	g	h	

Dopo la mossa Rg2, la "gabbia immaginaria" costringe il re nero a portarsi in g4.

fig 9.2

Ipotesi 2 : L' arbitro annuncia che la mossa è illegale e potrò tentare un'altra mossa, ma a questo punto ho acquisito un importante informazione, "il Re avversario è sicuramente in h3", (perchè se fosse stato in h4 saremmo nell'ipotesi 1). Siamo riusciti a condurre il gioco nello stadio successivo di fig 5.2, da qui sappiamo di vincere sicuramente in 18 mosse (vedi *Ferguson 1992*).

Rf3 la mossa è sempre legale, in quanto non cerca di occupare case potenzialmente controllate dal re avversario, in ogni caso si possono formulare anche qui due ipotesi:

			◦	◦	◦			8
		◦			◦	◦	◦	7
				C		A		6
		◦			◦	◦	◦	5
			◦	◦	◦	◦	X	4
			◦	◦	R	◦		3
		◦		◦	◦	◦		2
	◦							1
a	b	c	d	e	f	g	h	

L' unica mossa possibile per il nero è : Rh3.

fig 10.2

Ipotesi 1 : Il re nero era in h4, e dovrà forzatamente spostarsi in h3(Fig 10.2).
la mia replica sarà Rg2, la quale se illegale, mi darà conferma certa della presenza del Re in h3: saremo nello stadio della fig 4.2. con vittoria certa in 24 mosse.

Ipotesi 2 : Il re nero era in h3, sceglierà sicuramente di spostarsi in h4[37] , la mia replica Rg2, sarà legale e mi darà conferma che il re è in h4.
Dopo la mia Rg2, il nero risponderà forzatamente Rg4, e il gioco si riproporrà come nelle condizioni iniziali di fig. 3.2. In questo caso però,avro utilizzato 3 mosse in più per risolverlo.

Le sopraindicate deduzioni logiche, permettono di ricondurre i termini del problema ad una rappresentazione del gioco in forma estesa (Fig 11.2).
In particolare, partendo dalla posizione di fig 3.2 emergono due strategie proponibili sia per il bianco che per il nero. Per primo deve decidere il nero, il quale può scegliere tra Rh3 e Rh4; risponde il bianco che non è a conoscenza della strategia dell'avversario (informazione imperfetta), può comunque optare in risposta tra Rg2 e Rf3.

[37] Se andasse in h2, si porterebbe nella posizione di fig. 4.2, "regalando al bianco la possibilità di vincere in 24 mosse" (*Ferguson 1992*).

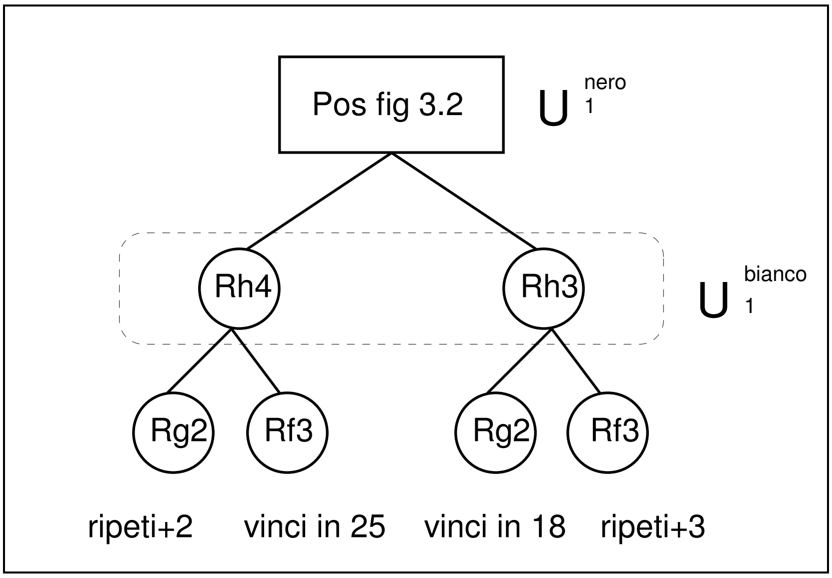


Fig 11.2

L'esito del gioco dipende dall' incontro tra i due profili di strategie.

La fig 12.2 illustra la rappresentazione del gioco in forma normale.

		Bianco	
		Rg2	Rf3
Nero	Rh3	18	$\Gamma + 3$
	Rh4	$\Gamma + 2$	25

fig 12.2

Il fatto che in corrispondenza delle strategie (Rg2 & Rh3) il pay-off sia 18, significa che il bianco vincerà in 18 mosse con probabilità 1.

Il pay-off indicato con Γ + 3 significa che il gioco si ripeterà nelle stesse condizioni di fig 3.2, ma con 3 mosse in più : se il gioco terminasse alla successiva iterazione, il pay-off in corrispondenza di (Rg2 & Rh3), non sarà più 18, ma 21.

E' immediato constatare che l'obiettivo del bianco è quello di minimizzare il numero di mosse per vincere, mentre l'obiettivo del nero è esattamente l'opposto[38].

Il gioco non ammette equilibrio in strategie pure: se infatti il bianco giocasse sistematicamente Rg2, il nero sapendo questo, giocherebbe sempre Rh4, prolungando il gioco all'infinito.

Si può dimostrare chesiste per il bianco una strategia ottima mista che gli garantisce un valore del gioco "v". Risolvendo il gioco otteniamo tale strategia ottima mista che impone di giocare Rg2 con p = 0.2785 e Rf3 con p = 0.7215, con un valore atteso di circa 26 mosse per vincere.

Ripercorriamo la strada seguita per arrivare a tali conclusioni.

Nella Fig. 13.2 viene illustrata la rappresentazione del gioco in forma normale.

	β_1^1	β_2^1
α_1^1	18	$\Gamma^1 + 3$
α_2^1	$\Gamma^1 + 2$	25

fig 13.2

Dobbiamo innanzitutto stabilire se esso va considerato come gioco ricorsivo o come gioco stocastico.

Gioco ricorsivo non può definirsi, in quanto dalla definizione di Everett (*Everett 1954*) risulta che il pay-off è costante, e viene pagato nel momento in cui il gioco ha termine. Nel nostro caso, invece, il pay-off incrementa ogni volta che il gioco viene reiterato.

[38] Si tratta dunque di un gioco a somma zero.

Il gioco posto come nella fig 13.2 non rientra completamente nemmeno nella definizione di gioco stocastico introdotta da Shapley (*Shapley 1953*). Infatti secondo questa definizione tutti i possibili esiti del gioco devono prevedere una probabilità di arresto strettamente maggiore di 0, mentre il nostro gioco, in corrispondenza delle strategie (α_1^1 & β_2^1) e (α_2^1 & β_1^1), prosegue con probabilità 1.

Nonostante questo, possiamo far sì che il gioco di fig 13.2 rientri a tutti gli effetti nella categoria dei giochi stocastici inserendo in ogni pay-off che ne è sprovvisto, una piccola probabilità di arresto (ε), la quale garantisce la correttezza dell'applicazione dell' equazione di Shapley, senza stravolgere la struttura del problema.

	β_1^1	β_2^1
α_1^1	18	$\varepsilon S + (1-\varepsilon)\Gamma^1 + 3$
α_2^1	$\varepsilon S + (1-\varepsilon)\Gamma^1 + 2$	25

A questo punto esistono due strade differenti per risolvere il gioco e individuare la strategia più efficiente da parte del bianco.

La prima è quella che individua il $\lim_{n \to \infty} T^n(w_1^{(0)}) = (w_1^*)$

attraverso un processo di reiterazione della funzione di trasformazione per un numero di volte sufficientemente elevato da garantire una approssimazione soddisfacente del valore del gioco.

Si procederà come segue :

$$\mathbf{T}^1(w_1^{(0)}) = \text{val}\begin{bmatrix} 18 & 3+w_1^{(0)} \\ 2+w_1^{(0)} & 25 \end{bmatrix} = w_1^{(1)}$$

$$\mathbf{T}^2(w_1^{(0)}) = \text{val}\begin{bmatrix} 18 & 3+w_1^{(1)} \\ 2+w_1^{(1)} & 25 \end{bmatrix} = w_1^{(2)}$$

... ...

$$\mathbf{T}^n(w_1^{(0)}) = \text{val}\begin{bmatrix} 18 & 3+w_1^{(n-1)} \\ 2+w_1^{(n-1)} & 25 \end{bmatrix} = w_1^{(n)}$$

Poichè w_1* è indipendente da $w_1^{(0)}$ possiamo assegnare a $w_1^{(0)}$ un valore arbitrario, che per comodità sarà 18, (il numero di mosse minimo per vincere)

Alla prima iterazione avremo :

$$\mathbf{T}^1(w_1^{(0)}) = \text{val}\begin{bmatrix} 18 & 3+\cong 18 \\ 2+\cong 18 & 25 \end{bmatrix} = \cong\ 21 \quad (^{39})$$

$$\mathbf{T}^2(w_1^{(0)}) = \text{val}\begin{bmatrix} 18 & 3+\cong 21 \\ 2+\cong 21 & 25 \end{bmatrix} = \cong\ 24$$

$$\mathbf{T}^3(w_1^{(0)}) = \text{val}\begin{bmatrix} 18 & 3+\cong 24 \\ 2+\cong 24 & 25 \end{bmatrix} = \cong\ 25.2$$

E dopo non molte iterazioni si noterà che il limite per $n \to \infty$ tende a $\cong 25.779...$

La seconda strada passa attraverso la soluzione dell'equazione di Shapley: sappiamo che esiste un valore w* tale che :

$$\text{w}* = \text{val.}\begin{bmatrix} 18 & w*+3 \\ w*+2 & 25 \end{bmatrix}$$

[39]Abbiamo indicato $\cong 18$ in quanto va inteso come prodotto tra 18 ed (1-ε), cioè la probabilità. che il gioco continui.

w* è l'incognita di un' equazione che verrà risolta con l'ausilio del seguente teorema:

Teorema : Sia A un gioco in forma normale rappresentato con una matrice 2x2.
Se A non ha un punto di sella, le uniche strategie ottime e il valore del gioco saranno rispettivamente :

$$\bar{x} = \frac{JA^*}{JA^*J'} \qquad \text{(formula 1.2)}$$

$$\bar{y} = \frac{A^*J'}{JA^*J'} \qquad \text{(formula 2.2)}$$

$$w. = \frac{|A|}{JA^*J'} \qquad \text{(formula 3.2)}$$

In cui A* é la matrice aggiunta di A, |A| è il determinante, J è il vettore riga (1,1), J' è il vettore colonna trasposto di J.
La matrice aggiunta di A è $\begin{bmatrix} 25 & -(w+2) \\ -(w+3) & 18 \end{bmatrix}$

Il valore del gioco si otterrà risolvendo l'equazione :

$$w = \frac{(18*25) - (w+3)(w+2)}{[1,1]\begin{bmatrix} 25 & -(w+2) \\ -(w+3) & 18 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \end{bmatrix}} = \frac{450 - (w+3)(w+2)}{[22-w \quad 16-w]\begin{bmatrix} 1 \\ 1 \end{bmatrix}} = \frac{450 - (w+3)(w+2)}{(38-2w)}$$

Con ovvi passaggi si ottiene la seguente equazione di 2°grado:

$$-w^2 + 43w - 444 = 0$$

che ammette le due soluzioni :

$$w_{1,2} = -43 \pm \sqrt{\frac{43^2 - 1776}{-2}} = \begin{matrix} 25.779 \\ 17.228 \end{matrix} \quad w_{1,2} = -\ 43\ \pm\ \sqrt{\frac{43^2 - 1776}{-2}}\ =\ \begin{matrix} 25.779 \\ 17.228 \end{matrix}$$

dalle quali ne eliminiamo una, in quanto priva di significato.[40]

Abbiamo quindi W* = 25.779...

Conoscendo il valore del gioco, possiamo utilizzare le formule *1.2* e *2.2* per ottenere la strategia mista ottimale del giocatore bianco, che è :

β_1^1 con p = 0.7125 e β_2^1 con p = 0.2875.

Tale strategia è la migliore in senso assoluto, nel senso che permette al bianco di vincere in media nel minor numero di mosse possibile. La strategia è dunque una strategia ottima minima con riferimento al teorema di Von Neumann.

[40] La soluzione 17.228 è priva di significato dato che il valore è minore di 18 : il numero minimo di mosse per vincere nella migliore delle ipotesi.

5. Posizione critica nel finale di Re e pedone contro Re in kriegspiel, come gioco ricorsivo.

Nel gioco degli scacchi tradizionali, (ad informazione perfetta) si è ormai da molto tempo giunti ad una conoscenza completa ed esaustiva di ogni posizione in cui re e pedone giocano contro il solo re avversario. Un giocatore con un minimo di conoscenza teorica è in grado di stabilire se una data posizione, a gioco corretto, sia vinta per chi ha maggiori forze in campo oppure patta[41]. Occorre sottolineare, comunque che tale risultato, non si sia ottenuto risolvendo il gioco per *backward induction*, ma attraverso una procedura di *pruning*, con la quale partendo dalla radice dell'albero vengono "potati" i rami che conducono a sottoalberi insoddisfacenti.

Negli scacchi esistono dei casi banali, in cui il giocatore che dispone di re e pedone (d'ora in poi il bianco), riesce a giungere con il proprio pedone nell'ultima traversa senza che esso possa essere contrastato dal re avversario[42] .

D'altronde esistono altri casi in cui il re nero riesce a catturare il pedone del bianco, e in questo caso la posizione è sicuramente patta.

La fig 14.2 illustra una classica posizione nel gioco degli scacchi tradizionali.

[41] Chi ha maggiori forze in campo non può perdere, in quanto l'avversario dispone del solo re, che non è in grado di dare scacco matto.

[42] Se il pedone riesce a giungere nell'ultima traversa, il bianco vincerà sicuramente, poichè verrà sostituito con un pezzo a piacere (es. Regina) con il quale si è in grado di dare scacco matto in poche mosse.

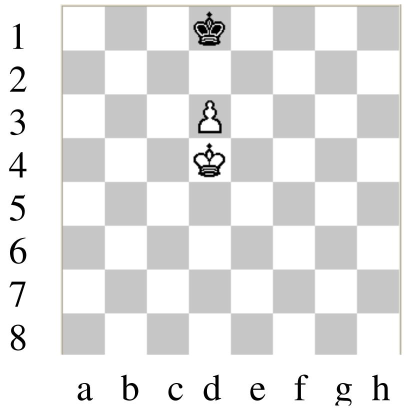


fig. 14.2

Affinchè il bianco non possa vincere il nero deve :

- Portarsi ogni volta che è possibile nella casa d7
- Se il bianco gioca Rc6 il re nero deve andare in c8
- Se il bianco gioca Re6 il re nero deve andare in e8.

Abbiamo visto come nel gioco degli scacchi questa posizione sia sicuramente patta; nel linguaggio della teoria dei giochi, abbiamo delineato una strategia minmax per il nero in grado di ottenere un valore del gioco pari a 0.

Lo stesso problema, con le regole di Kriegspiel ci porta ad una soluzione differente, che può essere individuata in modo brillante con l'aiuto della teoria dei giochi.

Supponendo di partire dalla posizione di fig. 15.2, nota ad entrambi i giocatori e con il tratto al nero, cerchiamo di dare una rappresentazione in forma estesa del gioco. Si tratta di un gioco ad informazione imperfetta, nel quale ogni insieme informativo a disposizione di ciascun giocatore, può consistere di più nodi.

La fig. 15.2 rappresenta quindi il set informativo iniziale del nero.

								8
			X					7
			P					6
			R					5
								4
								3
								2
								1
a	b	c	d	e	f	g	h	

fig 15.2

Alla prima mossa il nero ha a disposizione tre alternative : Rc8, Rd8, Re8. Dopo la mossa del nero, il bianco si troverà di fronte al set informativo indicato nella fig. 16.2.

		X	X	X				8
								7
			P					6
			R					5
								4
								3
								2
								1
a	b	c	d	e	f	g	h	

fig 16.2

Con lo stesso criterio utilizzato nell'esempio precedente, stabiliamo che le uniche alternative rilevanti da parte del bianco sono : Rc6 o Re6.

Alla seconda mossa, le alternative rilevanti per il nero sono :

1. Rd8 se la prima mossa era Rc8 o Re8.
2. Rc8 o Re8 se la prima mossa era Rd8.

Da notare l'opportunità da parte del bianco, alla seconda replica, di giocare delle mosse, (Rc7 o Re7[43]) che gli consentono di ottenere delle ulteriori informazioni in grado di semplificare e di rendere più trattabile la complessità della posizione.

Ad esempio supponiamo che il re bianco si trovi in c6 e cerchi di giocare Rc7.

Se l'arbitro dichiara che la mossa è legale ci troveremmo nella posizione di fig.17.2 con il tratto al nero. La vittoria sarebbe sicuramente del bianco[44].

				X				8
		R						7
			P					6
								5
								4
								3
								2
								1
a	b	c	d	e	f	g	h	

fig 17.2

Se invece l'arbitro dichiara che la mossa è illegale verremo a trovarci nella posizione di fig. 18.2, con il tratto al bianco. Sulla base delle nuove informazioni, il bianco ha due alternative rilevanti:

1. Spingere il pedone rischiando la patta per *stallo*[45].
2. Giocare Rd5, riproponendo il gioco nelle stesse condizioni iniziali.(fig.16.2).

---

[43] Il bianco tenterà Rc7 se alla prima replica aveva giocato Rc6, mentre tenterà Re7 se aveva giocato Re6.

[44] Il nero non potrebbe più contrastare l'avanzata del pedone bianco.

[45] Se il re nero era collocato in c8 si avrebbe patta per stallo. Se invece era in d8 il bianco vincerebbe.

		X	X					8
								7
		R	P					6
								5
								4
								3
								2
								1
a	b	c	d	e	f	g	h	

fig 18.2

La rappresentazione del gioco in forma estesa ci consente di descrivere in modo rigoroso le peculiarità di questo problema (Fig 19.2).

Occorre comunque sottolineare che l'albero del gioco non è esaustivo, in quanto è stato "potato" dai rami corrispondenti a mosse palesemente errate.

Dalla forma estesa, per poterlo risolvere in maniera più agevole, il gioco può essere trascritto in forma normale.

Il nero, che è il primo a muovere, può scegliere una tra le 4 strategie :

β_1 = **Rd8** seguita da **Rc8**

β_2 = **Rd8** seguita da **Re8**

β_3 = **Rc8** seguita da **Rd8**

β_4 = **Re8** seguita da **Rd8**

Le strategie a disposizione del bianco saranno invece :

α_1 = **Rc6** seguita da (**tenta Rc7**) seguita da **spinta**

α_2 = **Re6** seguita da (**tenta Re7**) seguita da **spinta**

α_3 = **Rc6** seguita da (**tenta Rc7**) seguita da **Rd5**

α_4 = **Re6** seguita da (**tenta Re7**) seguita da **Rd5**

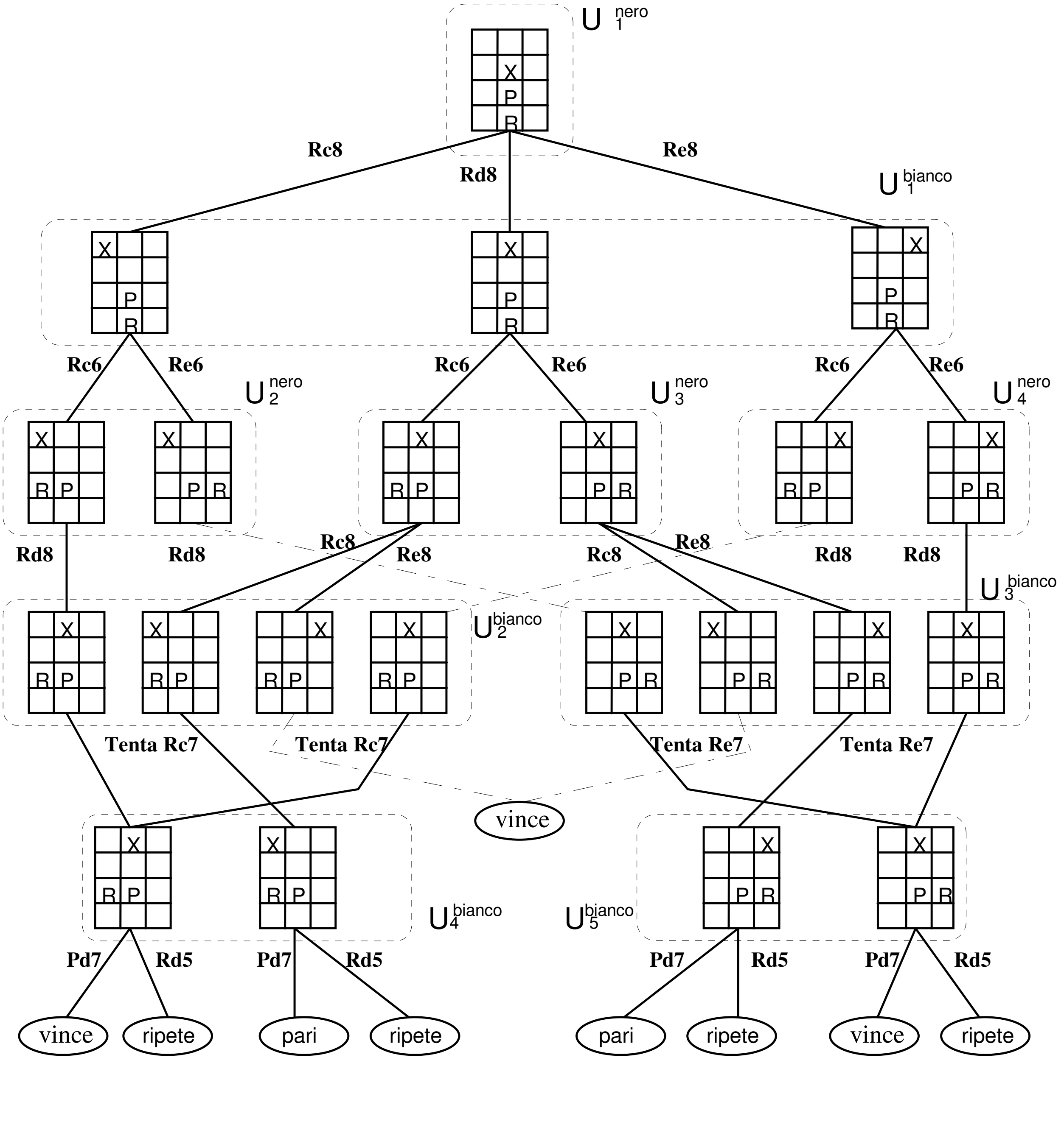


Fig 19.2 Rappresentazione in forma estesa del problema

In una tabella a doppia entrata indichiamo i profili di strategie a disposizione dei due giocatori.

	β_1	β_2	β_3	β_4
α_1	0	1	1	1
α_2	1	0	1	1
α_3	Γ	1	Γ	Γ
α_4	1	Γ	Γ	Γ

Si tratta di un gioco ricorsivo, che può essere immediatamente semplificato, osservando che le strategie β_3 e β_4 sono equivalenti, per cui avremo:

	β_1	β_2	β_3
α_1	0	1	1
α_2	1	0	1
α_3	Γ	1	Γ
α_4	1	Γ	Γ

Il bianco dispone di una strategia mista ε-*ottimale*, giocando α_3 e α_4 con probabilità rispettivamente pari a (1-ε/2) e (1-ε/2) , e giocando α_1 e α_2 con probabilità ε/2 ed ε/2.

Quanto più ε sarà piccolo, tanto più il valore del gioco tenderà ad 1, come nel gioco ricorsivo presentato da Everett[46] (Colonnello Blotto).

La soluzione può essere commentata intuitivamente: se il bianco giocasse α_3 o α_4 con probabilità 1, sarebbe certo di non perdere, ma l' avversario rendendosi conto di questo giocherebbe sistematicamente la strategia β_3

[46] Vedi par 3 pag. 65

forzando la ripetizione indefinita del gioco. Per sbloccare questo "loop", al bianco non resta che rischiare di tanto in tanto le strategia α_1 e α_2. Se queste strategie vengono giocate con probabilità ε sufficientemente piccola, la vittoria sarà quasi certa.

6. CONCLUSIONI

In questi esempi abbiamo potuto vedere come alcuni strumenti della teoria dei giochi siano stati utilizzati per risolvere con *razionalità sostanziale* alcuni problemi del kriegspiel.
In particolare i modelli dei giochi stocastici e ricorsivi si sono dimostrati efficaci ed efficienti nella ricerca delle soluzioni ; il loro limite fondamentale resta comunque l'inadeguatezza a trattare problemi di complessità più elevata.
Purtroppo non è sempre possibile, in pratica, rappresentare il gioco in forma estesa o normale in quanto l'albero è troppo complesso.
Viene allora spontaneo chiedersi quale sia l'utilità pratica di questi risultati che, anche se ottimali, sono ottenuti da un contesto estremamente semplice.
Ritengo che le conoscenze acquisite con la soluzione di questi sottoproblemi "semplici" debbano far parte della base di conoscenza di chi si accinge ad affrontare problemi più complessi con razionalità procedurale.
A mano a mano che la complessità di un problema aumenta, la razionalità sostanziale lascia il posto a quella procedurale; non per questo la prima viene accantonata, ma come vedremo in seguito, essa rimane nel "bagaglio di conoscenze" del giocatore e viene utilizzata nel momento in cui è possibile farlo.

BIBLIOGRAFIA DEL CAPITOLO 2 :

Aumann R., S.Hart : *Handbook of game theory* - North Holland (1992)

Castelli A."Chess variants , varianti scacchistiche ad informazione incompleta" : in "Eteroscacco" n. 60 (1992).

Chicco E., Porreca A. *Dizionario enciclopedico degli scacchi* (1971)

Everett H."Recusive games" : Annals of math studies 39, pag. 47-78 (1953).

Ferguson T.S."Mate with bishop and knight in kriespiel" : in Theoretical Computer Science n. 96 (1992) pag. 389-403.

Friedmann J.W. *Game theory with applications to economics* : Oxford University press (1986).

Kreps *Teoria dei giochi e modelli economici* : Il Mulino (1990)

Leoncini M., Magari R. *Manuale di scacchi eterodossi* (1978) Siena : Tipografia senese

Luce, Raiffa *Games and decisions* : Wiley, New York (1957)

Magari R. "Giochi matematici ad informazione parziale" : "Sapere" (aprile 1985)

Myerson *Game theory* : Harvard press (1991).

Naddeo A. "Prefazione" in : E.Burger (1967) *Introduzione alla teoria dei giochi*

Owen G. *Game theory* : Academic press, New York (1982)

Shapley L. "Sthocastic games" Proc.Nat.Acad.Sci. U.S.A. pp 327-332 (1953).

Thuijsman F. *Optimality and equilibria in stochastic games* : CWI Tract, Netherland (1992)

Capitolo 3

Kriegspiel e razionalità procedurale

1. Introduzione.

Nel capitolo precedente abbiamo accennato al limite fondamentale dell'approccio con razionalità sostanziale ad un gioco complesso come il kriegspiel.
I raffinati strumenti della teoria dei giochi permettono di ottenere delle soluzioni ottimali a problemi che non vanno oltre un limitato grado di complessità. La posizione critica esaminata nel par. 5.2, ad esempio, è solo una situazione tipica di un problema più complesso che è quello di gestire il finale R + P contro P partendo da una qualunque disposizione iniziale dei pezzi sulla scacchiera, ma il microcosmo del kriegspiel è ancora più complesso, il numero delle forze in campo può essere maggiore, così come l'incertezza sulla loro collocazione.
In questi contesti la ricerca di una strategia ottimale risulterebbe oltremodo dispendiosa, e anche se fosse possibile giungere a dei risultati significativi, gli sforzi impiegati non verrebbero sicuramente compensati dai risultati.
In un gioco come il kriegspiel la componente di incertezza è così pervasiva da rendere inopportuno ogni tentativo di pianificazione nel lungo periodo. La concomitanza dei fattori sopra indicati, non preclude comunque la

possibilità di un approccio razionale al gioco; anche se l'uso che facciamo del termine "razionale" si discosta molto da quello dall' analisi economica tradizionale.

In particolare, osserveremo come la *conoscenza* acquisita nei confronti di uno specifico problema sia un elemento fondamentale che caratterizza un comportamento *proceduralmente razionale*.

Se da un lato, la conoscenza viene acquisita attraverso l'esperienza e l'apprendimento, parte del bagaglio di conoscenze utilizzate per affrontare il problema viene ottenuto con un approccio con razionalità sostanziale, al quale non si può certo negare il merito di un' elevata capacità descrittiva, nonchè di trovare soluzioni ottimali in contesti semplici, che con le dovute cautele possono essere utilizzati per risolvere contesti reali assai più complessi.

Proprio uno dei risultati ottenuti con la teoria classica dei giochi (che nel nostro caso è la strategia *ε-ottimale* di Everett) viene a far parte della base di conoscenza di un modello proceduralmente razionale che affronta un problema più complesso.

Nei giochi complessi, allora, la *conoscenza* è sicuramente un fattore che condiziona il comportamento dei giocatori e di conseguenza la manifestazione di un punto di equilibrio. Ad esempio, un mediocre giocatore di scacchi, giocando contro il campione del mondo perderebbe sicuramente: esiste quindi un equilibrio reale nel gioco che fa pendere il piatto della bilancia dalla parte del soggetto che dispone di maggiori conoscenze, cioè di un modello interiorizzato più potente in grado di far fronte a quell' incertezza computazionale di cui abbiamo parlato nel primo capitolo.

La parte rimanente del capitolo illustra un approccio con razionalità procedurale al finale R + P contro R nella versione completa. Lo sbocco naturale di questo approccio è la modellizzazione di un giocatore artificiale che si *comporta razionalmente* in relazione al raggiungimento del suo obiettivo utilizzando la base di conoscenza di cui dispone.

2. Il finale R + P contro R in kriegspiel nella versione completa

Il tipo di finale che era stato affrontato nel capitolo precedente aveva delle caratteristiche ben definite :

- Il pedone bianco era collocato sulla *sesta traversa*[47]
- Il re bianco stava dietro al pedone e in una casa adiacente ad esso
- Il re nero stava davanti al pedone. (Fig 1.3)

								8
			X					7
			P					6
			R					5
								4
								3
								2
								1
a	b	c	d	e	f	g	h	

fig 1.3

Chiaramente il tipo di posizione sopra descritto rappresenta solo un limitato sottinsieme di tutti i possibili modi in cui si può presentare il problema. Nella maggior parte di queste posizioni, un approccio analogo a quello utilizzato nel capitolo precedente, risulta scoraggiante, se il nostro obiettivo è quello di giungere a dei risultati non esclusivamente descrittivi, ma anche normativi.

[47] Nel gergo scacchistico, le colonne di una scacchiera si dicono *colonne*, mentre le righe si dicono *traverse*

In qualunque posizione ci si trovi, l'obiettivo è quello di far sì che il pedone riesca a raggiungere l'ottava traversa evitando che vanga catturato dal re nero o che questo si porti in una situazione di stallo.
Illustriamo a titolo esemplificativo, alcune fra le innumerevoli posizioni in cui può presentarsi il problema.

					X	X	X	8
					X	X	X	7
			P		X	X	X	6
			R		X	X	X	5
					X	X	X	4
								3
								2
								1
a	b	c	d	e	f	g	h	

Il re nero è sicuramente collocato nell'area indicata dalle X vi è però incertezza sulla casella precisa in cui si trova.

fig 1.3.a

X								8
X	X							7
X	X			R				6
	X							5
				P				4
						X		3
					X	X	X	2
							X	1
a	b	c	d	e	f	g	h	

Re davanti al pedone e incertezza sulla collocazione del re nero che può essere in una tra due differenti aree.

fig 1.3.b

X	X	X	X	X	X	X	X	8
X		X		X	X	X	X	7
X	X	P				X	X	6
X	X	X		R		X	X	5
X	X					X	X	4
X	X	X	X	X	X	X	X	3
X	X	X	X	X	X	X	X	2
X	X	X	X	X	X	X	X	1
a	b	c	d	e	f	g	h	

Assoluta incertezza sulla collocazione del re avversario.

fig 1.3.c

3. Programma in grado di gestire il finale R+P contro R in kriegspiel[48].

Verranno ora descritte la struttura e la filosofia di base di un programma, sviluppato in ambiente LPA Prolog, in grado di gestire completamente il finale nel kriegspiel in cui re e pedone giocano contro il solo re.
All'interno del programma convivono due anime, quella del giocatore artificiale e quella dell'arbitro.
<u>Il giocatore artificiale</u> può essere considerato un piccolo sistema esperto in grado di impostare una strategia favorevole per il bianco. Sulla base del set di informazioni di cui dispone, il giocatore artificiale è in grado di scegliere una mossa razionale coerente con l'obiettivo di far si che il pedone raggiunga l'ottava traversa senza essere catturato.
<u>L'arbitro</u> è un modulo del programma che, come dalle regole del kriegspiel, ha il solo scopo di verificare la legalità delle mosse giocate. Quando una

---

48 Per l'implementazione è stato utilizzato il compilatore LPA PROLOG 2.5 (Logic Programming Associatel Ltd. 1988)

mossa è illegale l'arbitro avverte il giocatore di turno, chiedendo di giocarne un'altra ; inoltre avverte se la mossa legale ha provocato uno scacco.

Nella fig. 2.3 viene rappresentato il diagramma di flusso che descrive lo svolgimento generale del programma.

<u>L'utente</u> avvia il programma inserendo i dati che corrispondono al set informativo iniziale, le informazioni necessarie al programma per svolgere le sue funzioni sono contenute nella seguente struttura:

Chimuove .. RBx:RBy .. PBx:PBy .. [Onda] .. [Tabù] .. RNx:RNy

Chimuove indica il colore del giocatore che deve muovere , la variabile può assumere due valori : *b* oppure *n* che significano rispettivamente "deve muovere il bianco" o "deve muovere il nero".

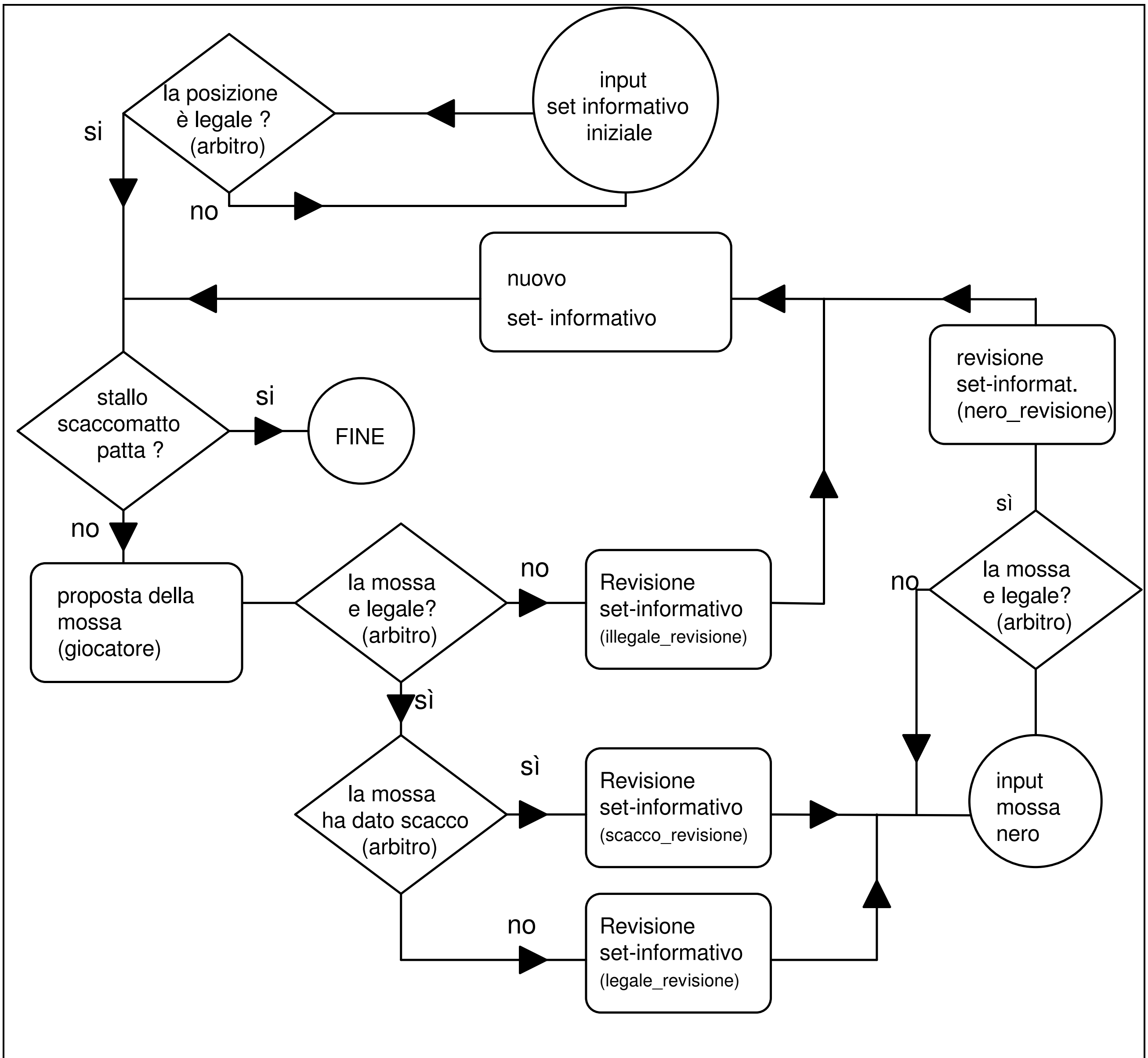


Fig. 2.3

RBx : RBy sono due interi collegati dal funtore ":" , che indicano le coordinate della casella in cui è situato il re bianco.

PBx : PBy sono le coordinate della casella in cui si trova il pedone bianco.

Onda è una lista che contiene le coordinate delle caselle in cui potrebbe essere situato il re avversario. La lista viene consultata dal giocatore artificiale, e fornisce a questo un insieme di caselle nel quale il re nemico è sicuramente collocato.

Tabù è un ulteriore lista che contiene le coordinate delle caselle in cui i pezzi bianchi non possono andare. Nel momento in cui il giocatore tenta di giocare una mossa e l'arbitro risponde che essa è illegale, alla lista Tabù vengono aggiunte le coordinate della casella in cui il nostro pezzo non è potuto andare.

RNx : RNy sono le coordinate della casella in cui si trova il re nero.

Sia il giocatore artificiale che l'arbitro, prendono in considerazione solo un sottinsieme delle informazioni contenute nella struttura descritta sopra.
In particolare al giocatore artificiale è inibito l'accesso all'informazione contenuta nella variabile *RNx : RNy* , in quanto se così non fosse, ci troveremo di fronte ad un giocatore ad informazione perfetta.
L'arbitro, invece, considera solo le seguenti variabili : Chimuove, RB, PB, RN, che sono necessarie e sufficienti a stabilire la legalità di una posizione.
Una volta stabilito che la posizione inserita dall'utente è legale, il giocatore artificiale, dopo aver valutato il set di informazioni a lui disponibile, propone la mossa. La routine del programma che esegue questo compito è sicuramente la parte più interessante, e verrà descritta dettagliatamente in seguito.
La proposta di mossa perviene all'arbitro che dichiara se è legale, illegale oppure legale con scacco. In tutti i casi, la dichiarazione dell'arbitro porterà ad una revisione del set informativo del giocatore artificiale, con la

differenza che se la mossa è illegale, il giocatore artificiale avrà diritto di tentare una nuova mossa.

Nel caso di mossa legale con o senza scacco il programma richiede che venga inserita la mossa del nero da parte dell'utente.

Il programma termina la propria esecuzione nel momento in cui il giocatore artificiale riesce a portare a promozione il proprio pedone, oppure quando il re nero cattura il pedone o ancora per stallo.

3.1 L' arbitro.

La routine che costituisce l'arbitro ha due compiti :

- Verificare la legalità di una posizione o di una mossa che viene proposta dal giocatore artificiale o dal giocatore umano, informando della presenza di un eventuale scacco
- Stabilire se la partita è finita con la vittoria del bianco o in parità .

La fig 3.3 descrive sinteticamente lo svolgimento della routine che verifica la legalità di una posizione.

Il test *condizioni di legalità* assume una diversa configurazione a seconda che sia stata inserita in input una mossa oppure una posizione, o che debba muovere il bianco oppure il nero. Ecco ad esempio le condizioni di legalità nel caso sia inserita in input una mossa dal nero. Ricordiamo che una mossa viene codificata indicando la casella di partenza e la casella di arrivo del pezzo mosso.

if casella di partenza *occupata* dal re nero **and**
casella di arrivo *adiacente* a casella di partenza **and**
casella di arrivo **not** *adiacente* a re bianco **and**
casella di arrivo **not** *controllata* dal pedone

then mossa legale
else mossa illegale.

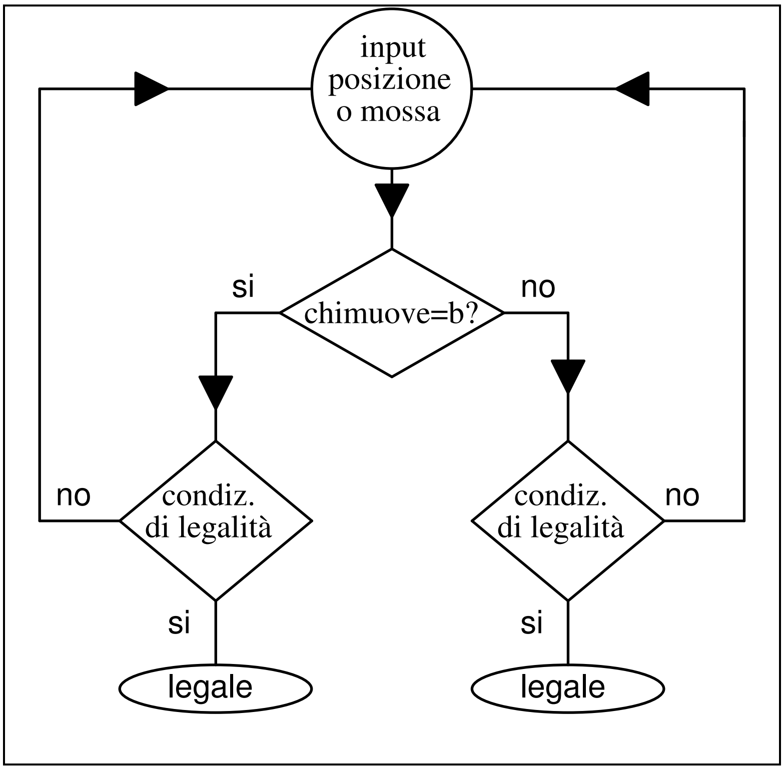


Fig 3.3

La fig. 4.3 descrive invece la procedura denominata *sentenza*.

Attraverso questa routine, l'arbitro è in grado di stabilire se la partita è terminata con il conseguimento dell'obiettivo da parte di uno dei giocatori, ricordiamo che per il bianco l'obiettivo è quello di portare il pedone in ottava, mentre per il nero è quello di catturare il pedone o di finire in una posizione di stallo.

Se la *sentenza* da esito affermativo, il programma termina la sua esecuzione comunicando il risultato della partita.

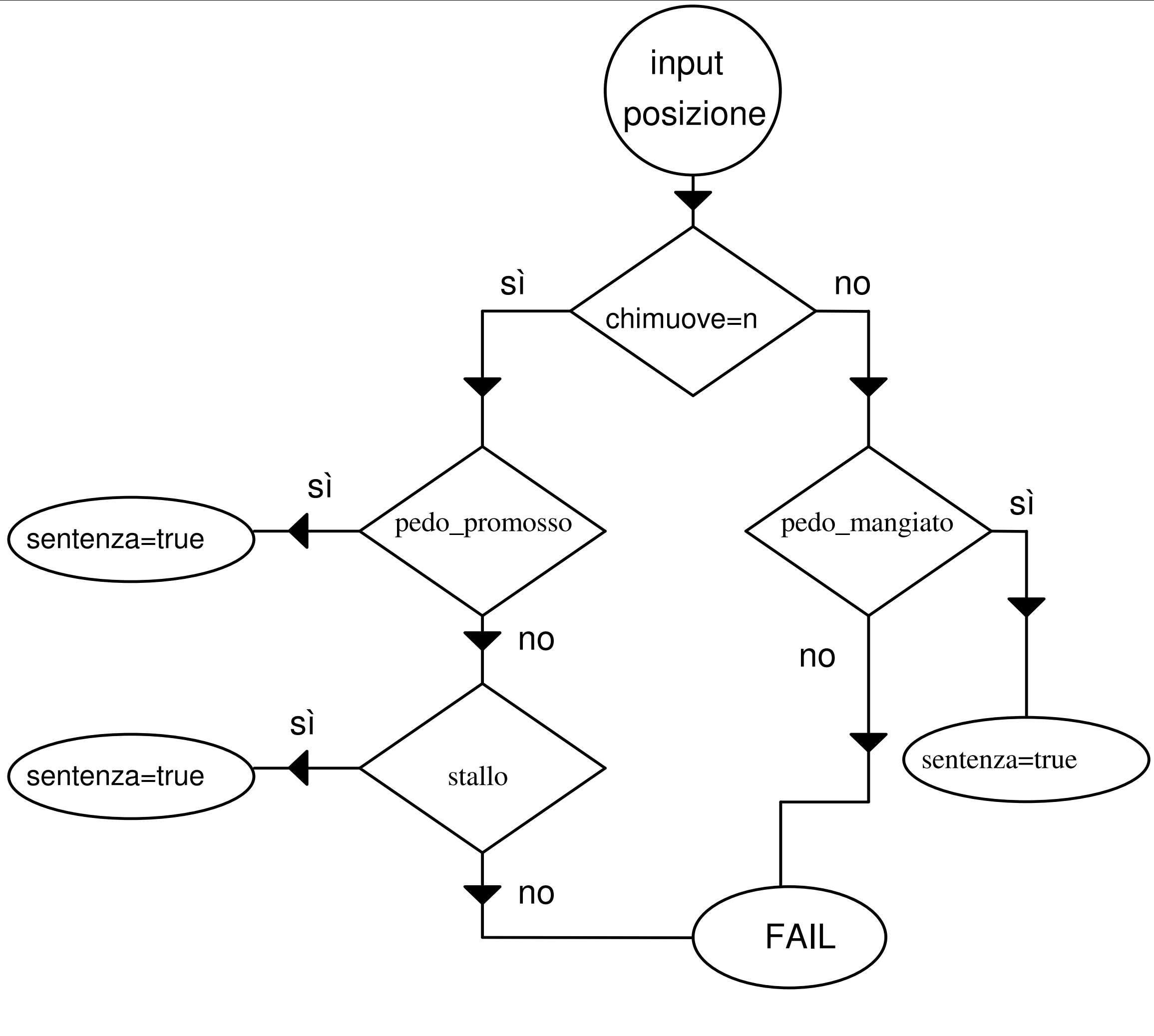


Fig 4.3

3.2 Il giocatore artificiale.

La routine che suggerisce di volta in volta le mosse da giocare rappresenta sicuramente il nocciolo del programma o se vogliamo dirlo in termini suggestivi, la parte "pensante" di esso.

Volendo effettuare un confronto con i programmi affini che giocano a scacchi, l'approccio utilizzato per affrontare questi problemi si avvicina di più a quello "Knowledge based", piuttosto che a quello "brute force" che come abbiamo visto nel capitolo 1 è stato un paradigma dominante nella ricerca degli ultimi anni.

Non risulta che fino ad oggi siano stati costruiti programmi in grado di giocare a kriegspiel, il territorio in cui si procede appare dunque inesplorato.

In ogni caso appare chiaramente improduttiva un'espansione dell'albero di gioco ad una prefissata profondità, poichè gli sforzi computazionali compiuti, sarebbero resi vani dalla componente incertezza che è presente in modo notevole.

Nella formulazione della strategia, il giocatore artificiale, non cerca di sviluppare alberi di gioco, ma sulle basi della propria conoscenza, cerca di far corrispondere le caratteristiche della posizione in esame con quelle di situazioni a lui note.

Ad ogni situazione è associata una particolare mossa oppure una lista di mosse in ordine di preferibilità, e il giocatore la propone all' arbitro il quale pronuncerà la sua sentenza di legalità o di illegalità. La fig 5.3 descrive sotto forma di diagramma di flusso la routine di formulazione della strategia.

L'implementazione risulta piuttosto agevole in linguaggio PROLOG, poichè la scelta di una mossa dipende dalla sua attitudine a soddisfare una sequenza di *goals*.

Il set informativo rilevante per il giocatore è costituito da 4 variabili : RB PB [Onda] e [Tabù]. Basandosi su queste informazioni il programma procede alla esecuzione dei test che permettono classificare la posizione in esame tra quelle incluse nella base di conoscenza del sistema.

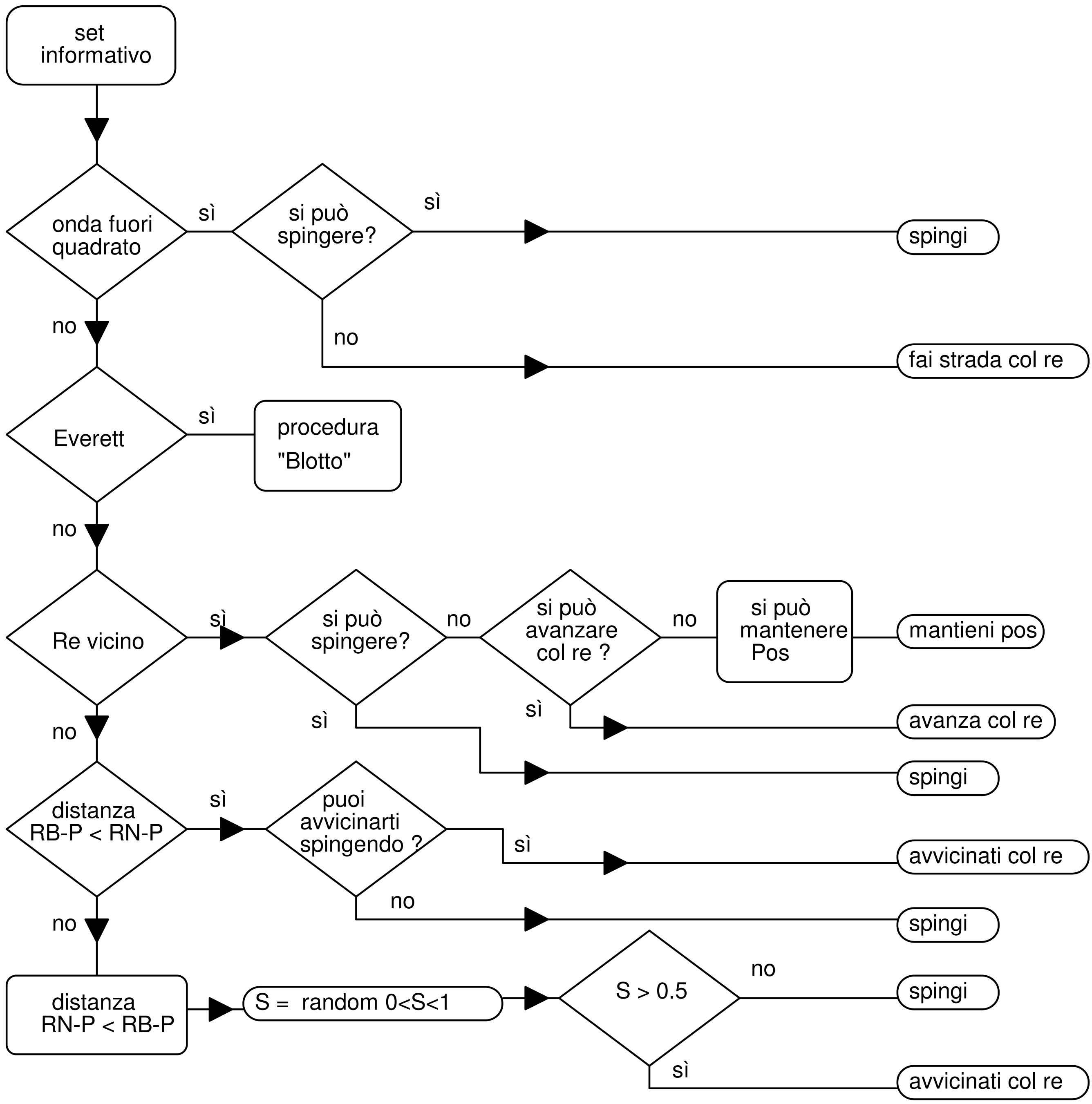

set informativo
onda fuori quadrato
sì
si può spingere?
sì
spingi
no
no
fai strada col re
Everett
sì
procedura "Blotto"
no
Re vicino
sì
si può spingere?
no
si può avanzare col re ?
no
si può mantenere Pos
mantieni pos
sì
sì
avanza col re
spingi
no
distanza RB-P < RN-P
sì
puoi avvicinarti spingendo ?
sì
avvicinati col re
no
spingi
no
distanza RN-P < RB-P
S = random 0<S<1
S > 0.5
no
spingi
sì
avvicinati col re

fig 5.3

I tipi di posizione riconosciuti dal sistema sono 5 e coprono esaustivamente ogni possibile situazione che può verificarsi sulla scacchiera; esaminiamoli uno alla volta :

<u>Onda fuori quadrato:</u>

E' il primo test che viene eseguito dal programma e verifica se la posizione esaminata è tale da permettere al pedone di andare a promozione senza essere ostacolato dal re nemico. Il test è basato sulla nota regola scacchistica del quadrato (Fig 6.3), che nel contesto viene estesa con alcune modifiche al kriegspiel.

	°				°			8
	°				°			7
	°	°	P	°	°			6
								5
								4
		R						3
								2
								1
a	b	c	d	e	f	g	h	

Il simbolo ° traccia i confini del *quadrato*: se il re nemico o l'insieme di case in cui può essere (Onda) sono fuori dal *quadrato*, il pedone può proseguire senza essere disturbato la "rotta" verso l'ottava traversa.

fig 6.3

Il test onda fuori quadrato, verifica dunque che ogni casella appartenente all'insieme [Onda], sia all esterno del quadrato.

<u>Everett:</u>

Il test *Everett* viene eseguito se il test *Onda fuori quadrato* da esito negativo. Una posizione che soddisfa il test *Everett*, è analoga a quella risolta come gioco ricorsivo nel capitolo precedente (Fig 7.3).

		X	X	X				8
		X			X	X		7
		P						6
			R					5
								4
								3
								2
								1
a	b	c	d	e	f	g	h	

La posizione soddisfa il test Everett in quanto
1. Il pedone bianco è collocato sulla sesta traversa
2. Il re bianco è adiacente e dietro al pedone
3. Una parte di *Onda* è inclusa nel *quadrato*

fig 7.3

La soluzione ottenuta con l'ausilio della teoria dei giochi[49], ci suggerisce la strategia migliore che conduce alla vittoria. Questo *prezioso consiglio* viene inserito nella base di conoscenze del giocatore artificiale, e sarà utilizzato quando le condizioni della posizione lo permetteranno[50].

E' importante, a mio avviso, sottolineare il modo in cui dei risultati ottenuti con razionalità sostanziale vengano utilizzati da un agente proceduralmente razionale (il giocatore artificiale), al fine di rendere più efficiente il proprio operato.

Se il test *Everett* ha successo viene chiamata la procedura *Blotto*, che suggerisce la strategia ottimale individuata nello studio del capitolo precedente.

Il grafo di fig 8.3 indica, con riferimento alla posizione di fig 7.3, quale sia la strategia *sostanzialmente razionale* da seguire. Ricordiamo che l'arco indica la mossa da giocare al primo tentativo, l'arco tratteggiato indica la mossa da giocare se il primo tentativo è stato dichiarato illegale dall'arbitro.

---

[49] Vedi cap. 2 par..5

[50] Nel caso in questione, quando il test *Everett* ha successo.

Se dallo stesso nodo partono due archi della stessa categoria, ad essi viene associata una distribuzione di probabilità.

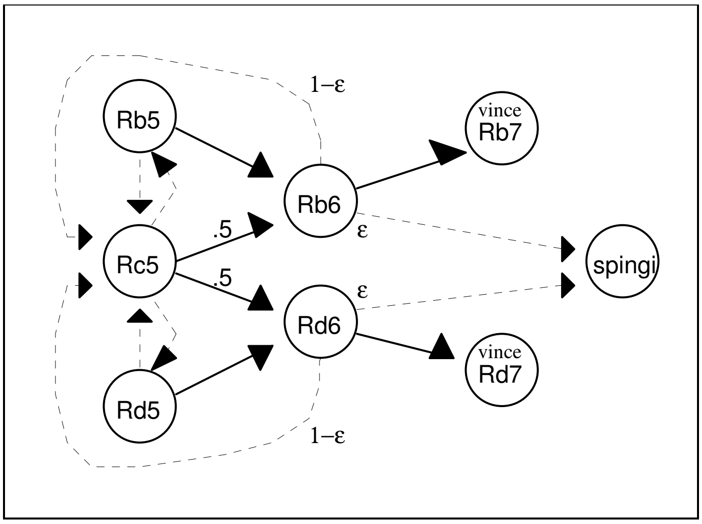


Fig 8.3

La *procedura Blotto* viene generalizzata nel diagramma di flusso di fig 9.3, in modo da essere adattata a qualunque posizione che soddisfi il test *Everett*.

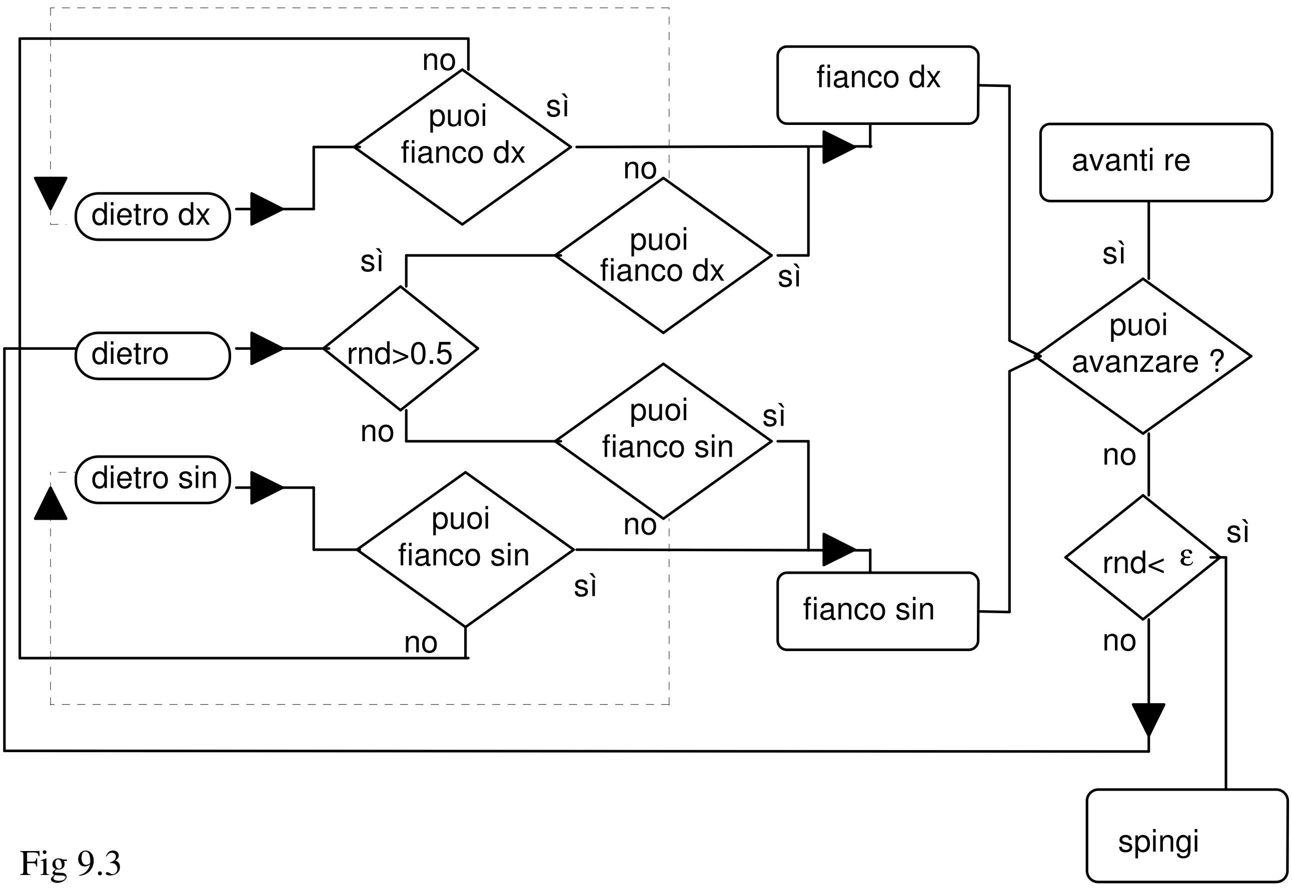


Fig 9.3

Le situazioni *fianco dx*, *fianco sin* ecc. esprimono la collocazione del re bianco rispetto al pedone bianco.(Fig 10.3).

			8
avanti re		avanti re	7
finco sin	P	fianco dx	6
dietro sin	dietro	dietro dx	5

fig 10.3

Ad esempio il test del tipo *Puoi fianco dx* verifica se è possibile raggiungere con il re la casa collocata a destra del pedone.

Re vicino:

Se falliscono i primi due test, *re_vicino* verifica se il re nero si trova in una casella adiacente al pedone bianco, in caso affermativo, il programma cerca di eseguire uno dei tre seguenti obiettivi : *Spingi*, *Avanza_col_re*, *Mantieni_Pos*.

Gli obiettivi sono posti gerarchicamente, nel senso che prima si cercherà di soddisfare *Spingi*, se questo non sarà possibile si procederà con *Avanza_col_re*, e se nemmeno questo fosse possibile si eseguirà *Mantieni_Pos* che è sempre possibile. Per stabilire se gli obiettivi sono possibili o no il programma ricorre al proprio set di informazioni, ed in particolare alla lista *Tabù*, che si modifica nel corso del gioco e indica le case che non possono essere occupate dai pezzi bianchi in seguito a informazioni dell'arbitro.

Oltre al vincolo di non potere occupare una casella *Tabù* , la mossa che verrà decisa dovrà anche rispettare l'obiettivo minimo di non lasciare sguarnito il pedone.

Pedo_difendibile:

Il test in questione verifica se nella posizione esaminata la distanza tra il nostro re e il pedone risulta essere minore alla distanza tra il pedone e il re avversario.

Poichè non sappiamo dove sia il re avversario, ma disponiamo solo di una lista di caselle in cui è potenzialmente collocato ([*Onda*]), l'unico modo per garantire con probabilità 1 che il pedone sia difendibile è che

$$\mathrm{Dist}(Rb, Pb) < \min_{X \in Onda} \mathrm{Dist}(X, Pb)$$

L'obiettivo immediato del programma è di avvicinarsi con il re al pedone in modo da portarsi nella condizione *re_vicino*.

Not Pedo_difendibile:

Se anche il test *Pedo_difendibile* fallisce, la posizione in esame rientra sicuramente in questa categoria di posizioni: il test in questione, quindi, è sempre positivo.

Mentre in tutte le altre posizioni il programma garantisce la vittoria con probabilità che tende a 1, in questa situazione è possibile che il re nero catturi il pedone e il gioco termini con risultato di parità. (Fig 11.3).

	X	X	X					8
				X	X			7
								6
				P				5
				X	X			4
	R							3
								2
								1
a	b	c	d	e	f	g	h	

Se il re nero fosse in e7 o f7, e conoscesse la collocazione del pedone, il bianco non potrebbe ostacolare in alcun modo la cattura del pedone da parte del re nero.
Il re nero però potrebbe anche essere nelle case b8,c8 o d8 in tal caso il pedone sarebbe difendibile dal re bianco.
Nell' ipotesi invece che il re nero sia in e4 o f4, l'unica mossa che garantirebbe la salvezza del pedone sarebbe la spinta.

fig 11.3

Come viene illustrato nella fig 11.3, la situazione in cui il giocatore artificiale deve decidere la mossa da eseguire è di completa incertezza.

Se ci fossero delle ragioni per sostenere che il re sia collocato con maggiore probabilità nelle case e4 o f4, sarebbe opportuno giocare la strategia che prevede la spinta di pedone.

Poichè non esiste alcuna ipotesi probabilisica sulla collocazione del re nemico, il giocatore artificiale sceglie con p = 0.5 la strategia di spinta del pedone oppure la strategia di avvicinamento col re.

3.3 La revisione del set informativo.

La revisione del set informativo è un' operazione che virtualmente viene eseguita dal giocatore artificiale, che sulla base delle risposte dell'arbitro cerca di inferire la reale posizione sulla scacchiera.
Abbiamo quattro diversi tipi di procedura che vengono utilizzati in altrettante fasi del programma.

- Se l'arbitro dichiara che la mossa giocata dal bianco è legale viene chiamata la procedura legale_revisione
- Se l'arbitro dichiara che la mossa è legale con scacco, viene chiamata la procedura scacco_revisione.
- Se l'arbitro dichiara che la mossa giocata dal bianco è illegale viene chiamata la procedura illegale_revisione
- Dopo la mossa del nero viene chiamata la procedura nero_revisione.

Legale_revisione :

La procedura modifica alcune variabili della struttura informativa principale del programma. Poichè la procedura viene chiamata dopo una mossa legale del bianco, per prima cosa verranno modificate le variabili *RBx : RBy* oppure *PBx : PBy* se il pezzo mosso è il re o il pedone.

L'aggiornamento della collocazione dei pezzi bianchi è rilevante sia per il set informativo del giocatore artificiale che per quello dell'arbitro.

Legale_revisione modificherà anche la lista *[Onda]*, che come abbiamo visto è rilevante solo per il set informativo del giocatore artificiale. In particolare dalla lista vengono tolte tutte le possibili collocazioni del re avversario che sono incompatibili con la mossa appena eseguita.

Ad esempio nella fig 12.3.a individuiamo il set informativo del bianco prima della mossa Re6. La lista *Onda* è costituita dalle caselle [d8,e8,f8,e7,f7,d6,d5].

			X	X	X			8
				X	X			7
		P	X					6
			X		R			5
								4
								3
								2
								1
a	b	c	d	e	f	g	h	

fig 12.3.a

			X	X	X			8
								7
		P		R				6
								5
								4
								3
								2
								1
a	b	c	d	e	f	g	h	

fig 12.3.b

Se la mossa Re6 viene dichiarata legale ci troveremmo nel set informativo di fig. 12.3.b. L' insieme di case [e7,f7,d6,d5] è incompatibile con la presenza del re in e6[51], quindi viene tolto dall' *Onda* che rimarrà composta da [d8,e8,f8].

[51] L'incompatibilità è dovuta al regolamento del gioco che impedisce che due re si trovino in caselle adiacenti.

<u>scacco_revisione :</u>

E' simile alla procedura descritta sopra con la differenza che la revisione della lista *Onda* è molto più efficiente. Se, infatti, dopo la mossa escogitata dal giocatore artificiale, l'arbitro dichiara "mossa legale con scacco", è immediato constatare che il re nero è sicuramente collocato in una casella sotto la minaccia del pedone. (Figg. 13.3.a, 13.3.b)

X	X	X	X	X	X	X	X	8
X	X	X	X	X	X	X	X	7
X	X					X	X	6
			P	R				5
								4
								3
								2
								1
a	b	c	d	e	f	g	h	

fig 13.3.a

								8
		X		X				7
			P					6
				R				5
								4
								3
								2
								1
a	b	c	d	e	f	g	h	

fig 13.3.b

Partendo dalla fig 13.3.a se dopo la mossa Pd5-d6, l'arbitro dichiarasse "legale con scacco" la lista *Onda* si ridimensionerebbe alle sole due caselle [c7,d7].

<u>illegale_revisione :</u>

L'aggiornamento della struttura informativa a seguito di una mossa illegale non modifica la collocazione dei pezzi, ma solamente le liste *Onda* e *Tabù*, che sono rilevanti solo per il set informativo del giocatore artificiale .

Supponiamo di trovarci nella fig. 14.3 e di aver tentato la mossa Re6 che è stata dichiarata illegale dall' arbitro.

La casella e6 è dunque inaccessibile al re bianco, e verrà inserita nella lista tabù.

Tabù è una lista che è vuota ogni volta che il giocatore inizia la sua serie di tentativi. A mano a mano che vengono tentate mosse illegali, la lista si riempie e arricchisce il set informativo del giocatore artificiale.

	X	X	X		X	X	X	8
			X			X	X	7
			P			X	X	6
				R				5
								4
								3
								2
								1
a	b	c	d	e	f	g	h	

fig 14.3.a

Oltre alla revisione della lista tabù, la procedura illegale_revisione aggiorna anche la lista onda, sulle basi delle implicazioni logiche che derivano dalla scoperta di una casella tabù.

Rimanendo all' esempio di fig 14.3.a, il fatto che la casa e6 sia tabù implica che il re avversario sia collocato in adiacenza ad e6.

Poichè d7 è l'unica casa dell'onda adiacente ad e6, posso stabilire con certezza che il re nemico è collocato in d7, la nuova lista Onda sarà allora composta da un solo elemento [d7]. (fig 14.3.b)

								8
			X					7
			P					6
				R				5
								4
								3
								2
								1
a	b	c	d	e	f	g	h	

fig 14.3.b

<u>nero_revisione :</u>

La procedura effettua l'aggiornamento di tre variabili.

<u>*RNx : RNy*</u> che è la collocazione del re nero, che è nota solo all'arbitro.

La lista <u>*Tabù*</u> viene azzerata, in quanto essendosi mosso il re nero non siamo più in grado di stabilire con certezza se le case inibite al movimento dei nostri pezzi siano rimaste tali.

La parte più rilevante di questa procedura è la revisione della lista <u>*Onda*</u>, che contiene le caselle in cui il re nemico è potenzialmente collocato.

La lista Onda tende a crescere, poichè il giocatore artificiale non sa dove il re nero si sia mosso e dunque il numero delle case in cui può collocarsi cresce progressivamente. In fig 15.3.a e 15.3.b osserviamo due set informativi, in cui si nota la differenza nelle dimensioni del set Onda, prima e dopo la mossa del nero.

								8
	X					X		7
			P					6
					R			5
								4
								3
								2
								1
a	b	c	d	e	f	g	h	

fig 15.3.a

X	X	X			X	X	X	8
X					X		X	7
X	X	X	P				X	6
					R			5
								4
								3
								2
								1
a	b	c	d	e	f	g	h	

fig 15.3.b

4. Listato del programma :

```
/*    Questo programma è in grado di giocare un finale di kriegspiel P+R vs R   utilizzando una
base di conoscenza .
Autore : Fabio Maran 1993
Linguaggio : Prolog.  Compilatore LPA PROLOG Ver 2.5
*/



/* operatori predefiniti   */

:- op(220, xfy, ..).
:- op(200, xfy, :).
:- op(140, fx, not).


/*   -----------------------------------------------------------------------------------------------------------------   */
/*                                          programma principale                                                         */
/*   -----------------------------------------------------------------------------------------------------------------   */


start :-
   grafici,
   inserisci_pos(Pos),
   playgame(Pos),
   chiudi.


playgame(Pos) :-
      chimuove(Pos,b),
      sentenza(Pos),!.

playgame(Pos) :-
      chimuove(Pos, b),
      show(Pos),nl,
      write('lasciami pensare ...'),nl,
      strategia(Pos, Mossa),
      write('Provo la seguente mossa: '),
      scrivi(Mossa),nl,
      arbitro(Pos, Mossa, Informa),
      revisione(Informa, Pos, Mossa, Pos1),
      get0('TRM:',X), nl,!,
      playgame(Pos1).


playgame(Pos) :-
      chimuove(Pos,n),
      sentenza(Pos),!.
```

```
playgame(Pos) :-
       chimuove(Pos, n),
       show(Pos),
       nl,write('inserisci la mossa del nero'),nl,
       mossa_nero(Pos, Pos1),
       write('Premi un tasto per vedere il mio set-informativo'),
       write(' dopo nero_revisione'),
       get0('TRM:',X),
       nero_revisione(Pos1, Pos2),!,
       playgame(Pos2).

mossa_nero(Pos1, n..RB..PB..Onda..[]..Arriv) :-
       rb(Pos1, RB),
       pb(Pos1, PB),
       onda(Pos1, Onda),
       leggimossa(P-Arr),
       (legale(Pos1,P-Arr), Arriv = Arr,! ;
        nl, write('Mossa illegale'),nl,
        mossa_nero(Pos1, n..RB..PB..Onda..[]..Arriv)).

revisione(scacco,Pos,Mossa,Pos1) :-
  write('Arbitro dice che la mossa è legale con scacco'),nl,
  scacco_revisione(Pos, Mossa, Pos1),
  write('Premi un tasto per vedere il mio set-informativo'),
  write(' dopo scacco_revisione').


revisione(no_scacco,Pos, Mossa, Pos1) :-
  write('Arbitro dice che la mossa è legale '),nl,
  legale_revisione(Pos, Mossa, Pos1),
  write('Premi un tasto per vedere il mio set-informativo'),
  write(' dopo legale_revisione').


revisione(illegale, Pos, Mossa, Pos1) :-
  write('Arbitro dice che la mossa è illegale'),nl,
  illegale_revisione(Pos, Mossa, Pos1),
  write('Premi un tasto per vedere il  mio set'),
  write('-informativo dopo illegale_revisione').


/*  -----------------------------------------------------------------------------------------------------------------  */
/*              Test che propongono le mosse da giocare sulla base dei dati contenuti              */
/*                                              nel set-informativo                                                   */
/*  -----------------------------------------------------------------------------------------------------------------  */


strategia(Pos, Mossa) :-
       not onda_nel_quadrato(Pos),
       (spingi(Pos, Mossa);
       fai_strada(Pos, Mossa)).

strategia(Pos, Mossa) :-
```

```
      everett(Pos),
      blotto(Pos, Mossa).

strategia(Pos, Mossa) :-
      re_vicino(Pos),
      (spingi(Pos, Mossa),spia(Pos, Mossa, Pos1), re_vicino(Pos1) ;
      avanti_re(Pos, Mossa),spia(Pos, Mossa, Pos1), re_vicino(Pos1) ;
      mantieni_re(Pos, Mossa),spia(Pos, Mossa, Pos1), re_vicino(Pos1)).

strategia(Pos, Mossa) :-
      pedo_difendibile(Pos),
      (spingi(Pos, Mossa),
      spia(Pos, Mossa, Pos1),
      new_dist_smaller(Pos1,Pos);
      avvicinati(Pos, Mossa)).

strategia(Pos, Mossa) :-
      not pedo_difendibile(Pos),
      S is irand(100),
      ((S =< 50), spingi(Pos, Mossa) ;
      (S  > 50), avvicinati(Pos, Mossa)).




/* La posizione è rappresentata da:

   Chimuove .. RBx : RBy .. PBx : PBy .. Onda .. RNx : RNy

   Chimuove è colui che deve muovere che può essere 'b' o 'n'
   RBx RBy  sono le coordinate del re bianco
   PBx PBy  sono le coordinate del pedone bianco
   Onda   è una lista che contiene le coordinate delle caselle in cui
         il re nero potrebbe essere presente.
   Tabù   é una lista di caselle in cui il nostro re, sicuramente non
         può andare, in quanto, in seguito ad un informazione dell'arbitro
         abbiamo dedotto concertezza che la casa è controllata dal re nemico
   RNx RNy  sono le coordinate del re nero, che non sono note al sistema
           esperto, ma sono note all'arbitro, il programma infatti funge
           anche da arbitro                                                  */


/*  Richiama le informazioni volute su una posizione                          */

chimuove( Chimuove.._, Chimuove).          /* il colore che deve muovere         */
rb( _..RB.._, RB).                         /* coordinate del re bianco           */
pb( _.._..PB.._, PB).                      /* coordinate del pedone bianco       */
onda(_.._.._..Onda.._,Onda).               /* possibili coordinate del re nero   */
tabù(_.._.._.._..Tabù.._,Tabù).            /* case in cui sicuramente non si     */
                                           /* può andare                         */
rn( _.._.._.._.._..RN, RN).                /* coordinate re nero                 */
```

```
col(_:Col,Col).                                   /* ordinata di una casella          */


coord(1). coord(2). coord(3). coord(4).           /* database di coordinate      */
coord(5). coord(6). coord(7). coord(8).


eve(3:3). eve(4:3). eve(5:3).          /* coordinate in cui vale  la condizione everett   */
eve(6:3). eve(7:3).


/*  ---------------------------------------------------------------------------------------------------------------  */
/*                                  Procedure sui movimenti dei pezzi                                                 */
/*  ---------------------------------------------------------------------------------------------------------------  */

spingi(Pos, PBx:PBy-PBx1:PBy1) :-
  pb(Pos, PBx:PBy),
  rb(Pos, RB),
  tabù(Pos, Tabù),
  succ(PBx:PBy,PBx1:PBy1),
  PBx1:PBy1 \== RB,
  not membro(PBx1:PBy1,Tabù).



fai_strada(Pos, RBx:RBy-RBx1:RBy1) :-
   rb(Pos, RBx:RBy),
   tabù(Pos,Tabù),
   orizzadiac(RBx:RBy, RBx1:RBy1),
   not membro(RBx1:RBy1,Tabù).

avanti_re( Pos, RB-RB1) :-
   rb(Pos,RB),
   tabù(Pos, Tabù),
   pb(Pos, PB),
   adiac( RB, RB1),
   re_avanza(_..RB1.._,_..RB.._),
   RB1 \== PB,
   not membro(RB1,Tabù).

re_avanza(Pos,RootPos) :-
  rb(Pos, X:Y),
  rb(RootPos, X1:Y1),
  (Y < Y1).

mantieni_re(Pos,RB-RB1) :-
   rb(Pos, RB),
   tabù(Pos, Tabù),
   orizzadiac(RB, RB1),
   not membro(RB1,Tabù).

avvicinati(Pos, RBx:RBy-RB1x:RB1y) :-
   rb(Pos, RBx:RBy),
   pb(Pos, PB),
```

```
   tabù(Pos, Tabù),
   adiac(RBx:RBy, RB1x:RB1y),
   new_dist_smaller(_..RB1x:RB1y..PB.._,Pos),
   not membro(RB1x:RB1y, Tabù).
/* --------------------------------------------------------------------------------------------------------------- */
/*                                       Procedura blotto                                                        */
/* --------------------------------------------------------------------------------------------------------------- */


blotto(_..RB..PB.._..Tabù.._, RB-Arriv) :-
  dietro(PB, RB),
  fianco_sin(PB, FS),fianco_dx(PB, FD),
  not membro(FS, Tabù),not membro(FD, Tabù),
  I is irand(10),
  (I > 4, Arriv = FD ;
   Arriv = FS).

blotto(_..RB..PB.._..Tabù.._, RB-Arriv) :-
   dietro(PB, RB),
   I is irand(10),
   (I>4, dietro_dx(PB, Arriv) ;
    dietro_sin(PB, Arriv)).

blotto(_..RB..PB.._..Tabù.._, RB-Arriv) :-
   dietro_dx(PB, RB),
   fianco_dx(PB,YY),
   ((not membro(YY, Tabù), Arriv = YY) ;
   dietro(PB, Arriv)).


blotto(_..RB..PB.._..Tabù.._, RB-Arriv) :-
   dietro_sin(PB, RB),
   fianco_sin(PB,YY),
   ((not membro(YY, Tabù), Arriv = YY) ;
   dietro(PB, Arriv)).

blotto(_..RB..PB.._..Tabù.._, Part-Arriv) :-
   (fianco_dx(PB, RB) ; fianco_sin(PB, RB)),
   succ(RB, XX),
   ( not membro(XX, Tabù), Part = RB, Arriv = XX ;
     I is irand(9), I < 1, Part = PB, succ(PB, Arriv) ;
     Part = RB, dietro(PB, Arriv) ).


/* --------------------------------------------------------------------------------------------------------------- */
/*                                       Libreria dei predicati                                                  */
/* --------------------------------------------------------------------------------------------------------------- */

spia(b..Part..PB..Onda..Tabù..RN,Part-Arriv,b..Arriv..PB..Onda..Tabù..RN).

spia(b..RB..Part..Onda..Tabù..RN,Part-Arriv,b..RB..Arriv..Onda..Tabù..RN).

pedo_promosso(Pos) :-
   pb(Pos, PBx:PBy),
```

```
   rb(Pos, RB),
   rn(Pos, RN),
   PBy = 1,
   (not adiac(RN, PBx:PBy);
    adiac(RB,PBx:PBy)).
stallo(Pos) :-
   not re_mobile(Pos).

scacco(Pos) :-
   pb(Pos, PB),
   rn(Pos, RN),
   sottopedone(PB, RN).

re_mobile(Pos) :-
   pb(Pos, PB),
   rb(Pos, RB),
   rn(Pos, RN),
   adiac(RN, X),
   not adiac(RB, X),
   not sottopedone(PB, X).


pedo_mangiato(Pos) :-
   pb(Pos, PB),
   rn(Pos, RN),
   PB == RN.

onda_nel_quadrato(Pos) :-
   pb(Pos, PB),
   onda(Pos, Onda),
   member(X, Onda),
   casa_nel_quadrato(PB, X).

casa_nel_quadrato(PBx:PBy, Xx:Xy) :-
   Xy =< PBy  ,
   dist(PBx:PBy, Xx:Xy, D),
   D =< (PBy-1).

re_vicino(_..RB..PB.._.._.._) :-
   adiac(RB, PB).

pedo_difendibile(_..RB..PB..Onda.._..RN) :-
   dist( RB, PB, D),
   member(X, Onda),
   dist( PB, X , D1),
   D =< D1 + 1.

new_dist_smaller(_..RB1..PB1.._,Pos) :-
   rb(Pos,RB),
   pb(Pos,PB),
   dist(PB, RB, Olddist),
   dist(PB1, RB1, Newdist),
   Newdist < Olddist.
```

```
sottopedone(PBx:PBy, PB1x:PB1y) :-
   PB1y is PBy-1 ,
   (PB1x is PBx-1 ; PB1x is PBx+1).



everett(Pos) :-
   pb(Pos, PB),
   rb(Pos, RB),
   eve(X),
   PB == X,!,
   re_vicino(Pos),
   not sottopedone(PB, RB),
   not succ(PB,RB).



/*  ------------------------------------------------------------------------------------------------  */
/*                              Procedure dell' arbitro                                               */
/*  ------------------------------------------------------------------------------------------------  */

arbitro(Pos, Mossa, Informa) :-
    legale(Pos, Mossa),
    spia(Pos, Mossa, Pos1),
    (scacco(Pos1),
    Informa = scacco;
    Informa = no_scacco).

arbitro(Pos, Mossa, Informa) :-
       Informa = illegale.

legale(b..RB.._.._.._..RN, Part-Arriv) :-
  Part == RB,!,
  not adiac(Arriv, RN).

legale(b.._..PB.._.._..RN, Part-Arriv) :-
  Arriv \== RN.

legale(n..RB..PB.._.._..RN, Part-Arriv) :-
  Part == RN,
adiac(Part,Arriv),
not adiac(Arriv, RB),
not sottopedone(PB, Arriv).


casella_legale(Cx : Cy) :-
    in(Cx),
    in(Cy).




sentenza(Pos) :-
     chimuove(Pos,n),
```

```
    pedo_promosso(Pos),nl,
    write('il pedone è promosso: la partita è vinta'),nl.

sentenza(Pos) :-
    chimuove(Pos,n),
    stallo(Pos),nl,
    write('il re nero è in stallo: la partita è patta'),nl.
sentenza(Pos) :-
    chimuove(Pos,b),
    pedo_mangiato(Pos),nl,
    write('Il nero è riuscito a catturate il pedone: la partita è patta'),nl.



/* ------------------------------------------------------------------------------------------------------------------- */
/*                  Revisione dei set-informativi sulla base delle informazioni dell'arbitro                            */
/* ------------------------------------------------------------------------------------------------------------------- */

legale_revisione(Pos, Part-Arriv, n..Arriv..PB..Onda1..[]..RN) :-
 rb(Pos, RB),
 pb(Pos, PB),
 rn(Pos, RN),
 onda(Pos, Onda),
 Part == RB,
 setof(X, adiac(Arriv, X), Set_adiac),
 togli(Set_adiac, Onda, Onda1),!.

legale_revisione(Pos, Part-Arriv, n..RB..Arriv..Onda1..[]..RN) :-
 rb(Pos, RB),
 rn(Pos, RN),
 onda(Pos, Onda),
 setof(X, sottopedone(Arriv, X), Set_sottopedone),
 XX = [Arriv|Set_sottopedone],
 togli(XX, Onda, Onda1),!.



scacco_revisione(Pos, Part-Arriv, n..RB..Arriv..Onda1..[]..RN) :-
 rb(Pos, RB),
 rn(Pos, RN),
 onda(Pos, Onda),
 setof(X, sottopedone(Arriv, X), Set_sottopedone),
 intersezione(Onda, Set_sottopedone, Onda1).


nero_revisione(n..RB..PB..Onda..Tabù..RN,b..RB..PB..New_onda..[]..RN) :-
    revisione(Onda, Onda1),
    setof(X, adiac(RB, X), Set_adiac),
    togli(Set_adiac, Onda1, Onda2),
    setof(X, sottopedone(PB, X), Set_sottopedone),
    togli(Set_sottopedone, Onda2, New_onda).
```

```
revisione([], []).                              /* dato un insieme di case in cui        */
                                                /* poteva collocarsi il re nero          */
revisione([X|Rest], New_onda) :-                /* restituisce il nuovo insieme          */
 revisione(Rest, Onda1),                        /* in cui si può collocare               */
 setof(X1,adiac(X,X1), Set),
 unione(Set, Onda1, New_onda).


illegale_revisione(b..RB..PB..Onda..Tabù..RN,Part-Arriv,b..RB..PB..Onda1..Tabù1..RN) :-
  Part == RB,
  aggiungi(Arriv,Tabù,Tabù1),
  revisione(Tabù1, Adiac_tabù),
  setof(X,adiac(RB,X),Adiac_rb),
  setof(X,sottopedone(PB,X),Sottopedone),
  togli(Adiac_rb,Adiac_tabù,Set),
  togli(Sottopedone,Set,Set1),
  intersezione(Onda,Set1,Onda1).

illegale_revisione(_..RB..PB.._..Tabù..RN,_-Arriv,b..RB..PB..Onda1..Tabù1..RN) :-
  Tabù1 = Arriv,
  Onda1 = Arriv.




/*  -----------------------------------------------------------------------------------------------------------------  */
/*                                          relazioni tra le caselle                                                   */
/*  -----------------------------------------------------------------------------------------------------------------  */


n(N,N1) :- ( N1 is N-1 ;              /* relazione tra il numero precedente e il  */
             N1 is N+1),              /* successivo ad N, purchè entro i limiti   */
             in(N1).                  /* della scacchiera                         */

  in(N)  :-                           /* Verifica se le coordinate sono           */
     N>0, N<9.                        /* dentro la scacchiera                     */


diagadiac( X : Y, X1 : Y1) :-         /* Caselle adiacenti in diagonale a X:Y     */
 n(Y, Y1),n(X, X1).

veradiac(X : Y, X : Y1) :-            /* Caselle diacenti in verticale a X:Y      */
  n(Y, Y1).

orizzadiac( X : Y, X1 : Y) :-         /* Caselle adiac. in orizzontale a X:Y      */
 n(X,X1).



adiac( S, S1) :-                      /* L'insieme di tutte le caselle che        */
  diagadiac( S, S1);                  /* confinano con S                          */
  orizzadiac( S, S1);                 /* In altri termini lo spazio controllato   */
  veradiac( S, S1).                   /* dal re                                   */
```

```
dietro(PBx:PBy, RBx:RBy) :-
   RBx = PBx,
   RBy is PBy+1.

dietro_dx(PBx:PBy, RBx:RBy) :-
   RBx is PBx+1,
   RBy is PBy+1.
dietro_sin(PBx:PBy, RBx:RBy) :-
   RBx is PBx-1,
   RBy is PBy+1.

fianco_dx(PBx:PBy, RBx:RBy) :-
    RBx is PBx+1,
    RBy = PBy.

fianco_sin(PBx:PBy, RBx:RBy) :-
    RBx is PBx-1,
    RBy = PBy.


dist( X : Y, X1 : Y1, D) :-         /* distanza tra due caselle  */
  absdiff( X, X1, Dx),
  absdiff( Y, Y1, Dy),
  max( Dx, Dy, D).

absdiff(A, B, D) :-
  A > B, !, D is A-B;
          D is B-A.

max(A, B, M) :-
  A >= B, !, M = A;
           M = B.


succ(PBx:PBy,PBx1:PBy1) :-     /* restituisce la casella successiva */
  PBx1 = PBx,
  PBy1 is (PBy - 1).



/*   ----------------------------------------------------------------------------------------------------------------------   */
/*                                               operazioni tra insiemi                                                       */
/*   ----------------------------------------------------------------------------------------------------------------------   */


togli( [], Onda, Onda) .                    /* togli(A,B,C) è la sottrazione tra */
                                            /* insiemi dove C = B - A             */
togli( [X|Rest], Onda, Ondan) :-
  canc( X, Onda, Onda1),
  togli( Rest, Onda1, Ondan).
```

```
togli( [X|Rest], Onda, Ondan) :-
  togli( Rest, Onda, Ondan).

  canc( X, [X|Tail], Tail) .
  canc( X, [Y|Tail], [Y|Tail1]) :-
          canc( X, Tail, Tail1).



unione([X|[]], S, S1) :-                 /* unione(A,B,C) è l'unione di insiemi */
   not member(X,S),                      /* C = A U B                           */
   aggiungi(X, S, S1);
   S1 = S .

unione([X|Rest], S, New_S) :-
   not member( X, S),
   aggiungi( X, S, S1),
   unione(Rest, S1, New_S).

unione( [X|Rest], S, New_S) :-
   unione( Rest, S, New_S).



intersezione([], S, []).                 /* intersezione(A,B,C) è l'intersezione  */
                                         /* di insiemi C = A int B                */
intersezione([X|Rest], S, I) :-
    intersezione(Rest, S, Ins),
    (membro(X, S), aggiungi(X, Ins, I);
    I = Ins).


aggiungi(X, L, [X|L]).                   /* aggiunge l'elemento X alla lista L */



member(X,[X|L]).

member(X,[Y|L]) :-
  member(X, L).


membro(X,[X|Rest]) :- !.

 membro(X,[Y|Rest]):-
```

```
   membro(X,Rest).
/*  -----------------------------------------------------------------------------------------------------------------  */
/*                                  Visualizzazione della posizione                                                      */
/*  -----------------------------------------------------------------------------------------------------------------  */

show(Pos) :-
cuwind(giocatore),
coord(Y),nl,
coord(X),write(' '),
stampa_g( X : Y, Pos),              /*  una ad una richiama tutte le 64 caselle */
fail.
show(Pos) :-
cuwind(set_arbitro),
coord(Y),nl,
coord(X),write(' '),
stampa_a( X : Y, Pos),              /*  una ad una richiama tutte le 64 caselle */
fail.

show(Pos) :- cuwind(bassa).

stampa_g( Casella, Pos) :-
  rb(Pos, Casella),!, write('R');      /* scrive R  dove c'è il  re bianco        */
  pb(Pos, Casella),!, write('P');      /* scrive P  dove c'è il pedone bianco   */
  onda(Pos, Onda),
  membro( Casella,Onda),!, write('X');
                                       /* scrive X  dove ci potrebbe  essere     */
                                       /* il re nero                             */
  write('.').                          /* scrive . dove non c'è nulla             */



stampa_a( Casella, Pos) :-
  rb(Pos, Casella),!, write('R');      /* scrive R  dove c'è il  re bianco        */
  pb(Pos, Casella),!, write('P');      /* scrive P  dove c'è il pedone bianco   */
  rn(Pos, Casella),!, write('X');      /* scrive X  dove c'è il re nero           */
  write('.').                          /* scrive . dove non c'è nulla             */

grafici :-
      write('~L'),
      crwind(giocatore,2,12,8,17,0,16,1),
      crwind(set_arbitro,2,50,8,17,0,16,1),
      crwind(bassa,14,2,9,76,0,0,16),
      crwind( num_g,2 ,31,8,1 ,0,5,16),
      crwind(lett_g,11,12,1,17,0,5,16),
      crwind( num_a,2 ,69,8,1 ,0,5,16),
      crwind(lett_a,11,50,1,17,0,5,16),
      crwind(tratto,2,37,3,9,0,8,16),
      cuwind(tratto),
      write('Muove il'),
      cuwind(lett_g),
      write(' a b c d e f g h '),
      cuwind(num_g),
      write('87654321'),
      cuwind(lett_a),
```

```
      write(' a b c d e f g h '),
      cuwind(num_a),
      write('87654321'),
      show(_..0:0..0:0..[]..[]..0:0).







/*  ---------------------------------------------------------------------------------------------------------------  */
/*                         Procedure per l'inserimento della posizione                         */
/*  ---------------------------------------------------------------------------------------------------------------  */


inserisci_pos(Chimuove..RB..PB..Onda..Tabù..RN) :-
    cuwind(bassa),
    write('Ora dobbiamo stabilire il set-informativo'),
    write('  del giocatore artificiale'),
    ins_chimuove(Chimuove),
    ins_rb(RB),
    ins_pb(RB,PB),
    ins_onda(RB,PB,Onda),
    ins_rn(Chimuove,RB,PB,Onda,RN),
    Tabù = [].



ins_chimuove(Chimuove) :-
  nl,
  write('Chi deve muovere per primo ? (se bianco digita "b", se'),
  write(' nero digita "n") '),
  leggi(Chimuove),
  cuwind(tratto),
  cursor(tratto,2,1),
  (Chimuove == b, write('BIANCO'); write(' NERO')),
  cuwind(bassa).


  leggi(Chimuove) :-
    get0('TRM:',Key),
    (Key == 98, Chimuove = b, ! ;
    Key == 110, Chimuove = n, ! ;
    leggi(Chimuove)).

ins_rb(RB) :-
  cursor(bassa,1,0), write('~K'),
  write('Dammi le coordinate del re bianco (es. e:5.) :  '),
  read(Coord),
  (traduci(Coord,RB),! ;
   write('I dati sono privi di significato'),beep(200,500),get0('TRM:',Key),
   ins_rb(RB)),
```

```
  show(_..RB..0:0..[]..[]..0:0).

ins_pb(RB,PB) :-
  cursor(bassa,2,0), write('~K'),
  write('Dammi le coordinate del pedone bianco (es. e:6.) :  '),
  read(Coord),
  (traduci(Coord,Pb),! ; nl,
   write('I dati sono privi di significato'),beep(200,500),get0('TRM:',Key),
   ins_pb(RB,Pb)),
  (RB \== Pb,Pb = Pb1,! ; nl,
   write('Attento : la casa '),
   write(Coord),
   write(' è gia occupata dal re bianco'),beep(200,500),get0('TRM:',Key),
   ins_pb(RB,Pb1)),
   (col(Pb1,Col), Col \== 1, PB = Pb1,! ; nl,
   write('Non puoi mettere il pedone gia in ottava '),beep(200,500),
   get0('TRM:',Key),
   ins_pb(RB,PB)),
   show(_..RB..PB..[]..[]..0:0).



ins_onda(RB,PB,Onda) :-
  cursor(bassa,3,0), write('~K'),
  write('Dammi la lista di coordinate in cui potrebbe collocarsi il re nero '),
  nl,write('es. [a:3,a:4,a:5]'), nl,
  leggilista(Lista),
  (verificalista(RB,PB,Lista,Lista1), Onda = Lista1,! ; nl,
   write('I dati o parte di essi sono privi di significato'),
   beep(200,500),get0('TRM:',Key),
   ins_onda(RB,PB,Onda)),
   show(_..RB..PB..Onda..[]..0:0),!.

verificalista(RB,PB,[],[]).

verificalista(RB,PB,[X|Rest],[Y|Coda]) :-
  traduci(X,Y),
  RB\==Y,
  PB\==Y,
  not adiac(RB,Y),
  verificalista(RB,PB,Rest,Coda).

leggilista(Lista) :-
   cursor(bassa,5,0),write('~K'),
   read(A),
   (A \== [], Lista = A ;
    write('Non cercare di fregarmi !'),beep(200,500),get0('TRM:',Key),
    leggilista(Lista)).

ins_rn(Chimuove,RB,PB,Onda,RN) :-
  write('~L'),
  write('Dammi le coordinate del re nero'),nl,
  write('Ti ricordo che :'),nl,
  write('Questa informazione rimarrà sconosciuta al giocatore artificiale'),
```

```
  nl,write('Le coord. del re nero devono essere incluse nella Lista '),
  read(Coord),
  (traduci(Coord,Rn),! ; nl,
   write('I dati sono privi di significato'),
   ins_rn(Chimuove,RB,PB,Onda,Rn)),
  (RB \== Rn,PB \== Rn, Rn = Rn1, ! ; nl,
   write('Attento : la casa '),
   write(Coord),
   write(' è gia occupata da un pezzo bianco'),
   ins_rn(Chimuove,RB,PB,Onda,Rn1)),
   (member(Rn1,Onda), Rn2 = Rn1, ! ; nl,
   write('Attento : La casella occupata dal re nero deve essere'),
   write('  inclusa nella lista Onda'),
   ins_rn(Chimuove,RB,PB,Onda,Rn2)),
   (Chimuove == n, RN = Rn2, ! ;
    not sottopedone(PB,Rn), RN = Rn2 ;
    write('Il nero non può rimanere sotto scacco se tocca al bianco'),
    ins_rn(Chimuove,RB,PB,Onda,Rn3)).

leggimossa(Part-Arriv) :-
  read(P-A),
  traduci(P,Part),
  traduci(A,Arriv).

traduci(Lett:Num,Col:Rig):-
  charof(Lett,L),
  Col is L-96,
  Rig is -(Num-9),
  casella_legale(Col:Rig).

traduci1(Col:Rig,Lett:Num):-
  L is Col+96,
  charof(Lett,L),
  Num is -(Rig-9).

scrivi(Part-Arriv) :-
  traduci1(Part,Part1),
  traduci1(Arriv,Arriv1),
  write(Part1-Arriv1).

chiudi :-
  get0('TRM:',Key),
  cuwind(&:),
  close(bassa),
  close(num_g),
  close(lett_g),
  close(num_a),
  close(lett_a),
  close(set_arbitro),
  close(tratto),
  close(giocatore).
```

Bibliografia del capitolo 3:

Bratko I. *Prolog programming for artificial intelligence.* Addison Wesley (1986)

Ciancarini P. Gaspari M. "*A Knowledge-based system and a development interface for the middle game in chess*" in Advances in computer chess 5 (1989) pagg 219-230.

Clocksin W.F. Mellish C.S. *Programmare in prolog* (Trad it. Programming in prolog) Ed Franco Angeli (1986)

Coelho H. Cotta J. *Prolog by example* Springer Verlag (1988).

Rich E. *Intelligenza Artificiale* (Trad it. Artificial Intelligence) Mc Graw Hill Italia (1986)

Capitolo 4

Considerazioni conclusive

Dal nostro viaggio nel microcosmo del kriegspiel, abbiamo potuto constatare come il gioco in questione sia una metafora molto più ricca e suggestiva del gioco degli scacchi, per rappresentare un contesto decisionale complesso.

Il limite essenziale del gioco degli scacchi come rappresentazione di un contesto decisionale reale è di cogliere solo un parziale aspetto dell' incertezza, e di questo era cosciente anche H. Simon, che molto spesso adottò gli scacchi come modello da cui trarre l' ispirazione per le sue teorie.

" *La difficoltà dello scacchista nel comportarsi razionalmente non ha niente a che fare con l'incertezza "..." nel senso in cui questo termine è usato in economia o nella teoria statistica delle decisioni. "..."*

Quello a cui ci riferiamo con "incertezza" nel gioco degli scacchi o nella dimostrazione di teoremi è quindi l'incertezza introdotta in un ambiente perfettamente certo, dall' incapacità - incapacità computazionale - di accertare la struttura di tale ambiente " (*Simon 1972*).

La peculiarità del Kriegspiel di essere un gioco ad informazione incompleta, consente di riprodurre un contesto decisionale nel quale oltre all'incertezza derivante dal deficit computazionale, è presente un fattore di interdipendenza strategica con asimmetria nella distribuzione di

informazioni, che accresce la complessità del problema e ripropone anche l' incertezza così come viene intesa dalla teoria statistica delle decisioni.
Ci si pone allora un quesito : Nel contesto decisionale del Kriegspiel, caratterizzato da una duplice fonte di incertezza, per un giocatore è possibile scegliere la propria strategia in modo razionale ?
Nei capitoli precedenti abbiamo visto come sia possibile cercare una risposta a questa domanda da due differenti punti di vista.
Scegliere la strategia migliore ⇒ Approccio con razionalità sostanziale.
Come scegliere una buona mossa ⇒ Approccio con razionalità procedurale.

La possibilità di porsi di fronte ad un problema complesso in due differenti modi viene suggerita da Simon con particolare riferimento al gioco degli scacchi (*Simon & Schaeffer 1992*).
Nel momento in cui l'obiettivo è quello di ottenere risultati non solo descrittivi ma anche normativi, i limiti sopra accennati privano di efficacia l' approccio agli scacchi con razionalità sostanziale . Occorre però considerare che, l'uso della metafora scacchistica potrebbe essere fuorviante in quanto confina la razionalità sostanziale ad una funzione esclusivamente descrittiva, e altrimenti non può essere, dato che non esiste incertezza sull'interazione strategica degli avversari, e il gioco è dotato di un punto di equilibrio con strategie pure. (*Teorema di Zermelo*).
Il Kriegspiel, che come abbiamo visto riesce a cogliere entrambi gli aspetti dell'incertezza, è in grado di lasciare maggior spazio alla razionalità sostanziale, alla quale viene affidato un ruolo non solo descrittivo, ma anche normativo.

In particolare due modelli matematici, (giochi stocastici e giochi ricorsivi) si rivelano efficaci ed efficienti nella soluzione di alcuni sottoproblemi del gioco.
E' chiaro che la razionalità che guida la soluzione di tali problemi è di tipo sostanziale. In un particolare contesto (che nel nostro caso sono i problemi trattati nel Cap 2) è possibile controllare la complessità del problema inserendolo in un modello che, è bene sottolinearlo,entro certe condizioni è in grado di suggerirci la strategia migliore.
La metafora del Kriegspiel riesce a farci intuire che le soluzioni ottenute con razionalità sostanziale (es. una strategia *ε-ottimale* alla "Blotto"), per quanto suggestive, sono proponibili solamente in situazioni costruite "ad hoc"; si intuisce inoltre come all'aumentare della complessità ma anche solo della *complicazione*[52] di un contesto decisionale, il potere normativo dei risultati ottenibili con razionalità sostanziale, perde via via di efficacia, ferma restando la capacità descrittiva in grado di consentire una comprensione e formalizzazione in termini rigorosi del problema.
Al fine di ottenere delle indicazioni normative, entriamo così nell'ottica della razionalità procedurale, vale a dire : pensare a ciò che è ragionevole fare quando è impossibile stabilire quale sia la decisione migliore; il baricentro della nostra attenzione si sposta dalla ricerca della strategia migliore, alla ricerca di un metodo in grado di "produrre" alternative soddisfacenti.
Nel capitolo 3 tentiamo questo approccio con un problema in cui è presente solo un numero ridotto di forze in campo, ma che è sufficientemente complesso da rendere frustrante ogni tentativo di individuare una strategia di

[52] Un modo per definire l'incertezza derivante da incapacità computazionale.

tipo sostanzialmente razionale: si tratta del finale R+P contro R nella versione completa.

La formalizzazione dei risultati ottenuti con un approccio proceduralmente razionale ha come sbocco naturale la creazione di un giocatore artificiale in grado di condurre autonomamente, e se possibile di vincere una qualunque posizione del tipo R+P contro R.

Esaminando le procedure che costituiscono la base portante del programma, emergono alcune considerazioni che evidenziano il pregio della metafora del kriegspiel.

Importanza della base di conoscenza

Abbiamo visto come la conoscenza di un problema sia un elemento discriminatorio, riguardo la validità di un approccio proceduralmente razionale. ("More Knowledge, more success" *Ciancarini 1989*).

Nel nostro esempio gran parte del bagaglio di conoscenza che guida il giocatore artificiale è stato ottenuto per mezzo di analogie tra il finale nel gioco degli scacchi e nella versione kriegspiel.

L'esperienza maturata in tale ambito ci porta ad azzardare delle considerazioni collegate alla teoria classica dei giochi e alle recenti direzioni di ricerca. Da tempo, infatti tale teoria si è posta degli interrogativi ancora senza una definitiva risposta come : "Quando va usata l'analisi di equilibrio?" (la ricerca di uno o più equilibri di Nash), " o "Come si comportano i giocatori quando non è possibile applicare l'analisi di equilibrio?" o ancora "Se l'analisi di equilibrio non è appropriata quali (utili) nozioni si possono usare al suo posto?". (*Kreps 1990*).

Appare evidente che gli scacchi e a maggior ragione il kriegspiel, sono dei giochi per i quali l'analisi di equilibrio è inadeguata. Un' analisi di questi giochi con ottica proceduralmente razionale trascura il concetto di equilibrio, ma dalla implementazione delle "procedure razionali" in macchine in grado di giocare autonomamente, possiamo constatare un effettivo *svolgimento* del gioco inteso come percorso lungo l'albero che lo rappresenta. Tale svolgimento non è casuale, ma è funzione in primo luogo della conoscenza, sia essa formalizzata sotto forma di funzione di valutazione (giocatori artificiali di scacchi) o come "knowledge base" nel (giocatori artificiali di kriegspiel).
Quest' affermazione è suggerita da alcune constatazioni empiriche nell'ambito dei giocatori artificiali. Ad esempio un giocatore artificiale posto di fronte ad una posizione scacchistica gioca sempre allo stesso modo[53], (il suo modello proceduralmente razionale è statico e valuta sempre la strategia che ritiene migliore migliore). Partendo da una certa posizione, due giocatori artificiali giocano sempre le stesse mosse, generando una partita che è sempre la stessa.
Esistono modelli di decisione proceduralmente razionali più efficienti di altri, lo dimostra il fatto che tra programmi girano sullo stesso PC, (quindi a parità di potenza computazionale) ve ne sono alcuni che emergono nei risultati vincendo sistematicamente contro gli altri.
Il programma che gioca il finale di kriegspiel vince in quanto è dotato di una sufficiente base di conoscenza, ma un qualunque altro giocatore che ne sia privo, molto difficilmente riuscirà a giungere alla vittoria.

[53] Eccettuata la fase di apertura negli scacchi in cui la variante che si giocherà viene scelta casualmente tra una numerose libreria di "strategie di apertura"

Se quindi non è opportuno parlare di analisi di equilibrio, si può constatare che tra individui proceduralmente razionali esiste una correlazione tra svolgimento del gioco e formalizzazione di modelli di razionalità procedurale.
In tale correlazione la nozione di conoscenza gioca sicuramente un ruolo determinante.

Il programma evita di formulare una strategia nel lungo periodo, sostituendola con la capacità di adattamento a situazioni fortemente dinamiche e imprevedibili.
Risulta suggestivo un parallelo tra il comportamento utilizzato per affrontare la complessità del kriegspiel e quanto asserito dalle teorie del *management science* riguardo le filosofie di pianificazione strategica in relazione all ambiente.
In particolare : come l'esercizio del *Long Range Planning* abbia successo nell' ambiente complicato che caratterizza gli anni '50 e '60, mentre nel nuovo ambiente economico che si afferma con gli anni '80 ed assume i connotati di ambiente complesso " *i processi di pianificazione debbono porsi orizzonti temporali più raccorciati e darsi una nuova "filosofia" basata sulla rapidità di reazione ai segnali (spesso di non facile lettura ed interpretazione) provenienti dall' ambiente...*" (*Faccipieri 1989*).
Ricordiamo come nel gioco degli scacchi (ambiente complicato) gli algoritmi con approccio "forza bruta", abbiano come punto di forza " l'esplorazione nel futuro" in un ambiente certo. Spesso vince chi è in grado di prevedere un numero più elevato di mosse e contromosse. Nel programma

esposto nel cap. 3 il giocatore artificiale non si sforza di prevedere il futuro, o almeno lo fa in misura estremente ridotta.
L'incertezza in cui si trova è così forte da rendere dispendiosa e inefficiente la costruzione di un albero di gioco. Quello che le routine del programma cercano di eseguire è un operazione di matching tra la situazione corrente e il data base di conoscenza: una sorta di capacità di adattamento ai rapidi sconvolgimenti che il set informativo viene a subire ad ogni stadio del gioco.

Risulta interessante verificare l'esistenza di alcuni obiettivi che si affiancano alla base di conoscenza è che costituiscono il motore inferenziale, che suggerisce la scelta di una strategia.
Obiettivo minimo.
La scelta di ogni strategia, in questo approccio è sempre guidata da un *obiettivo minimo*, che pone come vincolo essenziale il mantenimento di un certo stato al termine di ogni stadio del gioco. Per essere più chiari : nel nostro problema l' obiettivo minimo è spesso quello di non perdere il pedone; esso però si modifica nel corso del gioco diventando più ambizioso[54]. E' d'obbligo a questo punto un parallelo con il *soddisfacentismo* di Simon, ed in particolare con il concetto di variabilità del livello di aspirazione in un processo di decisione.

Gestione dell' incertezza.
Nel programma una delle euristiche che guida le scelta della mossa è quella che privilegia le strategie in grado di ridurrre l'incertezza, che nel nostro

[54] In particolare diventa: "non perdere il pedone ed evitare lo stallo"

problema si manifesta come numero di case in cui potenzialmente può collocarsi il re avversario[55].
E' interessante notare come l'incertezza sia un fenomeno che in molte occasioni offre delle opportunità altrimenti improponibili. Pensiamo al finale R+P contro R nel gioco degli scacchi tradizionali, in cui l' assenza di incertezza non permette al giocatore in vantaggio di di materiale la concretizzazione in termini di vittoria[56].
Nel kriegspiel invece, lo stesso vantaggio (ricordiamo che il bianco dispone di maggiori forze in campo, per la precisione di un pedone) può essere sfruttato grazie ad un' abile "gestione dell'incertezza". Se l' incertezza è notoriamente una caratteristica indesiderata dell'ambiente, non bisogna trascurare che essa si manifesta anche al nostro avversario, il che non può che giocare a che a nostro favore. La distribuzione asimmetrica delle informazioni è una peculiarità dei giochi ad informazione incompleta, e il kriegspiel sembra indicarci che operando in modo di minimizzare la nostra incertezza e massimizzare l'incertezza del nostro avversario, si possono ottenere delle opportunità che in un contesto di informazione perfetta sarebbero escluse.

Vi è un ultima considerazione che a mio avviso meritevole interesse.
Nell'approccio con razionalità procedurale abbiamo potuto vedere come i risultati ottenuti con un metodo razionale in modo sostanziale non siano stati ignorati. Al contrario sono stati adottati in maniera diretta ed indiretta hanno contribuito ad una soluzione più efficiente del problema.

---

55 Nel programma è dato dalla dimensione dell lista [Onda].
56 Vedi Cap 2 par. 7

Con un *adozione in maniera diretta*, ci riferiamo all' opportunità di trasportare i risultati normativi, nel data base di conoscenze che deve comprendere un decisore proceduralmente razionale. Nel problema che abbiamo affrontato probabilmente si sarebbe giunti ad una soluzione di una situazione "alla blotto", anche attraverso altre strade, ma sicuramente i risultati non sarebbero stati altrettanto efficienti.

Per quanto riguarda l' *adozione in maniera indiretta* occorre ammettere che la razionalità sostanziale offre un contributo insostituibile per una comprensione profonda del problema: essa permette di organizzare in modo ancora più effiiente le conoscenze acquisite.

Se razionalità sostanziale e procedurale hanno quasi sempre viaggiato su binari separati, forse anche perchè sviluppatesi in due aree scientifiche con tradizioni molto differenti (scienze economiche e psicologia) la matefora dello kriegspiel sembra darci l'indicazione che la strada migliore per il decisore che vuole scegliere la mossa migliore è quella di attingere da entrambe.

La strada che porta ad una teoria dei giochi con razionalità procedurale è comunque ancora lunga e tortuosa.

Appendice 1

Descrizione e regole del "Kriegspiel".

1. Descrizione del gioco.

Kriegspiel è una variante eterodossa degli scacchi che ha la peculiarità di essere un gioco ad informazione incompleta.

Si sa che fu inventato in Svizzera agli inizi del 1800; dopo la guerra franco-prussiana del 1870 venne divulgato in Inghilterra da un giornalista, M. Henry Temple che se ne attribuì l'invenzione. Il gioco veniva anche utilizzato nelle accademie militari prussiane per stimolare le capacità di deduzione degli ufficiali (non a caso Kriegspiel significa in tedesco "gioco di guerra").

La partita si svolge su tre scacchiere, una per il bianco, dove ci sono solo i pezzi di colore bianco; una per il nero, dove ci sono solamente i pezzi di colore nero, e una per l'arbitro. Le scacchiere utilizzate dai due giocatori sono disposte in modo che ciascuno dei due non possa vedere l'altro. La scacchiera dell'arbitro è l'unica che contiene i pezzi di ambedue i colori , viene utilizzata per controllare la legalità delle mosse proposte dai giocatori e non è visibile agli stessi.

fig A.1 Scacchiere rispettivamente del bianco, nero, e arbitro.

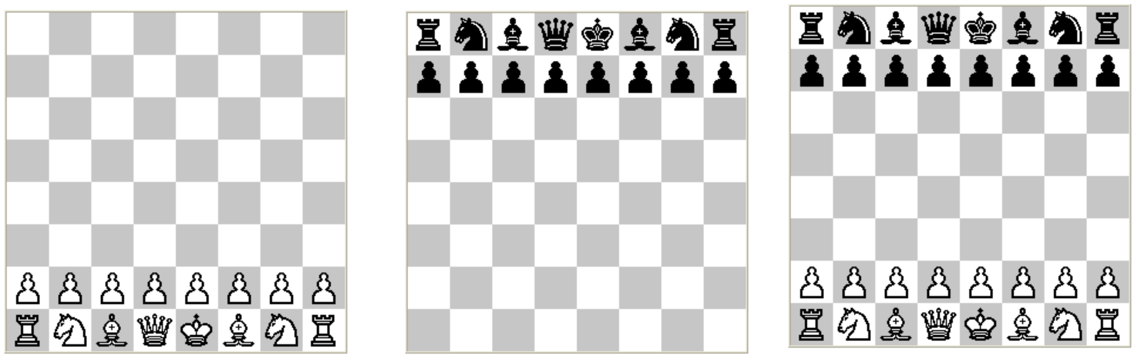


Fig. A.1

Ci si può rendere conto che i giocatori non hanno una visione completa della scacchiera e possono solo ipotizzare la posizione dei pezzi avversari utilizzando le informazioni che via via vengono fornite dall'arbitro.
Non esiste un regolamento universalmente accettato, quindi il gioco si diversifica per sottili sfumature a seconda delle fonti che si consultano. Restano comunque comuni a tutti le seguenti norme:

1. Le regole di movimento dei pezzi sono identiche a quelle degli scacchi: quindi una mossa sarà illegale se incompatibile con le regole del gioco.
2. Il giocatore di turno *comunica* la propria mossa all'arbitro senza farla conoscere all'avversario.
3. Se l'arbitro dichiara al giocatore di turno che la mossa giocata è illegale, il giocatore deve tentare un'altra mossa, e torna al punto 2.
4. Se la mossa è legale, l'arbitro *annuncia ad entrambi i giocatori,* che la mossa è stata giocata specificando quanto segue:
   4.1. L' esecuzione di una cattura, indicando la casa in cui è avvenuta, senza chiarire il tipo di pezzo catturato e catturante.
   4.2. La possibilità di effettuare catture di pedone, senza specificare quali. In questo caso il giocatore deve tentare almeno una cattura

di pedone, se quella tentata non è legale può giocare qualsiasi altra mossa, ivi compresa un'altra cattura di pedone.

4.3. La constatazione di uno scacco, indicando se è avvenuto sulla *traversa*[57] o sulla *colonna*, sulla *diagonale lunga* o *corta* passante per la casa occupata dal Re, o se si tratta di scacco di cavallo.

4.4. Non viene annunciato il tipo di pezzo che ha dato scacco. Se si tratta di *scacco doppio* vengono specificate tutte e due le minacce secondo i criteri di 4.3.

4.5. Il termine della partita, per *scacco matto* o per *stallo*.

Il gioco nella versione americana della RAND (Un' istituzione del Dipartimento della Difesa americana, in cui hanno lavorato Von Neumann, Simon, Shapley), che non si discosta particolarmente da quello sopra descritto, ha suscitato parecchio interesse nell'ambito di studiosi della teoria dei giochi. D'ora in avanti la nostra attenzione si concentrerà sulle posizioni finali, con vantaggio di materiale da parte di uno dei contendenti. (Es. Donna + Re contro Re, Torre + Torre + Re contro Re).

Per poter affrontare in modo non ambiguo questo argomento, necessitiamo di un linguaggio che ci permetta di descrivere le mosse che ciascuno dei giocatori può intraprendere.

Le caselle della scacchiera sono codificate con una notazione algebrica, nella quale colonne e righe sono indicate rispettivamente da lettere (a..h) e numeri (1..8); di conseguenza, ad esempio, l'angolo in alto a destra sarà "h8", mentre quello in basso a sinistra "a1". (Vedi fig. A.2)

[57] I termini in corsivo fanno parte della "terminologia scacchistica",per la quale si rimanda al paragrafo successivo.

Il movimento dei pezzi si rappresenta indicando il pezzo che viene mosso e la casella di arrivo.

Queste sono le abbreviazioni che indicano i pezzi :

R = Re	D = Regina	T = Torre
A = Alfiere	C = Cavallo	P = Pedone

Lo stato di conoscenza della posizione per la parte che deve muovere, viene rappresentato da una matrice 8 x 8, all'interno della quale ogni lettera in una singola casa indica il pezzo da cui essa è occupata.

Poichè non conosciamo lo schieramento delle forze avversarie, contrassegneremo con una "X" ogni casella nella quale è possibile che un pezzo si trovi.

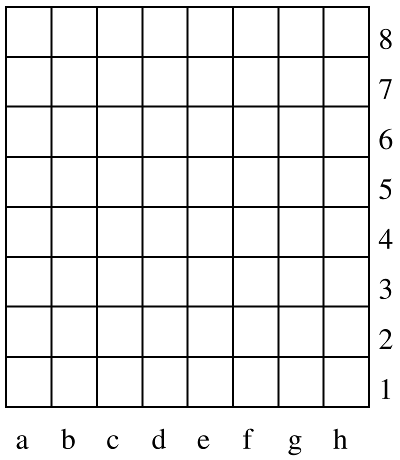


fig A.2

Dato che ci limiteremo ad analizzare posizioni nelle quali uno dei due giocatori combatte contro un solo pezzo avversario (il re), è chiaro che ogni "X" indica una possibile casa in cui il re nemico è collocato.

Nella fig. A.3 viene raffigurata una rappresentazione statica di una posizione, per mezzo della quale si individua in modo esustivo il set di informazioni a disposizione del giocatore.

	X	X		X		X	X	8
			T					7
				R				6
								5
								4
D								3
								2
								1
a	b	c	d	e	f	g	h	

fig A.3

Il giocatore che deve muovere sa che può disporre di una regina, una torre e un re, rispettivamente dislocati nelle caselle "a3", "d7","e6": inoltre può osservare che il re avversario è piazzato in una delle caselle indicate con "X", esso non può stare nelle case "a8","d8","f8" in quanto sarebbe sotto il "tiro" dei pezzi nemici.

Rimane l'assoluta incertezza per quanto riguarda l'effettiva posizione in cui il Re avversario si trova: a questo punto spetta all'abilità del giocatore, stabilire la più valida strategia per ridurre tale incertezza al fine di conseguire la vittoria.

Per completare la descrizione della notazione occorre considerare l'aspetto dinamico del gioco, cioè la notazione delle strategie che il giocatore ha intenzione di adottare.

Per codificare un mossa, che parte da una posizione, si indicherà il pezzo che si vuole muovere e la casa di arrivo, cosi "Rf6" significa, in relazione alla fig. A.3 , che il Re ha intenzione di spostarsi dalla casella "e6" alla casella "f6".

Partendo dalla fig. A.4 cercheremo di utilizzare la notazione appena descritta per indicare la stategia vincente.

					X	X	X	8
							X	7
					R			6
								5
								4
				D				3
								2
								1
a	b	c	d	e	f	g	h	

fig A.4

La notazione indica la strategia vincente. Il giocatore che deve muovere, che d'ora in poi chiameremo "bianco" per comodità, *comunica* all' arbitro Rf7 : se l' arbitro *annuncia* che la mossa è legale, allora significa che il Re avversario (nero), si trova in h8 o in h7, quindi prescindendo dalla successiva risposta del nero (in quanto ovunque esso sia al massimo potrà muoversi in h7 o h8) , il bianco può proseguire con lo scacco matto giocando Dh3 (il simbolo !! significa proprio scacco matto) .

Se invece Rf7 è illegale, allora il bianco può tentare di giocarne un'altra, ma con il vantaggio di aver acquisito delle preziose informazioni sulla posizione del Re nero che è sicuramente in f8 o in g8; tale ipotesi è infatti perfettamente compatibile con l'illegalità della mossa Rf7 da parte del Re bianco. (Fig A.4.b)

a	b	c	d	e	f	g	h	
					X	X		8
								7
					R			6
								5
								4
				D				3
								2
								1

Nuovo set informativo dopo la constatazione di illegalità di Rf7.

fig A.4.b

La mossa vincente che dovrà essere giocata dal bianco è De7.

Infatti, se De7 è <u>legale senza scacco</u>: il Re nero è sicuramente in g8 e sarà obbligato a muovere in h8 e verrà mattato alla successiva con Dg7.

Se De7 è <u>legale con scacco</u>: si deduce che il Re nero è in f8, dovrà allora muovere in g8 e verrà mattato alla successiva con Dg7.

Nella fig A.4.c abbiamo la descrizione della strategia.

```
Gioca Rf7
   If legale
   then attendi risposta nero e gioca Dh3 !!
   else gioca De7
        if legale con scacco or legale senza scacco
          then attendi risposta nero e gioca Dg7 !!
```

Fig A.4.c

E' interessante notare come questa strategia sia vincente con probabilità 1 (dato che la posizione è molto semplice), ma potrebbero esistere altre strategie che contemplano una rapida vittoria con probabilità α, mettendo a repentaglio la vita dei propri pezzi con probabilità $(1-\alpha)$.

Nella fig A.5.1 viene raffigurata una posizione nella quale il bianco può decidere di utilizzare una strategia che lo può condurre alla vittoria immediata con probabilita α, oppure ad un grave danno con probabilità $1-\alpha$.

	X	X	X	X	X	X		8	Da8 !! $\alpha = 5/6$
							T	7	$(1-\alpha) = 1/6$
						R		6	
								5	
								4	
								3	
						D		2	
								1	
a	b	c	d	e	f	g	h		fig A.5

Riguardo al posizionamento del re nero esiste una situazione di incertezza, nella quale, a differenza di una situazione di rischio, non siamo in grado di probabilizzare ogni singolo evento. Al più possiamo congetturare una equiprobabilità tra le 6 caselle che possono ospitare il re avversario (criterio di scelta della "ragione insufficiente" in condizioni di incertezza). Qualora giocassimo Da8, vinceremmo con P = 5/6 , perchè nel caso in cui il Re nemico fosse nell insieme di case {c8,d8,e8,f8,g8), non avrebbe la possibilità di catturare la regina del bianco. Se però il nero fosse in b8, (e questo era stimato dal bianco con P = 1/6), il Re nemico potrebbe catturare la regina alla mossa successiva, e il bianco pagherebbe caro il prezzo del rischio. (fig A.5.2)

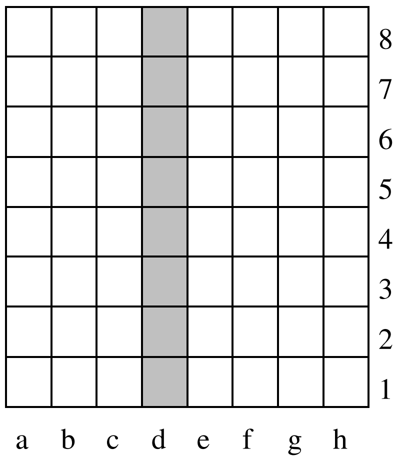
X								8
							T	7
						R		6
								5
								4
								3
								2
								1
a	b	c	d	e	f	g	h	

fig A.5.2

2. Terminologia scacchistica

Data la frequente ricorrenza di termini scacchistici, è sembrato opportuno riportare un breve glossario in grado di chirire alcune espressioni.

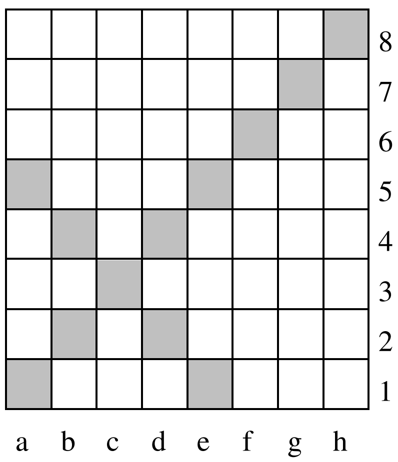
Colonna : L' insieme di caselle che percorre verticalmente la scacchiera



L'insieme di caselle ombreggiate indica la colonna "d".

fig A.6

Diagonale : L'insieme di caselle che percorre obliquamente la scacchiera, ogni casella è attraversata da due diagonali, una lunga e una corta; non è possibile che una casella venga attraversata da due diagonali della stessa lunghezza. (Fig A.7)



La casella c3 è attraversata dalla diagonale lunga a1-h8 e dalla diagonale corta e1-a5.

fig A.7

Scacco doppio : Si ha scacco doppio quando il re, a seguito di una mossa, è contemporaneamente sotto la minaccia di due pezzi nemici. (Fig A.8)

◦				X				8
	◦						◦	7
◦	◦	◦				◦		6
◦	R	◦	◦		◦			5
◦	◦	◦		A				4
			◦	◦	◦			3
		◦		◦		◦		2
◦	◦	◦	◦	T	◦	◦	◦	1
a	b	c	d	e	f	g	h	

Partendo dalla posizione a sinistra, dopo la mossa Ag6, il re nero (X) è sotto scacco doppio (T+A).

				X				8
				◦	◦		◦	7
◦	◦	◦		◦		A		6
◦	R	◦		◦	◦		◦	5
◦	◦	◦		◦				4
			◦	◦				3
		◦		◦				2
◦	◦	◦	◦	T	◦	◦	◦	1
a	b	c	d	e	f	g	h	

fig A.8

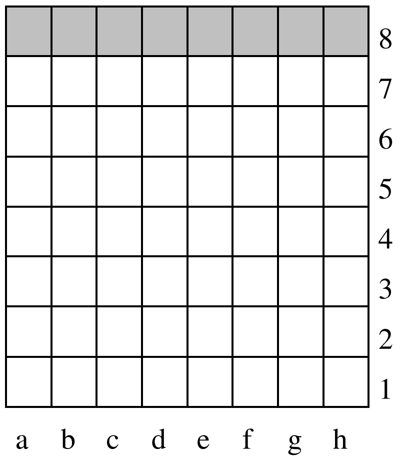
Stallo : Con il termine stallo si indica una posizione in cui il giocatore che deve muovere non ha a disposizione mosse legali. In caso di stallo, la partita finisce patta. (Fig A.9)

X	◦							8
◦	◦							7
R	◦							6
◦	◦							5
	◦							4
◦	T	◦	◦	◦	◦	◦	◦	3
	◦							2
	◦							1
a	b	c	d	e	f	g	h	

Una posizione di stallo: tocca ad X che però è immobilizzato, poichè tutte le case che potrebbe raggiungere sono sotto il tiro dei pezzi nemici.

fig A.9

Traversa : L'insieme di caselle che percorre orizzontalmente la scacchiera(fig.A.10)



L'insieme di caselle ombreggiate indica l' "ottava traversa".
fig A.10

Appendice 2

Teoria dei giochi e definizioni di base

---

1. Rappresentazione di un gioco in forma estesa

In base alla definizione di Kuhn (1953) possiamo definire un gioco con n-persone in forma estesa **Γ** costituito da:

1) Un insieme **N** = {1,2,..,n} di giocatori.
2) Un albero **T** detto "*albero di gioco*"
3) Una partizione dell'insieme dei nodi non terminali di **T** in *n+1* sottoinsiemi $P^0, P^1,...,P^n$. Gli elementi di P^0 sono chiamati "*chance nodes*" ; per ogni i ∈ **N**, gli elementi di P^i sono chiamati "nodi del giocatore i".
4) Per ogni nodo in P^0 una distribuzione di probabilità sui rami uscenti da esso.
5) Per ogni i ∈ **N** una partizione di P^i in k(i) "*insiemi di informazioni*" : $\mathbf{U}^1$, $\mathbf{U}^2,...,\mathbf{U}^{k(i)}$ tali che per ogni j = 1,2,..,k(i):

   a) tutti i nodi in $\mathbf{U}_j$ hanno lo stesso numero di rami uscenti e c'è una corrispondenza univoca con gli insiemi di rami uscenti dai singoli nodi componenti $\mathbf{U}_j$

   b) ogni percorso nell'albero dalla radice ad un nodo terminale può attraversare ogni $\mathbf{U}_j$ al massimo una volta sola.
6) Per ogni nodo terminale t ∈ L(T) un vettore n-dimensionale $g(t) = [g_1(t), g_2(t),.., g_n(t)]$ di payoffs.

Nella figura 1.B viene rappresentato un possibile gioco in forma estesa, le regole di questo gioco prevedono che il giocatore "1" sia il primo a muovere e il suo insieme di informazioni consista di un solo elemento.

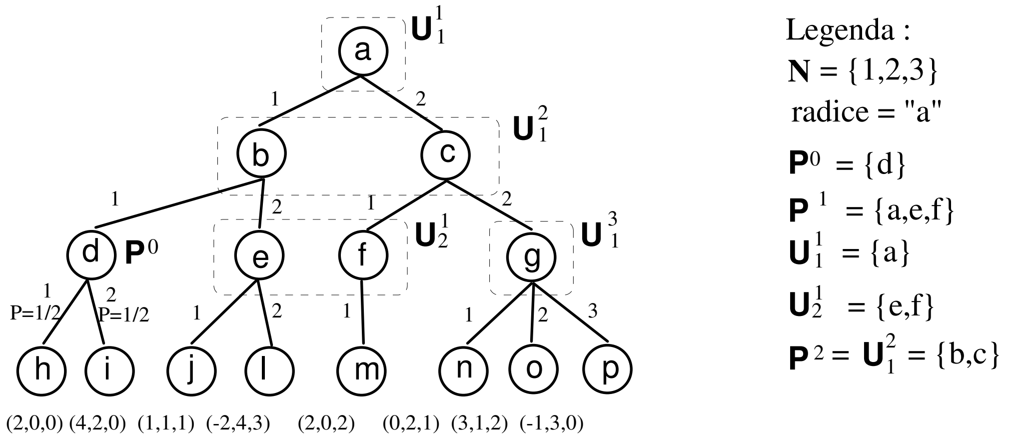


Fig 1.B

Il fatto che un insieme di informazione consista di più nodi significa che il giocatore associato a tale insieme non è in grado di identificare il nodo in cui esattamente si trova.

Da ogni nodo appartenente allo stesso insieme di informazione, partono gli stessi archi (Def. 5.a) proprio perchè il giocatore sa quali sono le mosse a sua disposizione, ma non sa da quale nodo scaturiscano. Tornando all'esempio, se il giocatore "2" decide di giocare la strategia 1, il gioco potrà proseguire in due diversi modi:

1) Se il giocatore "2" si trovava nel nodo "*b*", il gioco prosegue in "*d*".

2) Se il giocatore "2" si trovava nel nodo "*c*", il gioco prosegue in "*f*"

Il nodo "*d*", è definito "*chance node*", cioè un nodo in cui la scelta del nodo fuoriuscente non dipende dalla volontà dei giocatori, ma dal caso.

Ad ogni nodo terminale del gioco è associato un vettore di pay-offs, indicante l'utilità per ciascun contendente.

Assieme alla definizione di gioco, la teoria classica introduce gli assiomi che caratterizzano il comportamento dei giocatori :

a) Per ogni giocatore *i*, esiste una funzione di utilità lineare, definita sull'insieme dei nodi terminali dell'albero del gioco (sui payoffs).
b) Ogni giocatore è a conoscenza della struttura del gioco estensivo, ed è completamente informato sulle regole del gioco e sulle funzioni di utilità di ogni altro giocatore
c) Tra due alternative, un giocatore sceglierà sempre quella che massimizzerà la sua utilità attesa.

2. Rappresentazione di un gioco in forma normale.

Un gioco $\boldsymbol{\Gamma}$ di n giocatori è costituito da :

1) Un insieme N = {1,2,..,n} di giocatori.
2) Per ogni giocatore $i \in \mathbf{N}$, un insieme finito S^i di strategie pure.
3) per ogni giocatore $i \in \mathbf{N}$, una funzione $h^i : S \rightarrow \mathbf{R}$ detta funzione di payoff del giocatore *i*, i cui valori dipendono dalle strategie scelte dagli altri giocatori.

La rappresentazione in forma normale si presta molto bene ai giochi a due persone, che sono quelli di maggior interesse ai fini di questa dissertazione. Supponendo che le *strategie pure*[58] a disposizione due giocatori siano:

[58] Vedi definizione nel paragrafo seguente.

$S^1 = \{ \alpha_1, \alpha_2, ..., \alpha_m \}$

$S^2 = \{ \beta_1, \beta_2, ..., \beta_n \}$

Se il giocatore *1* sceglie α_i e il giocatore *2* sceglie β_j, il payoff sarà indicato dall'elemento H_{ij}.

Il tutto viene rappresentato (fig 2.B) da una matrice m x n.

	β_1	β_2	...	β_j	...	β_n
α_1	$H_{1,1}$	$H_{1,2}$	...	$H_{1,j}$	...	$H_{1,n}$
α_2	$H_{2,1}$	$H_{2,2}$	...	$H_{2,j}$	...	$H_{2,n}$
...	...	...	...	...	...	...
α_i	$H_{i,1}$	$H_{i,2}$	...	$H_{i,j}$	...	$H_{i,n}$
...	...	...	...	...	...	...
α_m	$H_{m,1}$	H_{m2}	...	$H_{m,j}$	...	H_{mn}

fig 2.B

Nella matrice di fig 2.B , la *i*-esima riga rappresenta la strategia pura del giocatore *1*, la *j*-esima colonna rappresenta la strategia pura del giocatore *2*, e il generico elemento H_{ij} indica il pay-off per entrambi i giocatori.

3. Definizione di strategia pura

Sia $I^i := \{ U^i_1, U^i_2, ..., U^i_{k(i)} \}$ l'insieme degli insiemi di informazione per il giocatore *i*.

Per ogni insieme di informazione U^i_j , sia $v = v(U^i_j)$ il numero di rami uscenti da ogni nodo in U^i_j , si numerino questi rami da 1 a v in modo da preservare una corrispondenza univoca tra gli insiemi di rami uscenti dai diversi nodi di U^i_j .

Detto ciò sia $C_i(U_j^i) = \{1,2,..,v\}$ l'insieme di scelte disponibili per il giocatore *i* in ogni nodo in U_j^i .
Una *strategia pura* è una funzione che assegna ad ogni $U_j^i \in I^i$ una scelta :

$$S^i\,(U_j^i) \in C_i(U_j^i\,)$$

L'insieme di tutte le strategie pure per il giocatore i è indicato con S .
Quindi se un giocatore dispone di k(i) insiemi di informazione, una strategia pura può essere rappresentata con un insieme di k(i) numeri dove l' n-esimo numero indica il ramo scelto quando e se il gioco raggiunge l' n-esimo insieme di informazione.
Tornando all'esempio di fig 1.B il giocatore *1*, dispone di due set di informazioni, uno iniziale corrispondente al nodo "a", ed un secondo qualora il gioco continuasse verso uno dei due nodi "e" o "f"; poichè da ciascuno dei due set di informazione si dipartono due possibili mosse, il numero totale di strategie pure a disposizione del giocatore "1" ammonta a 4.
Per cui $S^1 = \{(1,1),(1,2),(2,1),(2,2)\}$.

4. Definizione di strategia mista

Una *strategia mista* è una distribuzione di probabilità sull'insieme delle strategie pure S^i.
Nel caso in cui un giocatore disponga di m strategie pure, una strategia mista viene rappresentata da un vettore di m componenti $\overline{x} = (x_1,x_2,....,x_m)$ in cui :

$$x_i \geq 0 \quad e \quad \sum_{x=1}^{m} x_i = 1$$

5. Definizione di strategia comportamentale

Una *strategia comportamentale* assegna ad ogni insieme di informazione $U^i_j \in I^i$ una distribuzione di probabilità sull'insieme $C_i(U^i_j)$.

Si può pensare ad una *strategia pura* come ad un libro d'istruzioni, dove per ognuno degli insiemi di informazione dei giocatori c'è una pagina che indica la scelta deve essere fatta in quel momento. L'insieme di strategie pure è uno scaffale di libri d'istruzioni.

Una *strategia mista* è una distribuzione di probabilità che ci fa scegliere uno dei libri sullo scaffale.

Una *strategia comportamentale* è invece un libro d'istruzioni scritto in un altro modo: ogni pagina si riferisce ad un singolo insieme di informazioni per il giocatore, e in essa viene indicata una distribuzione di probabilità sulla scelta da fare in quel momento. Le strategia, dunque, cambia di volta in volta, non è quindi una scelta a priori come nella strategia pura. (*Hart 1992*)

6. Giochi a somma nulla , Giochi a somma non nulla

Un gioco Γ si dice a somma nulla, o strettamente competitivo, se per ogni elemento della matrice dei pay-offs si ha che $\alpha_{ij} + \beta_{ij} = 0$.

Vale a dire che il payoff per il giocatore *1* è l'opposto del giocatore *2*.

Sia gli scacchi che il kriegspiel sono un gioco a somma nulla, in quanto la vittoria per un giocatore (pay-off = 1) implica la sconfitta per l'altro (pay-off = -1), mentre in caso di parità il payoff è nullo per entrambi.

Per questi due giochi la funzione del pay-off può essere impostata in modo alternativo: ipotizzando che uno dei giocatori debba vincere (qualora ad

esempio disponesse di un notevole vantaggio di materiale), potremmo far corrispondere il suo pay-off al numero delle mosse necessarie per forzare la vittoria.
Anche con questa impostazione, il gioco rimarrebbe a somma zero, in quanto se ad esempio per il primo giocatore un pay-off = 5 significa "scegliendo quella strategia vincerai in 5 mosse", per il secondo giocatore quel pay-off significa la capitolazione in 5 mosse.
Allora l'interesse del giocatore *1* sarà quello di minimizzare tale pay-off, mentre quello del giocatore *2* sarà di massimizzare la sua sopravvivenza.

7. Giochi ad informazione perfetta, Giochi ad informazione imperfetta

Un gioco si dice ad informazione perfetta, se con riferimento alla rappresentazione in forma estesa, ogni insieme di informazioni consiste di un solo elemento, questo significa che, ciascun giocatore, nel momento in cui deve muovere, è a conoscenza del nodo in cui si trova all'interno dell'albero di gioco. Gli scacchi, la dama, il filetto ecc. sono giochi ad informazione perfetta, mentre il gioco del pari e dispari, il poker, e in certi casi il kriegspiel sono alcuni esempi di giochi ad informazione imperfetta .
La presenza di "*chance nodes*", 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

alcune posizioni del kriegspiel sono ad informazione imperfetta, nonostante la totale assenza di eventi casuali.

8. Giochi finiti e infiniti

Un gioco Γ si dice finito se, con riferimento alla rappresentazione in forma estesa, il suo albero contiene un numero finito di nodi. Gli scacchi sono un gioco finito, grazie alle regole che fermano il gioco dopo un certo numero di mosse. (Regola delle 50 mosse, patta per ripetizione di mosse o di posizione).
Anche il kriegspiel può essere considerato un gioco finito, anche se non esiste esplicitamente la regola delle 50 mosse, il limite è comunque considerato entro un numero "ragionevole" di mosse.

9. Giochi ad informazione completa e incompleta

In un gioco ad informazione completa ogni giocatore è a disposizione delle seguenti informazioni:

- il numero dei giocatori
- le strategie a disposizione di ciascuno dei giocatori
- i pay-offs dei singoli giocatori

Se uno o più giocatori non dispongono di tutte le sopraindicate informazione, il gioco si dirà ad informazione incompleta.

10. Giochi multistadio

Owen (*Owen 1982*) definisce "multistadio" i giochi nei quali il pay-off ha luogo solamente dopo un certo numero di iterazioni da parte dei giocatori. Ogni iterazione strategica da parte di due o più giocatori prende il nome di "stadio". Al termine di ogni stadio il set informativo di uno o più giocatori può essere modificato da nuove informazioni che sono sopraggiunte.
I giochi stocastici e i giochi ricorsivi rientrano nella categoria dei giochi multistadio. Si può immediatamente constatare come il kriegspiel sia un gioco multistadio.

11. Equilibri in un gioco strettamente competitivo.

Dato un gioco Γ , un profilo di strategie $(\sigma_{1*},\sigma_{2*},...,\sigma_{n*})$ si dice *in equilibrio* se e solo se per ogni $i \in \mathbf{N}$ e per ogni $\boldsymbol{\sigma}_i \in \boldsymbol{\Sigma}_i$ [59]

$$\Pi^i\ (\sigma_{1*},\ ...,\ \sigma_{i-1*},\ \sigma_i,\ \sigma_{i+1*},\ ...,\ \sigma_{n*}) \leq (\sigma_{1*},\ ...,\ \sigma_{n*})$$

Dove Π^i è il pay-off che risulta dalle strategie.

[5959] Σi è l'insieme di strategie a disposizione del giocatore *i*.

Detto in altri termini, un vettore di strategie è *in equilibrio* quando nessun giocatore trae vantaggio cambiando la sua strategia nell'ipotesi che i suoi avversari rimangano fermi nelle loro decisioni.
E' opportuno a questo punto aprire una parentesi sul significato che *Luce e Raiffa* danno al punto di equilibrio. Il fatto che in un gioco esista un punto di equilibrio non significa che la teoria dei giochi prescriva la strategia più corretta da giocare. La teoria dei giochi non cerca di dire cosa un giocatore dovrebbe fare, essa si limita ad informare il giocatore che scegliendo una certa strategia si può garantire un pay-off di sicurezza.
Se un giocatore non scegliesse la strategia corrispondente al punto di equilibrio, sarebbe errato definirlo "irrazionale" , in quanto egli potrebbe congetturare un comportamento "irrazionale" da parte del suo avversario e allora essere "irrazionali" sarebbe una scelta "razionale" (*Luce & Raiffa 1957*).
Non tutti i giochi hanno punti di equilibrio, possiamo allora dividere l'insieme dei giochi a somma zero in due classi, quelli con e quelli senza punto di equilibrio.

12. Giochi senza punti di equilibrio.

Se non esiste un punto di equilibrio, il gioco non si potrà stabilizzare su una coppia di strategie pure, in quanto, nel momento in cui uno dei due giocatori utilizzasse sistematicamente una strategia, l'avversario ne approfitterebbe a suo vantaggio.
Il *teorema del minmax (o di Von Neumann)* afferma che in qualunque gioco finito in forma normale esiste sempre un equilibrio con una strategia mista.

Esistono : un valore v , una strategia mista (maxmin) per il giocatore "1" la quale gli garantisce un payoff ≥ v , una strategia mista (minmax) per il giocatore "2" la quale gli garantisce un payoff ≤ -v. Le due strategie sono in equilibrio.

Questo teorema, è il più importante della teoria dei giochi ed è stato dimostrato in modi differenti (Von Neumann Morgenstern ; Nash). E'interessante notare che una strategia pura, altro non è che un caso particolare di strategia mista, nel cui vettore esiste un solo elemento con valore 1 e tutti gli altri con valore 0.