Politecnico di Milano

Facoltà di Ingegneria dell'Informazione

Corso di laurea in Ingegneria Informatica

COMPUTAZIONE DI EQUILIBRI PERFETTI NEI GIOCHI IN FORMA ESTESA A SOMMA ZERO CON DUE GIOCATORI: ALGORITMI E ANALISI SPERIMENTALE

Relatore: Ing. Nicola GATTI

Elaborato di Laurea di:

Matteo SANCINI

Matricola n. 734600

Anno Accademico 2010-2011

Alla famiglia e a tutte le persone che mi sono state vicine,

che mi hanno sempre sostenuto in tutti questi anni.

Grazie di tutto Cuore.

Matteo

Indice

Elenco delle figure

Capitolo 1

Introduzione

L'intelligenza artificiale (IA), ovvero la capacità di un computer di svolgere funzioni e ragionamenti tipici della mente umana ha negli ultimi anni avuto interesse particolare da parte di scienziati, filosofi e addetti del campo. Il termine fu coniato nel 1956 da John McCarthy [1] durante un seminario svoltosi nel New Hampshire. Fin dai primi anni sessanta lo scopo primario dell'IA era quello di costruire entità intelligenti che potessero prendere decisioni indipendenti, senza alcun bisogno di intervento da parte dell'uomo. Nel corso degli anni questa scienza è andata sempre più verso il suo perfezionamento ma ancora oggi possiamo definirla come una scienza nuova, di incredibili potenzialità, ma in cui ancora molto vi è da approfondire. Un particolare campo di interesse dell'IA riguarda i *sistemi multiagente* , ovvero lo studio sulla cooperazione o competizione di più agenti riguardo alla modellizzazione di un problema riscontrabile nella società di tutti i giorni. Con agente noi intendiamo, invece, un entità in grado di percepire l'ambiente che lo circonda e di prendere quindi delle decisioni sulla base di ragionamenti simili a quelli della mente umana. In questa tesi noi vogliamo focalizzarci in particolar modo su una disciplina dei sistemi multiagente. Questa disciplina prende il nome di *Teoria dei Giochi.*

La teoria dei giochi è la scienza matematica che analizza situazioni di conflitto e ne studia le soluzioni tramite determinati modelli tali a portare ad un soggetto coinvolto il massimo del guadagno possibile, tenendo conto delle decisioni di altri soggetti coinvolti a loro volta nel conflitto. Ogni soggetto partecipante esercita una sua azione tra le possibili scelte che ha a disposizione per raggiungere un proprio determinato obiettivo.

Questa scienza rientra in quel ramo dell'intelligenza artificiale che suscita al giorno d'oggi molto interesse per il suo possibile utilizzo in moltissimi campi, dal mondo economico-finanziario al mondo politico, dal mondo sociale al mondo militare.

In particolare la tesi sviluppata in questo contesto si preoccupa dello studio di interazione tra due agenti e nello specifico si occupa della risoluzione di giochi a due partecipanti a somma zero.

Questo ramo nell'immenso panorama delle possibili interazioni fra soggetti nella teoria dei giochi, si riferisce al particolare problema in cui la somma delle vincite dei due contendenti in funzione delle strategie utilizzate risulti sempre zero. In questo tipo di problemi, in pratica, uno dei giocatori sarà sempre vincente sull'altro ed è al più prevista la possibilità che il conflitto termini in pareggio, ma non è prevista la situazione in cui entrambi i giocatori possano perdere o vincere contemporaneamente. Le motivazioni che hanno portato allo studio di questo tipo di interazioni fra due agenti è dovuta al fatto che nella società ci sono moltissimi casi in cui è possibile

riscontrare un'interazione competitiva fra due persone in qualsiasi dei campi precedentemente citati ed inoltre lo studio di questo tipo di interazione può essere utilizzato come base e come punto di slancio per lo studio di interazioni fra agenti ben più complesse.

1.1 Oggetto di studio e contributi

Entrando in profondità nella tesi, la motivazione che ha spinto al suo sviluppo consiste nel risolvere il problema del conflitto che si crea tra due giocatori in giochi a somma zero in forma estesa e nel dettaglio di formulare quindi, un programma in grado di computare differenti alberi decisionali di giochi a due partecipanti già formalizzati e di differenti dimensioni con lo scopo di trovare dopo determinate valutazioni dell'albero stesso, in base alle strategie utilizzate, un equilibrio computazionale del suddetto.

Nel nostro problema in particolare, si è voluto tener conto di giocatori che potessero utilizzare diverse strategie decisionali possibili e quindi di risolvere il conflitto presente sulla base di un comportamento idoneo alla strategia presa in atto e per questo sono stati adottati concetti di soluzioni che non sono ancora stati studiati in letteratura.

Nello specifico, il nostro programma è stato implementato utilizzando il linguaggio di programmazione C e includendo in esso anche delle librerie adatte alla rappresentazione dei dati, caricati dal file contenente l'albero decisionale del gioco. Si è voluto tener conto di due diverse rappresentazioni dei dati stessi per poter valutare un diverso comportamento del programma e quindi avere maggiori valutazioni e dati sperimentali a riguardo. L'algoritmo di risoluzione dei suddetti giochi si basa fortemente sulla risoluzione di un problema a programmazione lineare intera e in particolar modo si affronta la sua risoluzione per mezzo del noto algoritmo del simplesso primale.

Un ultima considerazione di cui tener conto riguarda il problema di dover risparmiare spazio in memoria per la possibilità di dover caricare alberi di dimensioni davvero notevoli. Sono state implementate tecniche che potessero massimizzare il più possibile il risparmio ma che garantissero al contempo la massima velocità possibile nel calcolo dell'equilibrio computazionale.

In sostanza i contributi apportati riguardano:

- Adattamento dell'algoritmo di risoluzione del simplesso per giochi a somma-zero;
- formulazioni matematiche per la ricerca del normale equilibrio di Nash (NE);
- formulazioni matematiche per la ricerca di un equilibrio quasi perfetto (QPE) in giochi a forma estesa;
- formulazioni matematiche per la ricerca di un equilibrio perfetto in forma estesa (EFPE);

- risultati sperimentali di utilizzo dell'algoritmo nelle sue varie casistiche.

1.2 Struttura della tesi

La struttura della tesi è stata pensata in modo tale da poter far seguire, a una prima breve parte di introduzione in cui si spiega a grosse linee l'argomento su cui verte la tesi stessa e l'implementazione successiva che è stata affrontata sulla base del problema in questione, una parte in cui si parlasse in modo generale e via via più approfondito sulla teoria dei giochi.

Nello specifico si spiegherà che cosa si intende per gioco e se ne darà una sua formulazione sia in forma strategica, sia in forma estesa. Parleremo dei vari tipi di giochi presenti nella teoria soffermandoci in particolar modo sui giochi a somma zero. Discuteremo sugli equilibri in forma strategica e di conseguenza analizzeremo l'importante cardine rappresentato dal calcolo dell'equilibrio computazionale di Nash e analizzeremo le strategie che prendono il nome di MINMAX e MAXMIN. Successivamente parleremo invece degli equilibri appropriati per giochi in forma estesa e tratteremo in particolar modo il calcolo computazione degli equilibri discutendo delle varie tecniche adottate in tal senso e come esempio possiamo citare SPE, QPE e EFPE.

Dopo aver affrontato la teoria a livello generale scenderemo nel dettaglio per quanto riguarda il calcolo degli equilibri nei giochi a due giocatori a somma zero e vedremo la loro formulazione in forma strategica, in forma normale e in forma sequenza e al contempo tratteremo le tecniche sopra citate applicate ai vari tipi di formulazione.

Faremo seguire a tutto questo una trattazione approfondita dei giochi contraddistinti da perturbazione, in particolar modo la ricerca di un equilibrio in EFPE che come vedremo rifinirà la teoria sull'equilibrio di Nash. Discuteremo poi la formulazione per quanto riguarda il nostro caso specifico parlando in maniera dettagliata di come è stata affrontata l'implementazione, dei problemi che si sono presentati e di come sono stati affrontati.

Faremo terminare il tutto con i risultati delle prove sperimentali ottenute facendo girare il programma creato sui vari alberi decisionali presi in esame.

L'elaborato risulta quindi così strutturato:

2. **Stato dell'Arte**: in questo capitolo tratteremo le principali definizioni di gioco, inquadreremo le tematiche e faremo degli esempi generali per lo sviluppo del lavoro;

3. **Ricerca di un EFPE**: parleremo dell'equilibrio perfetto in forma estesa, faremo degli esempi sulla metodologia per ricercarlo e inseriremo un esempio;

4. **Algoritmo sviluppato**: approfondiremo nel dettaglio cosa ha comportato allo sviluppo del nostro algoritmo, le modifiche effettuate e la sua struttura;

5. **Valutazioni sperimentali**: in questo capitolo tratteremo i risultati ottenuti dal nostro algoritmo su determinati esempi di giochi e valuteremo le varie casistiche;

6. **Conclusioni**: riassunto di quanto presentato e possibilità di sviluppo per lavori futuri.

Capitolo 2

Stato dell'arte

In questo capitolo analizzeremo dettagliatamente la Teoria dei Giochi nel suo complesso avvalendoci di tutti gli strumenti che essa ci mette a disposizione per creare un agente in grado, attraverso un ragionamento, di trovare la strategia migliore per agire nell'ambiente in cui si trova. Ogni agente si suppone sia razionale ovvero egli cercherà di massimizzare sempre la sua utilità attesa indipendentemente dal fatto che esso possa causare danni o meno ai suoi avversari.

Nella sezione 2.1 tratteremo la teoria dei giochi in generale e le varie tipologie di gioco esistenti. Nella sezione 2.2, invece, parleremo delle prime formalizzazioni matematiche riguardanti la descrizioni di un gioco e delle sue caratteristiche. Nella sezione 2.3 spiegheremo i giochi in forma normale e ci soffermeremo in particolar modo sulla tipologia dei giochi a somma zero di interesse per lo sviluppo del nostro algoritmo. Nell'ultima sezione, la 2.4, tratteremo invece la formalizzazione dei giochi in forma estesa.

2.1 Teoria dei giochi

La teoria dei giochi è la scienza matematica che analizza situazioni di conflitto e ne ricerca soluzioni cooperative e competitive e studia le decisioni individuali in cui vi sia la presenza di diversi soggetti. Essa tiene conto del fatto che vi sono più giocatori e che l'esito finale del problema dipende dalle scelte che essi fanno durante il conflitto. Si assume spesso che i giocatori siano "intelligenti", cioè siano in grado di fare ragionamenti logici di complessità indefinitamente elevata, e siano "razionali", cioè hanno preferenze coerenti sugli esiti finali del processo decisionale e hanno l'obiettivo di "massimizzare" questa preferenza.

Definizione 2.1 (Concetto di gioco)

Un gioco rappresenta una situazione di conflitto in cui sono coinvolti diversi agenti e dove è necessario trovare soluzioni cooperative o competitive tenendo in considerazione le possibili interazioni fra i diversi soggetti. Esistono molti tipi di giochi tra cui giochi di carte, videogiochi, giochi sportivi, ecc. e le interazioni che si studiano in conflitti di questo tipo vengono poi utilizzate in moltissimi altri campi come ad esempio l'economia, la finanza, la politica e il militare. È possibile elencare tra gli altri giochi in cui partecipano due giocatori o anche più di due giocatori, giochi in cui è necessario tener conto delle strategie utilizzate dai vari soggetti, giochi in cui è possibile raggiungere diversi obiettivi e giochi in cui la vincita o la perdita dipendono anche dalle azioni degli

altri giocatori. Esistono quindi una notevole e svariata serie di casi possibili e ognuno di questi casi è oggetto di studio col fine di trovarne la migliore soluzione possibile per i soggetti coinvolti.

Un gioco in sé comprende il numero di giocatori partecipanti, una descrizione completa su ciò che i giocatori possono scegliere, quindi l'insieme di tutte le azioni possibili, le informazioni che i giocatori hanno a disposizione quando devono prendere una decisione, una descrizione delle possibili vincite di ogni giocatore per ogni possibile combinazione delle mosse scelte da tutti i giocatori che partecipano al gioco e una descrizione di tutte le preferenze dei giocatori sugli esiti.

2.1.1 Tipologie di gioco

Tra i tutti i possibili giochi inoltre è possibile fare una prima suddivisione tra giochi cooperativi e giochi non-cooperativi:

- Un gioco si dice cooperativo se c'è la possibilità per i giocatori di sottoscrivere accordi vincolanti, che possono essere di vantaggio ai singoli giocatori.

- Un gioco si dice non-cooperativo quando il meccanismo delle decisioni riguarda i singoli giocatori sulla base di ragionamenti vincolanti.

Una successiva suddivisone che è possibile fare riguarda i giochi a somma zero e i giochi a somma generale:

- Un gioco si dice a somma zero se la somma delle vincite è zero, ad esempio nel poker se un giocatore vince 100 euro, l'insieme delle perdite degli altri giocatori sarà 100 euro a sua volta.

- Un gioco si dice a somma generale quando non vi è un corrispettivo matematico sulle vincite o le perdite dei giocatori coinvolti.

Un'ulteriore caratterizzazione che possiamo trovare tra le tipologie dei giochi sono i giochi simultanei o i giochi sequenziali:

- Un gioco si dice simultaneo se i giocatori scelgono le azioni simultaneamente.

- Un gioco si dice sequenziale se i giocatori scelgono le azioni secondo una successione particolare.

Distinguiamo infine tra giochi a informazione perfetta e giochi a informazione imperfetta:

- Un gioco a informazione perfetta è quello in cui le regole del gioco e la funzione di utilità di tutti i giocatori sono conoscenza comune dei giocatori.

- Un gioco a informazione imperfetta è quello in cui le regole del gioco e la funzione di utilità dei giocatori non sono conoscenza comune dei giocatori.

Nella pratica e nella realtà è molto più realistico avere giochi a informazione imperfetta che giochi a informazione perfetta.

E' vero, però, che molte strategie comprendono sia la simultaneità che la sequenzialità delle azioni ed i giochi ad informazione imperfetta catturano tutti questi dettagli modellistici.

L'obiettivo quindi della teoria dei giochi e della ricerca sui sistemi multi-agente è quello di trovare dei metodi per costruire sistemi complessi composti di agenti autonomi che operano su conoscenze locali e abbiano un determinato comportamento per poter affrontare casi di studio sui vari tipi di conflitti possibili e trovare una soluzione ottimale al problema sotto esame.

2.2 Razionalizzazione degli agenti

La prima assunzione che dobbiamo fare è che la preferenza degli agenti nel gioco è determinata dalla loro funzione di utilità "u".

Definizione 2.2 (Funzione d'utilità)

La funzione d'utilità, associata ad ogni giocatore, genera un numero reale sulla base delle preferenze che l'agente coinvolto ha per i vari stati possibili presenti nel gioco in questione. Più è grande questo numero e più ovviamente l'agente si trova in uno stato a lui congeniale.

$$u(i): S \rightarrow R$$

Definizione 2.3 (Stati di gioco)

Per l'insieme S degli stati del gioco si intende l'insieme degli stati che il giocatore i-esimo riesce a percepire. La funzione di utilità è la base su cui un agente agisce, tenendo conto della strategia adottata, al fine di massimizzarne il suo valore quando si esegue una determinata azione.

Non bisogna confondere, però la funzione di utilità, con la vincita o la perdita che il giocatore otterrà alla fine del gioco. Sono indipendenti l'una all'altra.

Una volta ottenuta la funziona di utilità è necessario determinare come l'agente usa questa funzione nel gioco per scegliere una determinata azione piuttosto che un'altra. Per fare ciò è stata introdotta una funzione di transizione T che ritorna la probabilità di raggiungere lo stato voluto dal giocatore eseguendo una determinata azione dallo stato in cui il giocatore si trova.

Definizione 2.4 (Utilità attesa)

La funzione di transizione, in concomitanza con la funzione di utilità precedentemente descritta, permette al giocatore di calcolare quella che viene definita utilità attesa definita come:

$$\mathrm{E}[u_i, \mathrm{s}, \mathrm{a}] = \sum_{s' \,\epsilon\, S} \mathrm{T}(\mathrm{s}, \mathrm{a}, \mathrm{s}') * \mathrm{u}_{\mathrm{s}'}$$

La funzione di utilità attesa consente al giocatore di avere un informazione aggiuntiva utilizzabile per poter scegliere una determinata azione tra quelle possibili in quel momento. Ricordiamo che:

- u_i è la funzione di utilità del giocatore i-esimo;
- "s" è lo stato in cui si trova il giocatore;
- "a" è l'azione che il giocatore può scegliere di eseguire;
- "s' " è lo stato che si raggiunge se si esegue l'azione "a"
- T(s, a, s') è la funzione di transizione per passare dallo stato s a s' eseguendo l'azione a
- $u_{s'}$ è la funzione di utilità nello stato s'

Quello che vogliamo affrontare, quindi, riguarda lo studio di un comportamento ottimo per il giocatore i-esimo in modo che esso possa attraverso la suddetta strategia massimizzare ad ogni azione la sua funzione di utilità attesa.

E' possibile inoltre, invece di avere una funzione di transizione, dare una pianificazione al giocatore attraverso degli operatori, che specificano quando possono essere usati e l'effetto che si avrà passando al nuovo stato. Il problema sostanzialmente risiede nel trovare una sequenza di questi operatori per andare al'obiettivo dalla scelta di partenza. Esistono numerosi algoritmi possibili utilizzabili nella pianificazione e tra i più moderni ricordiamo quelli che danno un interpretazione grafica del problema.

2.3 Giochi in forma normale

In questa sezione tratteremo dettagliatamente i giochi non-cooperativi in particolar modo vedremo come essi possono essere formalizzati in giochi a forma normale o strategica e in giochi a forma estesa. Per primo discuteremo sui giochi a forma normale e di tutte le strategie e le particolarità connessa a questo tipo di formalizzazione per poi passare ad argomentare dettagliatamente i giochi in forma estesa.

Abbiamo visto che sotto determinate assunzioni tutti i giocatori hanno una funzione di utilità e il loro scopo è raggiungere quegli stati del mondo dove questa funzione è massima. Le cose, però si complicano quando all'interno del gioco ci sono due o più agenti che vogliono massimizzare ognuno la propria funzione di utilità ed è proprio in casi simili che interviene la teoria dei giochi per arrivare ad una soluzione del gioco. I giochi più semplici possibili tra le varie tipologie sono quelli a due giocatori in cui ogni agente prende le decisioni simultaneamente all'altro. Un gioco di questo tipo può essere rappresentato da una matrice di playoff dove vengono mostrate le utilità che i giocatori assumono se eseguono quella determinata azione. Mostriamo un breve esempio di gioco in forma normale:

		Bob	
		a	b
Alice	c	1,2	3,4
	d	3,2	2,4

Figura 1: esempio di gioco in forma normale

Per fare un esempio in questo semplice gioco se Alice esegue l'azione c e Bob esegue l'azione a allora Alice otterrà un utilità di 1 e Bob un utilità di 2 sempre sotto l'ipotesi che le azioni sono eseguite simultaneamente. Inoltre nei giochi a forma normale si prende spesso l'ipotesi di

conoscenza comune, cioè ogni giocatore conosce ogni cosa e ogni giocatore sa che gli altri giocatori conoscono ogni cosa.

Definizione 2.5 (Giochi in forma normale)

Possiamo definire un gioco in forma normale come una tupla (N, A, u), dove:

- N rappresenta un set di n giocatori, a cui e possibile riferirsi tramite un indice corrispondente;
- A = A_1 x … x A_n dove A_i è un set di azioni finito disponibili per il giocatore i-esimo. Ogni vettore a = (a_1, a_2, …, a_n) è definito come profilo d'azione del giocatore i-esimo;
- u = (u_1, u_2,…, u_n) dove u_i: A → R è il valore di utilità reale per il giocatore i-esimo.

Esistono differenti sotto categorie di giochi in forma normale e possiamo raggruppare queste tipologie in giochi a compenso comune e giochi a somma zero.

2.3.1 Giochi a compenso comune

Definizione 2.6 (Gioco a compenso comune)

I giochi a compenso comune sono quelli in cui, per ogni profilo d'azione, tutti i giocatori ricevono lo stesso compenso, quindi possiamo definire che per ogni a ϵ A_1x … x A_n e considerando l'esempio di un gioco a due giocatori l'utilità del primo giocatore sarà uguale all'utilità del secondo giocatore, ovvero u_1 (a) = u_2 (a).

Questa tipologia di giochi è chiamata anche giochi a coordinazione pura o giochi di squadra e sono quei giochi dove gli agenti che partecipano non hanno conflitti fra loro ma in cui si impegnano a coordinare le loro azioni per trarre il massimo beneficio per tutti. Spesso per questi tipi di giochi si usa una particolare rappresentazione matriciale in cui in ogni cella della matrice viene trascritto solo un numero per rappresentare l'utilità in quanto comune a ogni giocatore.

Esempio:

	Left	Right
Left	1,1	0,0
Right	0,0	1,1

Figura 2: esempio di gioco a compenso comune

Questo è un esempio di un semplice gioco a compenso comune dove i due giocatori devono coordinarsi per arrivare ad una soluzione congeniale ad entrambi. Nell'esempio in questione i due equilibri raggiungibili sono: (Left, Left) e (Right, Right).

2.3.2 Giochi non cooperativi

In tutti i sistemi multi agente si hanno un set di agenti autonomi che prendono delle decisioni sulla base delle informazioni che loro stessi hanno a disposizione. I soggetti in questione dovranno anche tener conto delle azioni che eseguiranno gli altri giocatori perché influiranno anch'esse sulle possibili azioni poi selezionabili dagli stessi e quindi sul particolare "premio" finale che si otterrà. Bisogna considerare che nei sistemi multi-agente i giocatori sono interessati a se stessi, il che significa che ogni agente ha la propria descrizione degli stati del mondo che vuole raggiungere e opererà determinate azioni con lo scopo di arrivare proprio in uno di questi stati. Il giocatore quindi prenderà le sue decisioni sempre tenendo conto della sua funzione di utilità precedentemente argomentata e di vari assiomi legati ad essa quali ad esempio la completezza, la transitività, la monotonicità e la continuità.

2.3.3 Giochi a somma zero

Definizione 2.7 (Gioco a somma zero)

I giochi a somma zero sono chiamati più propriamente giochi a somma costante e possiamo definirli come quei giochi in cui se esiste una costante c per ogni profilo d'azione a ∈ A_1 x A_2 è il caso in cui c = u_1 (a) + u_2 (a).

Per convenienza si usa impostare il valore di c = 0 ed è per questo che questo che si tendono a chiamare con il nome di giochi a somma zero. Se i giochi a compenso comune sono giochi a coordinazione pura, i giochi a somma zero sono quei giochi a competizione pura, in cui ogni agente deve tener conto delle azioni degli altri agenti per cercare di massimizzare il proprio profitto. Questa tipologia di giochi ha un importanza particolare quando a competere sono solo due giocatori perché se si permettono più di due agenti in competizione, il gioco può essere reso ancora a somma zero con l'introduzione di un "falso" giocatore le cui azioni non impattano sul compenso degli agenti e in cui ogni compenso è scelto in modo tale che la somma dei profitti in ogni nodo foglia sia uguale a zero. Qui mostriamo un esempio di gioco a somma zero.

Esempio:

	Sasso	Carta	Forbice
Sasso	0,0	-1,1	1,-1
Carta	1,-1	0,0	-1,1
Forbice	-1,1	1,-1	0,0

Figura 3: esempio di gioco a somma zero della morra cinese

In questo esempio del celeberrimo gioco della morra cinese possiamo notare come la somma dei corrispondenti premi in ogni singola cella sia zero e come in ogni possibile risultato è previsto la vittoria di uno dei due giocatori o al più un pareggio nel caso in cui entrambi scegliessero la stessa figura

Esistono diverse strategie in cui un agente può operare in giochi a somma zero, la più semplice della quale è quella di selezionare un' azione tra quelle disponibili e giocarla. In questo caso possiamo parlare di strategia pura e possiamo chiamare una scelta di strategia pura per ogni agente come profilo a strategia pura. Un giocatore può utilizzare anche una strategia meno ovvia, ovvero quello di randomizzare la scelta dell'azione da giocare secondo una distribuzione di probabilità. Questo tipo di strategia è definita strategia mista.

Definizione 2.8 (Strategia mista)

Consideriamo una tupla (N, A, u) di un gioco in forma normale come un set X, di strategie pure, tale che Π(X) è il set di tutte le distribuzioni di probabilità su X. Quindi il set di una strategia mista per il giocatore i-esimo è S_i= Π(A_i).

Il set dei vari profili di strategia mista, per il giocatore i-esimo, è semplicemente il prodotto cartesiano dei set delle strategie miste individuali ovvero S_1x … x S_n. Se denotiamo come s_i (a_i) la probabilità di eseguire l'azione a_i sotto la strategia mista s_i possiamo definire come supporto della strategia mista s_i del giocatore i-esimo il set di strategie pure {a_i | s_i (a_i) > 0} e come possiamo ben notare, la strategia pura di un giocatore i-esimo è un caso speciale di una strategia mista dove il supporto è una singola azione. La strategia mista si basa moltissimo sulla definizione della teoria di giochi di utilità attesa, precedentemente spiegata. In pratica prima si calcola la probabilità di raggiungere un determinato risultato dando il profilo di strategia e poi si calcola il compenso medio degli esiti. Ora mostriamo il tipico esempio della battaglia dei sessi dove è utile l'uso di una strategia mista.

Esempio:

		Marito	
		LW	WL
Moglie	LW	2,1	0,0
	WL	0,0	1,2

Figura 4: Esempio di un gioco con utilizzo di strategie miste

In questo tipo di gioco marito e moglie devono scegliere fra le due scelte per la visione di un film LW(“Arma Letale”) e WL(“Amore Meraviglioso”). Ovviamente questo è un tipo di gioco particolare in cui la moglie preferisce guardare il film WL mentre il marito il film LW. Se entrambi si mettono comunque d’accordo potranno raggiungere comunque un utilità positiva invece che essere in disaccordo ed ottenere utilità zero. E’ un gioco tipico dove l’uso delle strategie miste, basate sulla funzione di utilità attesa, può portare al raggiungimento di una stato in cui la funzione di utilità attesa per entrambi è ≥ 0, utilità irraggiungibile in caso di disaccordo tra i due.

2.4 Calcolo di equilibri e strategie

Ora che si è descritto la tipologia di giochi in forma normale possiamo considerare il metodo di ragionamento degli agenti nei singoli giochi in cui partecipano, partendo dalla definizione di strategia ottima ovvero quella strategia che permette al giocatore di massimizzare il suo compenso per un dato ambiente in cui egli stesso deve operare. I teorici hanno quindi individuato diverse soluzioni concettuali sul problema in questione, delle quali considereremo le due maggiormente importanti che prendono il nome di ottimalità di Pareto e di ricerca di un equilibrio di Nash.

Per ottimalità di una soluzione intendiamo la ricerca di un esito di gioco che sia migliore rispetto ad un altro esito, valutando l’utilità di ogni singolo agente e quindi considerandoci come un osservatore esterno al gioco stesso. Spesso non è possibile trovare la miglior soluzione possibile ma possiamo invece ricavare e capire che una soluzione è migliore di un'altra.

Definizione 2.9 (Dominanza di Pareto)

Un profilo di strategia di Pareto s domina il profilo di strategia s' se per ogni i ∈ N, u_i (s) >= u_i (s'), ed esistono alcuni j ∈ N tale percui u_j (s) > u_j (s').

Questa dominanza permette quindi di ordinare parzialmente i vari profili di strategia. Una profilo di strategia di Pareto s è ottimo se non esiste alcun altro profilo di strategia s' ∈ S che domina s. Questo ci garantisce che in ogni gioco ci sono almeno alcuni ottimi in cui ogni giocatore può adottare una strategia pura e che ogni gioco può avere ottimi multipli.

Se passiamo ad osservare il sistema dal punto di vista dell'agente stesso invece che da un osservatore esterno come nella ricerca dell'ottimalità di Pareto introduciamo il concetto più influente della teoria dei giochi ovvero la ricerca dell'equilibrio di Nash.

Definizione 2.10 (Risposta migliore)

Per prima cosa definiamo come la risposta migliore, del giocatore i-esimo al profilo di strategia s_{-i} , la strategia mista s_i* ∈ S_i tale che u_i (s_i*, s_{-i}) >= u_i (s_i, s_{-i})per tutte le strategie s_i ∈ S_i.

La risposta migliore non è necessariamente unica e nel caso questa risposta fosse davvero unica sappiamo che essa è sicuramente una strategia pura, perché solitamente il numero di risposte migliori è infinito. Un giocatore, comunque, non conosce le strategie adottate dagli altri agenti quindi l'idea di risposta migliore non può essere una vera e propria soluzione concettuale al gioco ma permette di introdurre il concetto di equilibrio di Nash.

Definizione 2.11 (Equilibrio di Nash)

Considerando un profilo di strategia s = (s_1 , …, s_n), esso è un equilibrio di Nash per tutti gli agenti i-esimi, se s_i è la risposta migliore a s_{-i}.

Possiamo concepire l'equilibrio di Nash come un profilo di strategia stabile dove ogni agente non vuole cambiare la propria strategia se si conoscono le strategie adottate dagli altri agenti. Possiamo scendere più nel dettaglio e definire un equilibrio di Nash stretto.

Definizione 2.12 (Equilibrio di Nash stretto)

Se consideriamo un profilo di strategia in cui per tutti gli agenti i-esimi e per tutte le strategie, $s_i' \neq s_i$, $u_i(s_i, s_{-i}) > u_i(s_i', s_{-i})$ e un equilibrio di Nash debole se per ogni agente i-esimo e per tutte le strategie, $s_i' \neq s_i$, $u_i (s_i, s_{-i}) >= u_i (s_i', s_{-i})$ e non è un equilibrio di Nash stretto.

2.4.1 Strategia maxmin

Oltre al concetto di equilibrio di Nash esistono altri concetti di soluzioni che sono state studiate dai teorici, alcune delle quali sono decisamente più restrittive della ricerca dell'equilibrio di Nash. Tra le più importanti studieremo le strategie maxmin e minmax spesso usate e implementate per la risoluzione di giochi a più agenti.

La strategia maxmin di un giocatore i-esimo in un gioco a N giocatori a somma generale è una strategia che massimizza il caso di compenso peggiore del giocatore i-esimo nella situazione in cui tutti gli altri giocatori giochino una strategia atta a dare il più grande danno possibile al giocatore i-esimo. Il valore maxmin del gioco per il giocatore i-esimo è l'ammontare minimo di compenso garantito dalla strategia maxmin.

Definizione 2.13 (Strategia maxmin)

Possiamo definire la strategia maxmin per il giocatore i-esimo come l'argmax(s_i) min(s_{-i}) u_i (s_i, s_{-i}) e come valore di maxmin il max(s_i) min(s_{-i}) u_i (s_i, s_{-i}).

Questo tipo di strategia rappresenta la scelta strategica migliore quando in primis il giocatore i-esimo deve adottare una strategia e tutti gli altri agenti osservano questa strategia e scelgono, poi, come propria strategia quella di minimizzare il compenso atteso del giocatore i-esimo. La strategia maxmin è una scelta conservativa per tutti gli agenti che l'adottano perché il loro scopo è solo quello di massimizzare la propria utilità attesa senza fare alcuna assunzione delle strategie adottate dagli altri agenti.

2.4.2 Strategia minmax

La strategia minmax invece svolge l'azione duale della strategia maxmin ed è quella strategia in cui il giocatore i-esimo vuole minimizzare il massimo compenso del giocatore avversario e il valore di minmax dell'avversario è proprio questo minimo.

Definizione 2.14 (Strategia minmax)

Possiamo definire la strategia minmax per il giocatore i-esimo come l'argmin(s_i) min(s_{-i}) u_i (s_i, s_{-i}) e come valore minmax il min((s_i) min(s_{-i}) u_i (s_i, s_{-i}).

Questo tipo di strategia è verosimilmente adottata quasi sempre per giochi in cui a confrontarsi sono due agenti perché la formalizzazione per un numero di giocatori superiore a due è più complicata. Infatti in un gioco a N giocatori la strategia minmax per il giocatore i-esimo contro i giocatori j-esimi è la componente i-esima del profilo di strategia mista s(-j) nell'espressione argmin(s_j) min(s_{-j}) u_j (s_j, s_{-j}) dove con (-j) si identificano tutti gli altri giocatori ad esclusione del j-esimo e come prima possiamo definire il valore minmax come min(s_{-j}) max(s_j) u_j (s_j, s_{-j}). In

un gioco a due giocatori, comunque, il valore minmax è sempre uguale al suo valore maxmin mentre per giocatori con più di due giocatori la condizione è più debole in quanto il valore maxmin di un giocatore e spesso inferiore o al massimo uguale al suo valore minmax. In conclusione abbiamo la possibilità di affermare che in tutti i giochi a somma zero a due giocatori, in ogni equilibrio di Nash ogni giocatore riceve un compenso che è uguale sia al suo valore maxmin che al suo valore minmax.

Lo studio di tutte queste soluzioni concettuali porta in definitiva a delle conclusioni per i giochi a due giocatori a somma zero:

- Il valore maxmin di ogni giocatore è uguale al suo valore minmax e per convenzione il valore maxmin del primo giocatore è detto valore del gioco;

- Per entrambi i giocatori, il set di strategie maxmin coincide con il set di strategie minmax;

- Ogni profilo di strategia maxmin è un equilibrio di Nash e in più possiamo dire che tutti questi profili sono equilibri di Nash pertanto tutti gli equilibri di Nash hanno lo stesso vettore dei compensi.

In molti casi comunque la strategia degli altri agenti non è sempre preventivabile e spesso non è legata alla minimizzazione del nostro guadagno bensì alla massimizzazione del loro. Spesso per un agente risulta meglio preoccuparsi di minimizzare la perdita dal compenso massimo piuttosto che massimizzare il proprio caso peggiore. In questo caso il "regret" per il giocatore i-esimo di giocare l'azione a_i se gli altri giocatori adottano un profilo d'azione a_{-i} è definito per tutte le azioni $a_i' \in A_i$ come $[\max u_i (a_i', a_{-i})] - u_i (a_i, a_{-i})$ che in sostanza rappresenta l'ammontare che il giocatore i-esimo perde giocando l'azione i-esima piuttosto che giocando la sua azione migliore. Possiamo definire come massimo "regret" l'ammontare che il giocatore i-esimo perde giocando l'azione i-esima piuttosto che giocare la sua azione migliore, se gli altri giocatori scelgono di giocare l'azione tale da rendere questa perdita il più grande possibile. Un altro modo per i giocatori di approcciarsi alla scelta di un azione da giocare è quella prima di tutto di valutare se esistono azioni che possono essere giocate senza che il guadagno dello stesso giocatore sia inferiore a quello di una sua qualsiasi altra azione, per ogni risposta possibile da parte degli avversari. Quindi se consideriamo s_i e s_i' come due strategie del giocatore i-esimo e S_{-i} come il set di tutti i profili di strategia dei giocatori rimanenti possiamo affermare che la strategia s_i domina la strategia s_i' se per tutte i profili $s_{-i} \in S_{-i}$, siamo nel caso in cui

$u_i\ (s_i, s_{-i}) > u_i\ (s_i', s_{-i})$. Facciamo un breve esempio per meglio capire il funzionamento del "regret":

	L	R
T	100, a	1-ϵ, b
B	2,c	1,d

Figura 5: esempio di gioco per verificare funzionamento del "regret"

Nell'esempio in questione consideriamo ϵ una quantità positiva e molto piccola e ipotizziamo i costi del secondo agente come a,b,c,d come se il primo giocatore non li conoscesse. L'agente uno ovviamente deve supporre che il secondo agente sia razionale e quindi potrebbe giocare una strategia maxmin o comunque una strategia conservativa per salvaguardarsi. Se così facesse si può notare facilmente che l'agente uno giocherebbe in questo caso l'azione B perché se il giocatore due adottasse una strategia minmax sceglierebbe sempre l'azione R. Ovviamente però l'agente uno potrebbe sempre pensare che l'agente due non sia malizioso e che quindi giochi per massimizzare la sua utilità piuttosto che minimizzare la sua utilità. In questo caso l'agente uno giocherebbe sicuramente l'azione T perché in ogni caso se il giocatore due giocasse R, rispetto all'azione B il primo agente avrebbe solo una perdita di ϵ molto piccola ma se il giocatore due giocasse L, allora il primo agente avrebbe un utilità di 100 e non una perdita di 98 come sarebbe giocando l'azione B.

Possiamo affermare alla luce di questo che una strategia è strettamente dominante per un agente se domina ogni altra strategia per quell'agente e che ogni strategia dominante per il giocatore i-esimo è un equilibrio di Nash del gioco. Inoltre se inizialmente possiamo eliminare una strategia rispetto ad un'altra perché dominata da quest'ultimo dobbiamo poi rivalutare su tutte le azioni rimaste se si sono create altre dominanze possibili e questo processo di eliminazione può continuare fino anche ad arrivare ad un unico risultato possibile. Va considerato che anche l'ordine di eliminazione di strategie dominate può influire sul risultato finale del gioco.

In definitiva possiamo concludere, che analizzando tutte le più importanti strategie adottabili dagli agenti in giochi in forma normale, spesso i giocatori preferiscono utilizzare determinate strategie piuttosto che giocare la propria azione migliore in quanto l'ammontare dell'utilità che possono ottenere così facendo è molto piccola.

2.4.3 Equilibrio ϵ-Nash

Definizione 2.15 (ϵ-Nash equilibrio)

Possiamo formalizzare il tutto con il concetto di ϵ-Nash equilibrio in cui fissato un ϵ > 0, un profilo di strategia s = (s_1, . . ., s_n) è un ϵ-Nash equilibrio se per tutti gli agenti i-esimi e per tutte le strategie s_i' ≠ s_i, l'utilità u_i(s_i, s_{-i}) >= u_i(s_i', s_{-i}) - ϵ.

Un ϵ-Nash equilibrio esiste sempre e può essere utilizzato per dimostrare che ogni equilibrio di Nash è circondato da una regione di ϵ-Nash equilibri per ogni ϵ > 0. Un ϵ-Nash equilibrio ha anche degli svantaggi in quanto è vero che ogni equilibrio di Nash è circondato da ϵ-Nash equilibri ma non è vero il contrario ed inoltre essi non necessariamente si trovano per forza vicini ad un vero equilibrio di Nash.

Solitamente le classi di giochi a due giocatori a somma zero sono le più semplici da risolvere perché il problema della ricerca dell'equilibrio di Nash può essere espresso, in questi casi, come un problema di programmazione lineare, la cui computazione per la ricerca dell'equilibrio ha un tempo di risoluzione polinomiale. Se facciamo un esempio di un gioco di questo tipo in cui il giocatore due adotta una strategia minmax sul giocatore avversario possiamo formalizzare il problema come segue:

$$\text{minimize } U_1$$

$$\text{subject to } \sum_{k \epsilon A2} u_1\left(\mathrm{a}_1^{\mathrm{j}}, \mathrm{a}_2^{\mathrm{k}}\right) * \mathrm{s}_2^{\mathrm{k}} \leq U_1$$

$$\sum_{k \epsilon A2} \mathrm{s}_2^{\mathrm{k}} = 1$$

$$\mathrm{s}_2^{\mathrm{k}} \geq 0$$

E' possibile notare come in un problema di questo tipo tutti i termini u_1 (.) siano costanti perché si tratta di un problema di programmazione lineare mentre i termini s_2 e U_1 sono variabili. Il giocatore 2, nel gioco in questione, tenta di minimizzare l'utilità dell'avversario.

2.5 Giochi in forma estesa

La rappresentazione dei giochi in forma normale non tiene conto di nessuna nozione per quanto riguarda la sequenza o il tempo delle azioni intraprese da ciascun giocatore. I giochi in forma estesa, invece, permettono di avere una rappresentazione alternativa che rende la struttura temporale del tempo e delle sequenze di azioni intraprese esplicita. Con questo tipo di

rappresentazione ha molto senso la suddivisione dei giochi in giochi a informazione perfetta e giochi ad informazione imperfetta. Per prima discuteremo il primo tipo tra essi.

I giochi in forma estesa a informazione perfetta sono un albero nel senso della teoria dei grafi, in cui ogni nodo rappresenta la scelta che un giocatore può aver fatto per arrivare fino a quel punto e ogni arco che dal nodo si estende rappresenta una possibile azione che il giocatore può intraprendere. I nodi finali da cui non si diramano più archi vengono chiamati nodi foglia e rappresentano la fine del gioco dove ogni giocatore ottiene la sua utilità. Questo tipo di rappresentazione spesso è semplicemente conosciuta come albero di gioco. Di seguito mostriamo un semplice esempio:

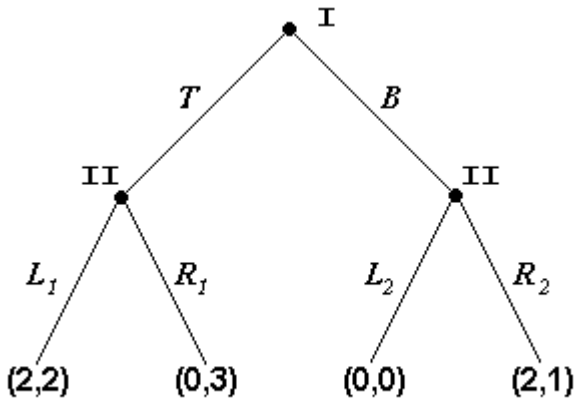


Figura 6: esempio di gioco in forma estesa(albero di gioco)

In questo semplice gioco il giocatore uno può scegliere tra le azioni T e B e in base alla scelta del giocatore uno e quindi al nodo decisionale a cui si arriva il giocatore due potrà scegliere rispettivamente tra le azioni (L_1 o R_1) oppure (L_2 o R_2) arrivando quindi al nodo foglia del gioco. Nel seguito del capitolo affronteremo il problema di come i giocatori sceglieranno un azione piuttosto che un'altra ai propri nodi decisionali.

Definizione 2.16 (Gioco in forma estesa ad informazione perfetta)

Quindi un gioco a informazione perfetta è una tupla G = (N, A, H, Z, χ, ρ, σ, u) dove:

- N è il set di n giocatori;
- A è il singolo set di azioni;
- H è il set di nodi di scelta non terminali;
- Z è il set di nodi terminali, non compresi in H;

- χ : H → 2^A è la funzione d'azione che associa ad ogni nodo di scelta il set delle possibili azioni;

- ρ : H → N è la funzione giocatore, che assegna ad ogni nodo non-terminale un giocatore i-esimo tra il set di N giocatori che sceglie l'azione da intraprendere a quel nodo;

- σ : H x A → H ∪ Z è la funzione successore, che mappa un nodo di scelta e un azione ad un nuovo nodo di scelta o ad un nodo terminale tale per cui per ogni h_1, h_2 ∈ H e a_1, a_2 ∈ A, se σ(h_1, a_1) = σ(h_2, a_2) allora $h_1 = h_2$ e $a_1 = a_2$;

- u = (u_1, . . ., u_n) dove u_i : Z → R è la funzione di utilità a valori reali per il giocatore i-esimo sul nodo terminale Z.

Come per i giochi in forma normale possiamo definire le strategie pure e miste anche per un gioco in forma estesa.

Definizione 2.17 (Strategia pura per gioco in forma estesa)

Se consideriamo un qualsiasi gioco ad informazione perfetta in forma estesa
G = (N, A, H, Z, χ, ρ, σ, u) allora una strategia pura del giocatore i-esimo consiste nella produttoria per tutti gli h ∈ H e per ρ(h) = i della funzione d'azione χ(h).

Possiamo quindi affermare che la strategia di un agente richiede una decisione ad ogni nodo di scelta che sia possibile o no raggiungere quel nodo in base alla scelta degli altri nodi. E' facile notare che per i giochi a forma estesa non cambiano le definizione ne per quanto riguardano l'azione migliore che un giocatore i-esimo può fare, ne per quanto riguarda la definizione di equilibrio di Nash e ciò avviene perché è sempre possibile passare da un gioco in forma estesa, quindi con rappresentazione ad albero ad un gioco in forma normale quindi con rappresentazione matriciale. Va considerato comunque che questo tipo di passaggio comporta spesso un aumento esponenziale della rappresentazione del gioco che non porta ad alcun tipo di beneficio nella risoluzione. Alla luce di questa corrispondenza possiamo altresì affermare che un gioco finito ad informazione perfetta possiede sempre un equilibrio di Nash raggiungibile attraverso una strategia pura e questo perché le azioni rispetto ad un gioco in forma normale sono prese a turni quindi ogni giocatore ha la capacità di vedere l'evoluzione del gioco avvenuto fino a quel momento senza dover inserire una funzione di probabilità casuale per la scelta della sua successiva azione. Alla fine in base alla scelta di ogni giocatore che gioca per massimizzare la propria utilità si arriverà ad un nodo foglia che rappresenta un equilibrio del gioco. Bisogna tener conto del fatto che in questi giochi non sempre si raggiunge un vero e proprio equilibrio di Nash ma molto spesso si arriva ad un equilibrio perfetto di sotto-gioco(SPE) in cui per sotto-gioco si intende una porzione a sé dell'albero dell'intero gioco che può essere estratta in base alle scelte che i giocatori hanno scelto fino a quel momento.

Definizione 2.18 (equilibrio perfetto di sotto-gioco SPE)

Possiamo definire un equilibrio perfetto di un sotto-gioco come tutti i profili di strategia s tali che per ogni sotto-gioco G' di G, la restrizione di s a G' è un equilibrio di Nash per G'.

Questo concetto è più forte che quello di equilibrio di Nash perché un SPE è un equilibrio di Nash ma va considerato che in un gioco completo a informazione perfetta in forma estesa possono esistere più di uno di questi equilibri e quindi non si garantisce l'unicità della soluzione. In giochi a forma estesa a somma zero in cui in campo competono solo due giocatori è possibile utilizzare sotto opportune modifiche le strategie minmax e maxmin viste per i giochi in forma normale. Ciò di cui bisogna tener conto in questo caso è che l'albero di gioco può essere ridotto perché è possibile già escludere in partenza rami in cui non si giungerà mai ad un equilibrio, inoltre nella valutazione di tutti i rami presi in esame si parte dalla valutazione dei nodi fogli per poi tornare indietro man mano e valutare le conoscenze di entrambi i giocatori su ogni nodo. Si tiene conto ovviamente sempre della massima utilità ottenibile in ogni sotto-gioco considerando anche il fatto che l'opponente ha la possibilità di giocare o no un azione diversa in base ai risultati del sotto-gioco fin lì ottenuti. Il vantaggio di questo tipo di algoritmo rispetto ad altri è che spesso non si valutano tutti i nodi presenti sotto una certa profondità h ma solo alcuni di essi rendendo quindi più veloce la risoluzione e l'arrivo ad un equilibrio del gioco stesso. Mettiamo un esempio per mostrare come un SPE è anche un equilibrio di Nash e di quanto sia importante esso nei giochi in forma estesa.

Esempio:

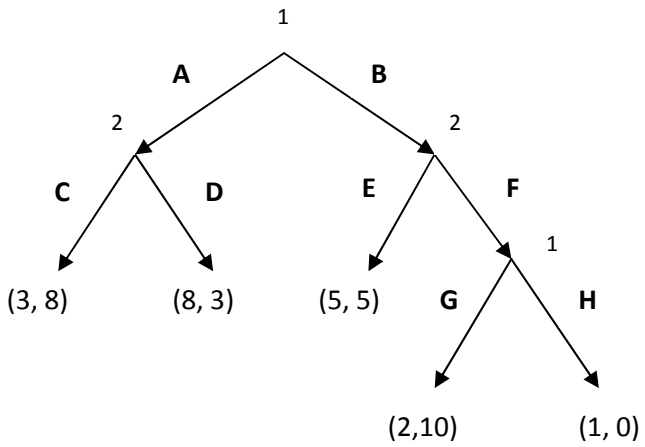


Figura 7: albero di gioco della forma strategica

Forma normale:

	(C, E)	(C, F)	(D, E)	(D, F)
(A, G)	3, 8	**3, 8**	8, 3	8, 3
(A, H)	3, 8	**3, 8**	8, 3	8, 3
(B, G)	5, 5	2, 10	5, 5	2, 10
(B, H)	**5, 5**	1, 0	5, 5	1, 0

Figura 8: rappresentazione in forma normale di un gioco a forma estesa

Se esaminiamo la forma normale di questo gioco a forma estesa (ricordiamo che è sempre possibile passare da un gioco in forma estesa alla sua rappresentazione in forma normale), notiamo che esistono tre equilibri di Nash possibili (evidenziati in rosso) che sono le azioni { (A, G), (C, F) }, { (B, H), (C, E) } e { (A, H), (C, F) }. Il primo e il terzo equilibrio sono abbastanza intuitivi ed immediati da trovare mentre il secondo lo è un po' meno. Esso lo è perché il giocatore due sa che il primo giocatore giocherebbe sempre l'azione H al suo secondo nodo di decisione(se nel primo è giocata l'azione B) in quanto è sicuro che otterrebbe sempre la stessa utilità ma darebbe un danno al giocatore due che è costretto a scegliere l'azione E per evitare a sua volta perdite di utilità. Tutta via un problema si porrebbe se il secondo giocatore gioca comunque F e il primo giocatore sa di questa scelta. Che farebbe quindi il primo giocatore? Giocherebbe ancora H e quindi perde una utilità ma contemporaneamente causa danno al secondo giocatore oppure gioca G per massimizzare la sua utilità a discapito dell'utilità del secondo giocatore?

Se esaminiamo la forma estesa ,invece, notiamo che partendo dai sotto-alberi più vicini ai nodi foglia e considerando sempre l'utilità migliore per il giocatore corrispondente al nodo otteniamo alla fine un unico e solo equilibrio che è: {(A, G), (C, F)}.

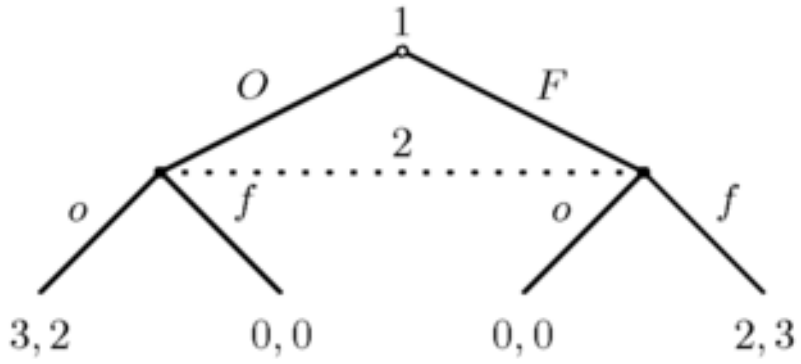
Arrivati a questo punto possiamo affermare che le azioni di un giocatore potrebbero in qualche punto dell'albero essere inosservabili (ad esempio quando si pesca una carta che non viene scoperta a tutti) e quindi non abbiamo disposizione sempre tutte le informazioni possibili. Nasce quindi l'esigenza di presentare i cosiddetti giochi a forma estesa ad informazione imperfetta. In questo tipo di giochi ogni nodo di scelta di ogni giocatore è partizionato in set di informazioni; solitamente se due nodi sono nello stesso set di informazione allora il giocatore non può distinguere in quale dei due si trova.

Definizione 2.19 (Gioco in forma estesa ad informazione imperfetta)

Un gioco a informazione imperfetta è una tupla G = (N, A, H, Z, χ, ρ, σ, u, I) dove:

- N è il set di n giocatori;
- A è il singolo set di azioni;
- H è il set di nodi di scelta non terminali;
- Z è il set di nodi terminali, non compresi in H;
- χ : H → 2^A è la funzione d'azione che associa ad ogni nodo di scelta il set delle possibili azioni;
- ρ : H → N è la funzione giocatore, che assegna ad ogni nodo non-terminale un giocatore i-esimo tra il set di N giocatori che sceglie l'azione da intraprendere a quel nodo;
- σ : H x A → H ∪ Z è la funzione successore, che mappa un nodo di scelta e un azione ad un nuovo nodo di scelta o ad un nodo terminale tale per cui per ogni h_1, h_2 ∈ H e a_1, a_2 ∈ A, se σ(h_1, a_1) = σ(h_2, a_2) allora $h_1 = h_2$ e $a_1 = a_2$;
- u = (u_1, . . ., u_n) dove u_i : Z → R è la funzione di utilità a valori reali per il giocatore i-esimo sul nodo terminale Z.
- I = (I_1, . . ., I_n), dove I_i = ($I_{i,1}$, . . ., I_{i,k_i}) è una relazione di equivalenza per ogni {h ∈ H: ρ(h) = i} con la proprietà che χ(h) = χ(h') e ρ(h) = ρ(h'), ogni qualvolta esiste un j tale per cui h ∈ $I_{i,j}$ e h' ∈ $I_{i,j}$.

Un esempio di gioco ad informazione imperfetta in forma estesa:



In questo tipo di gioco in forma estesa e ad informazione imperfetta possiamo notare come il giocatore due si trovi in un set di informazioni dopo la scelta dell'azione del primo giocatore(O oppure F), ovvero esso non sa in quale dei due nodi si trova e deve trovare una strategia adeguata per scegliere un azione tra quelle disponibili a sua scelta(o oppure f).

Definizione 2.20(Strategia pura per gioco in forma estesa ad informazione imperfetta)

Possiamo definire anche per questo tipo di gioco G = (N, A, H, Z, χ, ρ, σ, u, I) la strategia pura come il prodotto per tutti gli $I_{i,j} \in I_i$ della funzione d'azione $\chi(I_{i,j})$.

Come per i giochi ad informazione perfetta è possibile passare alla corrispettiva forma normale anche per i giochi a informazione imperfetta ma in questo caso dobbiamo considerare un particolare in più: dato che ogni giocatore non conosce perfettamente tutte le informazioni relative al gioco ha senso introdurre per questo tipo di giochi il concetto di strategia mista. La strategia mista prende valore quando il giocatore i-esimo è in un set d'informazione di nodi nel quale non sa esattamente dove si trova ed è possibile definire un set di tale strategia esattamente come i set di strategie miste definite per i giochi in forma normale e quindi con la conseguente definizione di set di equilibri di Nash. In questo tipo di gioco è possibile definire anche le strategie di comportamento che sono strategie per cui la scelta di ogni agente ad ogni nodo è presa indipendentemente dalla sua scelta negli altri nodi. Tuttavia le strategie miste e le strategie di comportamento non sono comparabili perché ci sono giochi in cui si raggiungono determinati nodi foglia con le strategie miste e non con le strategie di comportamento e viceversa anche se esistono alcuni giochi per cui entrambi i tipi di strategia coincidono. Questi giochi prendono il nome di giochi a memoria perfetta.

Definizione 2.21 (Giochi a memoria perfetta)

Il giocatore i-esimo ha una memoria perfetta in un gioco G ad informazione imperfetta se per ogni due nodi h, h' che si trovano nello stesso set di informazioni per il giocatore i-esimo, per ogni ramificazione possibile $h_0, a_0, h_1, a_1, h_2, \ldots, h_n, a_n$, h dalla radice del gioco ad h e per ogni ramificazione $h_0, a_0', h_1', a_1', h_2', \ldots, h_n', a_n'$, h' dalla radice del gioco ad h' si è nel caso in cui:

- n = m;
- per ogni 0 <= j <= n, h_j e h_j' sono la stessa classe di equivalenza per il giocatore i-esimo;
- per ogni 0 <= j <= n, se ρ(h_j) = i allora $a_j = a_j'$;

G è un gioco a memoria perfetta se ogni giocatore ha memoria perfetta in esso. In questo tipo di gioco ogni strategia mista può essere sostituita da una strategia di comportamento e viceversa. Le due strategie sono equivalenti nel senso che inducono con la stessa probabilità sui nodi foglia per ogni profili di strategia fissata per gli agenti rimanenti. E' possibile concludere, quindi, che per questo tipo e solo questo tipo di giochi il set di equilibri di Nash non cambia se adottiamo le strategie di comportamento o strategie miste.

Dopo aver analizzato le caratteristiche di questo tipo di giochi, che più si adattano alle situazioni che si possono riscontrare nella realtà, dobbiamo calcolare anche per essi degli equilibri di gioco che siano la risoluzione del problema stesso. La via più ovvia per trovare quest'equilibrio sarebbe quello di convertire i giochi a forma estesa in giochi a forma normale e trovare quindi un equilibrio sulla corrispondente matrice trovata attraverso uno dei noti algoritmi di risoluzione, come ad esempio il Lemke, ma questo metodo risulta altamente inefficiente in quanto il numero di azioni in un gioco a forma normale è esponenziale rispetto a quello in forma estesa. E' stata trovata quindi una soluzione in grado di gestire al meglio la computazione dell'equilibrio di gioco e questa soluzione prende il nome di forma sequenza, utilissima per rappresentare giochi in forma estesa ad informazione imperfetta.

Definizione 2.22 (Forma sequenza)

Se consideriamo un gioco G come un gioco a informazione imperfetta ma a perfetta memoria possiamo definire la rappresentazione in forma sequenza di G come una tupla (N, Σ, g, C) dove:

- N è un set di agenti;
- Σ = (Σ_1, . . ., Σ_n), dove Σ_i è il set di sequenza disponibile per l'agente i-esimo;
- g = (g_1, . . ., g_n), dove g_i : Σ → R è la funzione ricompensa per l'agente i-esimo;
- C = (C_1, . . ., C_n), dove C_i è un set di vincoli lineari sulla realizzazione di probabilità dell'agente i-esimo.

Adesso dobbiamo considerare che il concetto di sequenza non è legato all'idea di strategia pura dell'agente i-esimo ma bensì è legato alla sequenza di nodi attraversata dall'agente dalla radice fino ad giungere ad un certo nodo h. Quindi una sequenza di azioni per il giocatore i-esimo è il set di azioni ordinate del giocatore i-esimo che partono dal nodo radice fino ad arrivare ad un dato nodo h. E' usuale definire con il simbolo ∅ la sequenza corrispondente al solo nodo radice mentre il set di sequenze del giocatore i-esimo è denotato da Σ_i e Σ = Σ_1 x . . . x Σ_n è il set di tutte le sequenze. Il nodo h considerato non deve essere esclusivamente un nodo foglia ma può essere un qualsiasi nodo a qualsiasi profondità dell'albero. Comunque se dopo una sequenza considerata si arriva alla valutazione di un nodo che è anche nodo foglia dell'albero allora è possibile introdurre il concetto di premio o ricompensa g_i per il giocatore i-esimo come $g_\sigma = u_z$, con z nodo

foglia dell'albero, quel premio raggiunto dal giocatore i-esimo se gioca la sequenza σ(i) ϵ σ. Sarà, invece, g_σ = 0 il premio, se il nodo considerato non è un nodo foglia. Un esempio di forma sequenza:

	∅	A	B
∅	0,0	0,0	0,0
L	0,0	0,0	0,0
R	1,1	0,0	0,0
Ll	0,0	0,0	2,4
Lr	0,0	2,4	0,0

sequenza di un gioco in forma estesa

L'utilizzo della forma sequenza sembra a prima vista più dispendioso di una normale matrice di un gioco in forma normale, ma se si guarda attentamente è possibile notare come molti elementi delle celle della matrice stessa siano a zero. Quando si devono memorizzare i dati della forma sequenza verranno memorizzati solo i dati di quelle celle dove il valore effettivo è effettivamente diverso da zero, almeno per uno dei due valori. Questo permette un notevole risparmio di spazio e anche una maggiore efficienza di risoluzione. Nell'esempio precedente verranno quindi memorizzati solo i dati di tre celle. Questo tipo di rappresentazione è valida se i giocatori di un gioco a forma estesa adottano delle strategie miste mentre nulla è possibile dire se i giocatori adottano, invece, delle strategie di comportamento per cui l'ottimizzazione dovuta alla forma sequenza non è funzionale e di più difficile risoluzione.

Si dovrebbe parlare, nel caso di strategie di comportamento, di piano di realizzazione, che ha caratteristiche differenti rispetto alla forma sequenza. Ne diamo qua la definizione.

Definizione 2.23 (Piano di realizzazione – 1° formulazione)

Il piano di realizzazione per un giocatore i-esimo ϵ N è la mappatura $r_i : \Sigma_i \rightarrow [0, 1]$ definito come $r_i(\sigma_i) = \prod_{c \epsilon \sigma_i} \beta_i\,(c)$. Ogni valore $r_i(\sigma_i)$ è chiamato probabilità di realizzazione.

È possibile dare un ulteriore definizione di piano di realizzazione tenendo conto di due funzioni, la prima delle quali definisce una singola sequenza che il giocatore i-esimo può giocare in un gioco a forma estesa a memoria perfetta mentre la seconda funzione considera come questa sequenza può essere costruita dalle altre sequenze. Passiamo ora alla definizione ultima di piano di realizzazione.

Definizione 2.23 (Piano di realizzazione – 2° formulazione)

Un piano di realizzazione per un giocatore i-esimo ϵ N è una funzione $r_i : \Sigma_i \rightarrow [0, 1]$ che soddisfa i seguenti vincoli:

$r_i(\varnothing) = 1$

$\sum_{\sigma_i' \epsilon Ext_i(I)} r_i(\sigma_i) = r_i(seq_i(I))$

$r_i(\sigma_i) >= 0$

Come abbiamo detto precedentemente la rappresentazione in forma sequenza rende più agevole e più efficiente il calcolo di un equilibrio per giochi a forma estesa. Possiamo formalizzare dunque il problema di un qualsiasi gioco in forma estesa a due giocatori partendo dalla seguente generalizzazione:

maximize $\sum_{\sigma_1 \epsilon \Sigma_1} \left(\sum_{\sigma_2 \epsilon \Sigma_2} g_1(\sigma_1, \sigma_2) r_2(\sigma_2) \right) r_1(\sigma_1)$

subject to $r_1(\varnothing) = 1$

$\sum_{\sigma_1' \epsilon Ext_1(I)} r_{1(\sigma(1)')} = r_1(seq_1(I))$

$r_1(\sigma_1) >= 0$

In questo esempio di formalizzazione il giocatore uno dovrebbe scegliere r_1 per massimizzare la sua funzione di utilità. Se appunto introduciamo il concetto di utilità attesa, considerando il caso particolare di un gioco a somma zero, possiamo permetterci di formalizzare la forma sequenza, a partire dalla generalizzazione precedente, come un problema di programmazione lineare il cui obiettivo è quello di trovare un equilibrio di Nash in un tempo che sia polinomiale in base alla dimensione del gioco in forma estesa. Possiamo formalizzare il tutto nel seguente modo:

minimize v_0

subject to $v_{I1(\sigma(1))} - \sum_{I \epsilon I1(Ext1(\sigma(1)))} v_{I'} \geq \sum_{\sigma(2) \epsilon \Sigma(2)} g_{1(\sigma(1),\sigma(2) r_2(\sigma(2)))}$

$r_2(\varnothing) = 1$

$$\sum_{\sigma(2)'\epsilon \mathrm{Ext2(I)}} \mathrm{r}_{2(\sigma(2)')} = \mathrm{r}_2(\mathrm{seq2(I)})$$

$r_2(\sigma(2))$ >= 0

v_0 rappresenta l'utilità attesa per il giocatore due sotto l'ipotesi del piano di realizzazione che egli scelga di giocare dato il piano di realizzazione dell'avversario e solitamente questo valore corrisponde all'utilità attesa quando il giocatore due sceglie la sua risposta migliore. Ogni altra variabile $\mathrm{v_I}$ può essere interpretata come una porzione dell'utilità attesa, che il giocatore due percepirà, giocando la sua risposta migliore nel sottogioco partendo dal set di informazione I e avendo a disposizione anche il piano di realizzazione del giocatore avversario. In pratica la formalizzazione che abbiamo precedentemente esposto rappresenta una strategia minmax che il giocatore due utilizza contro il suo avversario in un gioco a forma estesa a somma zero.

Capitolo 3

Ricerca di un equilibrio perfetto in forma estesa

In questo capitolo parleremo della ricerca di un equilibrio perfetto in forma estesa(EFPE) per giochi in forma estesa. Introdurremo quindi il concetto di gioco perturbato del quale parleremo nella sezione 3.1 del suddetto capitolo nel quale parleremo anche di altri tipi di importanti equilibri. Nella sezione 3.2, invece, tratteremo la computazione di un EFPE per giochi perturbati uniformi e nella sezione 3.3 parleremo dello stesso problema ma riguardo giochi perturbati non uniformi.

3.1 Equilibri perfetti e giochi perturbati

Nel capitolo precedente abbiamo parlato della teoria dei giochi, della tipologia di giochi in forma normale e in forma estesa e dei relativi metodi per computare un equilibrio di Nash del gioco stesso. Abbiamo però affrontato questo problema per i giochi in forma estesa in maniera molto simile a quello utilizzato per i giochi in forma normale. Solitamente infatti nelle strategie adottate si tiene sempre conto che i giocatori attuino sempre la loro risposta migliore all'azione dell'avversario. Tipicamente in un problema reale non sempre ogni agente sa quale azioni tra quelle disponibile sia sempre la migliore e pertanto e anche opportuno considerare la possibilità che un agente possa intraprendere un azione che in realtà non si rivelerà altrettanto efficace come sarebbe stata prenderne un'altra. Entra così in gioco il concetto di perturbazione, ovvero la possibilità per un giocatore di sbagliare scelta dell'azione in un nodo decisionale a vantaggio del suo avversario. Questo concetto fu introdotto già nel 1975 da Selten ed è tutt'ora ampiamente utilizzato nel campo della computazione di un equilibrio per un gioco in forma estesa. In pratica in un gioco con perturbazione si considera sempre che l'avversario abbiamo una probabilità $\varepsilon > 0$ di prendere una decisione diversa da quella che sarebbe la sua azione migliore. Questa modifica al problema di computazione standard ovviamente introduce delle modifiche riguardo la funzione di utilità attesa di ogni giocatore e di conseguenza influisce in maniera potente sulla computazione dell'equilibrio finale. In questo capitolo tuttavia si vuole andare ancora maggiormente in profondità rispetto a questa modifica. Spesso appunto come si è spiegato precedentemente si considera sempre, che solo uno dei due giocatori possa prendere una decisione diciamo errata e quindi si tende a costruire per l'avversario una strategia che tenga conto dell'handicap dell'altro giocatore. In realtà sarebbe molto più consono pensare che entrambi i due giocatori possano sbagliare e quindi prendere una decisione differente dalla loro azione migliore. Nasce così il

concetto di equilibrio perfetto in forma estesa (EFPE) che tiene conto di queste modifiche per entrambi gli agenti in gioco.

Un EFPE raffina il concetto di equilibrio di Nash e rende più robusta la sua computazione in un gioco in cui si considerano le strategie di comportamento di entrambi i giocatori affette da perturbazione. Come abbiamo trattato nel capitolo precedente un gioco in forma estesa può essere rappresentato efficacemente dalla sua forma sequenza che è esponenzialmente più piccola della sua rappresentazione in gioco in forma normale ma sappiamo anche che il concetto di equilibrio per questo tipo di giochi si discosta leggermente da quello di giochi in forma normale e spesso i normali algoritmi di risoluzione non sono a loro volta applicabili. Sono stati prodotti nel corso degli anni numerosi algoritmi per computare un equilibrio tenendo conto di vari problemi: ricerca di un equilibrio perfetto di sottogioco(SPE) quando le informazioni del gioco stesso sono perfette, equilibrio quasi perfetto(QPE) per giochi ad informazione imperfette, brevemente spiegato precedentemente e approfondito in questa sezione e infine un equilibrio perfetto in forma estesa(EFPE) di cui tratteremo in questo capitolo. Innanzitutto dobbiamo notare che un EFPE non è un raffinamento del QPE ma rispetto ad esso tiene conto delle perturbazioni di entrambi i giocatori e include nel gioco le strategie dominate debolmente ed inoltre la computazione di un EFPE apre la strada allo studio di situazioni in cui un agente sostiene i costi nel controllare la sua strategia.

Per la computazione di un EFPE terremo conto di giochi in forma estesa ad informazione imperfetta in cui si considerano strategie di comportamento, ovvero quelle strategie dove la scelta ad un set di informazione è indipendente dalle scelte sugli altri nodi.

Come prima cosa definiamo un QPE e un metodo di computazione adeguato a questo tipo di equilibrio e poi passeremo a trattare il caso dell'EFPE.

Definizione 3.1 (Equilibrio quasi perfetto QPE)

Un QPE è un profilo di strategia σ tale che esiste un profilo di strategia perturbata σ(ϵ) per cui $\lim_{\epsilon \to 0} \sigma(\epsilon) = \sigma$ per tutti gli agenti i-esimi appartenenti all'insieme N degli agenti in cui la strategia σ(i) è ottimale rispetto a σ_{-i}(i) per ogni ε ϵ [0, ϵ_0] con $\epsilon_0 > 0$.

Il QPE è un raffinamento di un equilibrio di Nash per giochi a forma estesa in cui uno dei due giocatori adotta una strategia di gioco in cui assume che il giocatore avversario potrebbe commettere degli errori sulle sue scelte future e nello stesso tempo fa l'assunzione su se stesso, di non commettere mai alcun errore e di giocare sempre la sua azione migliore ad ogni nodo di scelta. In un gioco a due giocatori a somma zero questo è un problema che viene affrontato come la risoluzione di un problema di programmazione lineare. Possiamo fare un esempio di calcolo di un equilibrio quasi perfetto(QPE) per un gioco a due giocatori a somma zero.

Per prima cosa partiamo da una formalizzazione di base di un generico gioco perturbato in cui il giocatore uno adotta una strategia atta a contrastare una strategia minmax da parte del giocatore avversario:

$$max_x \; x^T(Ay)$$
$$Ex = e$$
$$x \geq k(\epsilon)$$

Il problema duale si presenta dunque nella forma:

$$min_{p,u}\, p^T e - \; u^T k(\epsilon)$$
$$E^T p \; \geq Ay + u$$
$$u \; \geq \; 0$$

Il valore della soluzione ottimale rappresenta la massima utilità attesa per il giocatore uno se il suo avversario gioca una strategia y. Adesso possiamo formalizzare la strategia minmax di risposta del giocatore uno al suo avversario:

$$max_{q,v,x}\, q^T f + \; v^T l(\epsilon)$$
$$F^T q \; \leq \; A^T x - v$$
$$Ex = e$$
$$x \geq k(\epsilon)$$
$$v \geq 0$$

Questo problema rappresenta il problema di base per il calcolo di equilibrio quasi perfetto(QPE) in giochi a forma estesa. Possiamo affrontare il suddetto problema di programmazione lineare continua utilizzando il noto algoritmo del simplesso primale che rende efficiente il calcolo dell'equilibrio. Come prima cosa per affrontare questo problema dobbiamo però portare il tutto in forma standard e ottenere quanto segue:

$$max_{p,u,y} \; -p^T e + \; u^T k(\epsilon)$$
$$-E^T p + Ay + u \; \leq \quad 0$$
$$-Fy \; \leq \; -f$$

$$-y \leq -l(\epsilon)$$

$$p, u, y \geq 0$$

In questo tipo di problema standardizzato possiamo a questo punto utilizzare l'algoritmo del simplesso e computare il nostro QPE. Per garantire l'arrivo ad una soluzione ed evitare un ciclo possiamo utilizzare nella scelta della variabile entrante la regola di Bland o del gradiente.

Definizione 3.2 (Equilibrio perfetto in forma estesa EFPE)

Un EFPE, invece, è un profilo di strategia σ tale che esiste un profilo di strategia perturbata σ(ϵ) per cui $\lim_{\epsilon \to 0} \sigma(\epsilon) = \sigma$ per tutti gli agenti i-esimi appartenenti all'insieme N degli agenti in cui la strategia σ(i) è ottimale rispetto a σ(ϵ) per ogni ε ϵ [0, ϵ_0] con $\epsilon_0 > 0$.

I due tipi di strategia(QPE ed EFPE) pur essendo molto simili nella loro formulazione portano come risultati ad equilibri che spesso sono disgiunti tra loro. Portiamo un paio di esempi chiarificatori riguardo alla computazione di equilibri utilizzando i due differenti profili di strategia.

Primo esempio:

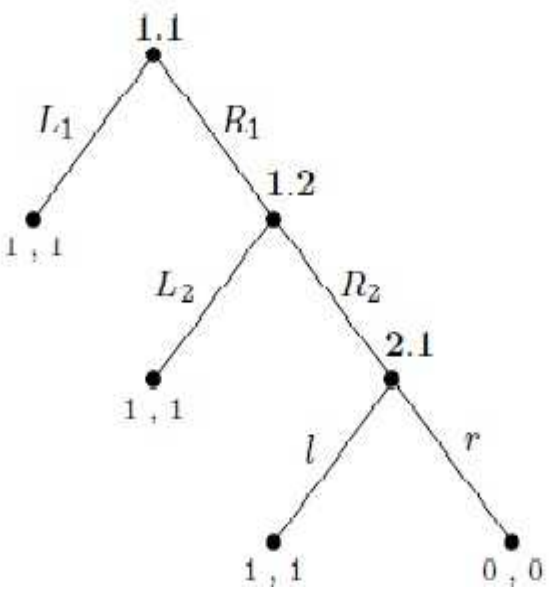


Figura 11: Albero di un gioco in forma estesa per il calcolo di un equilibrio EFPE

Se consideriamo il calcolo di un QPE, ovvero dove consideriamo che solo il secondo giocatore possa sbagliare con probabilità $\epsilon > 0$ nella scelta al suo nodo troviamo come QPE i nodi foglia: $(L_1,*)$, $(R_1L_2,*)$ mentre l'unico EFPE del gioco, ovvero considerando che entrambi i giocatori abbiano possibilità di sbagliare otteniamo : $(L_1,*)$. I due insiemi non coincidono sebbene l'EFPE sia anche QPE. Ora vediamo un secondo esempio più complesso.

Secondo esempio:

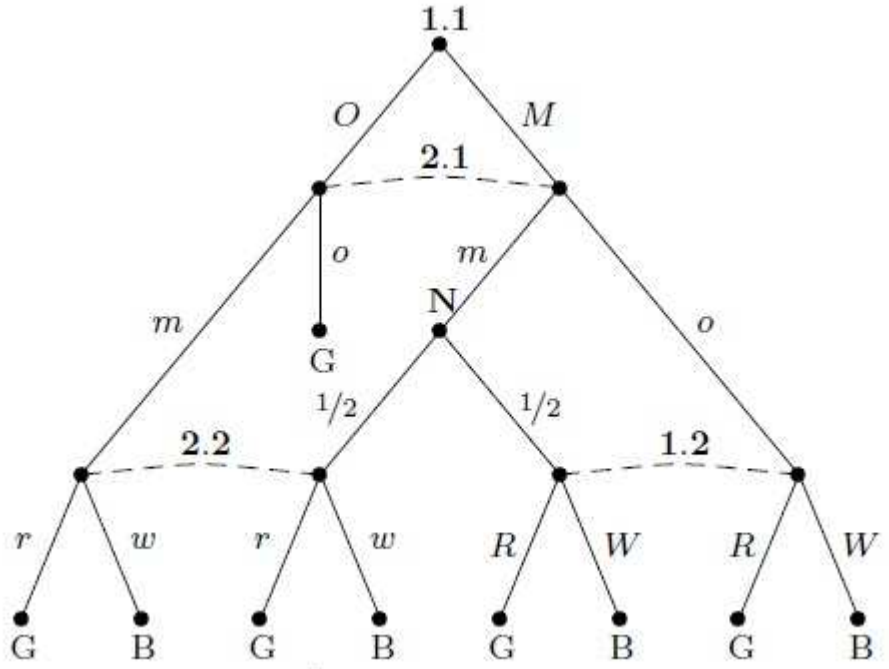


Figura 12: Albero di un gioco ad informazione imperfetta per il calcolo di un equilibrio EFPE

In questo caso abbiamo come EFPE: (O, mr) e (MR, o) mentre come QPE otteniamo: (MR, mr). In questo caso i due insieme di equilibri sono completamente disgiunti ottenendo nodi foglia totalmente differenti.

Questi esempi volevano mostrare come L'EFPE non sia un raffinamento del QPE ma che le due tipologie di equilibri sono differenti. Ora entriamo più nel dettaglio nel calcolo di un equilibrio EFPE.

Consideriamo la formalizzazione di un semplice gioco perturbato in forma estesa:

$$F_i{*}x'_i = f_i - F_i{*}l_i(\epsilon)$$

$$F_i^T{*}v_i - U_i{*}x'_{-i} - U_i * l_{-i}(\epsilon) \geq 0$$

$$x'_i \geq 0$$

$$x'^T_i * (F_i^T * v_i - U_i{*}x'_{-i} - U_i * l_{-i}(\epsilon)) = 0$$

Dove F_i è la matrice della forma sequenza del giocatore i-esimo, f_i è il vettore dei termini noti per il giocatore i-esimo, $l_i(\epsilon)$ è la perturbazione introdotta per il giocatore i-esimo, U_i è la matrice utilità per il giocatore i-esimo e v_i rappresenta la variabile duale della funzione di utilità attesa sempre per il giocatore i-esimo. Da questa generalizzazione di base passeremo a trattare l'EFPE in giochi perturbati uniformi e in giochi non uniformi.

3.2 Calcolo di un EFPE in giochi perturbati uniformi

In questo caso per parlare di EFPE, tenendo conto di un gioco a forma estesa per due agenti, dobbiamo introdurre un ulteriore perturbazione oltre $l_i(\epsilon)$ per il giocatore avversario. Si farà una differenza fra le due perturbazioni per evitare di confonderci con il calcolo di un QPE e si parlerà di perturbazione primale quella del giocatore i-esimo e di perturbazione duale quella del giocatore avversario e ciò che si ottiene quindi è un gioco la cui formalizzazione prevede la ridefinizione di alcuni vincoli mostrati precedentemente e si ha:

$$F_i{*}x'_i = f_i - F_i{*}l_i(\epsilon)$$

$$x'_i \geq 0$$

$$F_i^T(\epsilon) * v_i - U_i(\epsilon){*}x'_{-i} - U_i * l_{-i}(\epsilon) \geq 0$$

$$x'^T_i * (F_i^T(\epsilon) * v_i - U_i(\epsilon){*}x'_{-i} - U_i(\epsilon) * l_{-i}(\epsilon)) = 0$$

Così facendo abbiamo modificato le matrici utilità e della forma sequenza per il giocatore i-esimo tenendo conto della perturbazione dello stesso. Per una formalizzazione di questo tipo non possiamo utilizzare i noti algoritmi di computazione per il calcolo di equilibri, come può essere il Lemke, ma gli stessi vanno modificati per tener conto anche della perturbazione per il giocatore avversario. In un tableau di questo tipo ogni elemento è polinomiale in ϵ e ogni elemento del tableau è un vettore che riporta i corrispondenti coefficienti polinomiali e ad ogni passo di step il suo grado aumenta di un unità. Questo vettore dovrebbe essere inizializzato, quindi, in maniera dinamica perché appunto non è possibile sapere a priori di quanti gradi sarà composto lo stesso. E' garantito che questo tipo di algoritmo terminerà sempre arrivando ad una soluzione ma si pone il

problema della rappresentazione di un tableau tridimensionale. In un calcolatore odierno è impossibile tener traccia di una matrice a tre dimensioni se non per giochi veramente piccoli. Per questo è stato fatto un ulteriore raffinamento per rendere computabile questo problema. In questo caso si parla di calcolo di un EFPE in giochi perturbati non uniformi.

3.3 Calcolo di un EFPE in giochi perturbati non uniformi

Questo ulteriore raffinamento, di cui si è accennato precedentemente, viene utilizzato per rendere il calcolo di un EFPE più efficiente. Come abbiamo appena visto abbiamo modificato le matrici del giocatore i-esimo tenendo conto anche per esso della perturbazione ma così facendo abbiamo reso l'algoritmo di risoluzione inattuabile perché si tratterebbe ad ogni passo di pivoting di moltiplicare e dividere polinomi in ϵ la cui lunghezza aumenta all'aumentare degli step di pivoting e questo perché la perturbazione duale è moltiplicativa invece che additiva. Quello che vogliamo fare ora è rendere quindi questa seconda perturbazione di tipo additiva invece che moltiplicativa e per fare ciò l'idea base è quella di forzare la perturbazione primale in termini di grado di ϵ ai due agenti. Consideriamo quindi che la perturbazione per uno dei due agenti sia per tutte le strategie con gradi di perturbazione inferiori rispetto all'avversario. Possiamo definire quindi le due perturbazioni come:

$$l'_1(\epsilon) = l_1\ (\epsilon)$$

$$l'_2(\epsilon) = \epsilon^{l_1+1} l_2\ (\epsilon)$$

La perturbazione per il secondo giocatore ha in questo caso dei gradi più bassi rispetto al primo giocatore ed è dunque meno significativa.

E' possibile eseguire delle modifiche anche sulla perturbazione duale, che può essere trattata separatamente dalla primale perché non hanno gradi di perturbazione sovrapposti, in modo che essa diventi una sorta di malus che è minimo per i vincoli duali corrispondenti alle sequenze più corte ed è massimo in corrispondenza delle sequenze più lunghe. Definiamo dunque la perturbazione duale come $r_i(\epsilon)$ e quindi abbiamo nel caso di due giocatori:

$$r_1(\epsilon, q) = -\frac{\epsilon^{l_1+1}}{l_1(\epsilon,q)}$$

$$r_2(\epsilon, q) = -\frac{\epsilon^{l_1+l_2+3}}{l_2(\epsilon,q)}$$

Anche per questo tipo di algoritmo si garantisce sempre la terminazione e l’arrivo ad una soluzione.

Capitolo 4

Algoritmo di risoluzione

In questo capitolo tratteremo dell'algoritmo che abbiamo studiano e creato per la risoluzione di giochi a somma zero, giochi di interesse per la nostra tesi. Nella prima sezione, la 4.1 ci occuperemo della formulazione di base di un gioco a somma zero nei dettagli, nella sezione 4.2 parleremo, invece, della creazione del nostro tableau sulla base della formulazione poi esposta nella sezione precedente e infine nella sezione 4.3 tratteremo le ottimizzazioni che abbiamo inserito nell'algoritmo e nella memorizzazione del tableau per aumentarne l'efficienza.

4.1 Formulazione

Come prima cosa per la risoluzione di giochi a due giocatori a somma zero è necessario tener conto del nostro problema di programmazione lineare base di massimizzazione, come abbiamo accennato nel capitolo precedente, che è scritto nella forma:

$$Max\ x_i^T * U * x_{-i}$$
$$F_i * x_i = f_i$$
$$x_i \geq l_i(\epsilon)$$

dove x_i e x_{-i} sono i vettori delle variabili decisionali del giocatore i-esimo e del giocatore i-menounesimo, la matrice U è la matrice delle utilità del giocatore i-esimo, la matrice F_i è la matrice delle sequenze di scelta del giocatore i-esimo, f_i è il vettore dei termini noti sempre del giocatore i-esimo mentre $l_i(\epsilon)$ rappresenta la perturbazione per il giocatore i-esimo.

Al problema eseguiamo, quindi, una sostituzione di variabile del tipo y_i = x_i - $l_i(\epsilon)$ e otteniamo il nostro nuovo problema:

$$Max\ y_i^T * U_i * x_{-i} + l_i(\epsilon) * U_i * x_{-i}$$
$$F_i * y_i = f_i - F_i * l_i(\epsilon)$$
$$y_i \geq 0$$

Ora che abbiamo il nostro nuovo problema di massimizzazione eseguiamo il duale del suddetto e otteniamo il problema di minimizzazione nella forma:

$$Min\ (f_i - F_i * l_i(\epsilon))^T * V_i$$
$$F_i^T * V_i \geq U_i * x_{-i}$$
$$V_i\ libero$$

In questa formulazione appare la nuova variabile V_i per il giocatore i-esimo. Essa rappresenta la variabile duale che esprime l'utilità attesa per lo stesso giocatore.

A questo punto è necessario porre dei vincoli anche su x_{-i} del tipo:

$$F_{-i} * x_{-i} = f_{-i}$$
$$x_{-i} \geq l_{-i}(\epsilon)$$
$$V_i\ libero$$

Una volta posto i vincoli otteniamo il nostro problema finale di programmazione lineare dal quale possiamo ricavare il tableau definitivo. Il problema con l'introduzione dei vincoli diviene il seguente:

$$Min\ (f_i\ -\ F_i * l_i(\epsilon))^T * V_i$$

$$F_i^T * V_i\ \geq\ U_i * x_{-i} + U_i * l_{-i}(\epsilon)$$

$$F_{-i} * x_{-i} = f_{-i} - F_{-i} * l_{-i}(\epsilon)$$

$$x_{-i} \geq 0$$

$$V_i\ libero$$

Considerando che $V_i = V_{i+} - V_{i-}$ possiamo sostituire la seguente relazione nel nostro problema precedente e ottenere quindi un raffinamento del tipo:

$$Min\ (f_i\ -\ F_i * l_i(\epsilon))^T * (V_{i+} - V_{i-})$$

$$F_i * (V_{i+} - V_{i-})\ \geq\ U_i * x_{-i} + U_i * l_{-i}(\epsilon)$$

$$F_{-i} * x_{-i} = f_{-i} - F_{-i} * l_{-i}(\epsilon)$$

$$x_{-i},\ V_{i+},\ V_{i-}\ \geq\ 0$$

Otteniamo così un nuovo problema in cui eliminiamo lo stato di libertà delle variabili V(i) riuscendo a porre in questo modo una forzatura in segno positiva delle suddette. Attraverso quest'ultima formalizzazione possiamo ricavare il tableau del nostro problema di programmazione lineare ed esporre il funzionamento del nostro algoritmo nel dettaglio. Dopo una prima breve fase dove l'algoritmo caricherà i dati da un file di testo in cui già sono memorizzati tutti i valori necessari di un albero di gioco, tra cui il numero delle sequenze di entrambi i giocatori e i valori delle matrici di utilità e di sequenza, il nostro algoritmo comincerà a compiere step di pivoting consecutivi, secondo il noto algoritmo del simplesso su cui è basato lo stesso, al fine di arrivare ad una soluzione efficiente del problema. Chiaramente va detto che esistono più possibilità di configurazioni del risolutore in base al tipo di perturbazione richiesta e al tipo di metodo utilizzato nella selezione delle variabili entranti. Queste sono tutte condizioni che influiscono sull'efficienza dell'algoritmo stesso e quindi anche sui tempi di risoluzione effettivi di un generico albero di gioco. Ma vediamo ora nel dettaglio il nostro tableau.

4.2 Creazione del tableau

Dal problema di programmazione lineare precedentemente trovato possiamo creare il nostro tableau di base al quale applicheremo il noto algoritmo del simplesso per la sua risoluzione:

		V_{i+} | V_{i-}			
	S	$(f_i - F_i * l_i(\epsilon))^T$	x_{-i}	b	ε
	-I	F_i^T	$-U_i$	0	$U_i * l_{-i}(\epsilon)$
	0	0	F_{-i}	f_{-i}	$-F_{-i} * l_{-i}(\epsilon)$

Figura 13: Rappresentazione del tableau standard per un gioco in forma estesa a somma zero

Nel suddetto tableau rispetto alla formulazione precedente abbiamo aggiunto una parte aggiuntiva introducendo le cosiddette variabili di slack S per portare il problema in forma standard.

Il tableau sopra riportato in realtà non è quello che verrà utilizzato per la risoluzione del problema ma è quello in cui verranno introdotte delle ottimizzazioni al fine di ridurne le dimensioni con lo scopo di risparmiare spazio in memoria e rendendo anche minore l'eventuale introduzione di variabili di slack per avere una base di partenza ammissibile in quanto riusciremo ad ottenerne già una parte semplicemente attraverso una piccola operazione.

4.3 Ottimizzazioni

Nel seguito tratteremo tutte le migliorie e le ottimizzazioni introdotte per rendere l'algoritmo di risoluzione dei giochi a somma zero computazionalmente più efficiente e più veloce. Come prima cosa parleremo della riduzione del tableau per occupare meno spazio in memoria.

4.3.1 Riduzione del tableau

Consideriamo la formulazione precedente in cui abbiamo introdotto delle variabili di slack per portare il problema in forma standard. Nell'introduzione delle suddette variabili vista la formulazione del problema si vede la comparsa nel tableau di una matrice identità a valori negativi mentre perché l'algoritmo del simplesso possa inizialmente operare sul problema abbiamo bisogno di avere una base ammissibile di partenza e quindi di una matrice identità a valori positivi.

Consideriamo la seguente riga della nostra formulazione:

$$F_i^T * V_i \geq U_i * x_{-i} + U_i * l_{-i}(\epsilon)$$

Possiamo riscrivere la disuguaglianza nel seguente modo cambiandone il verso:

$$F_i^T * V_i \leq -U_i * x_{-i} - U_i * l_{-i}(\epsilon)$$

A questo punto otteniamo:

$$F_i^T * V_i + U_i * x_{-i} \leq -U_i * l_{-i}(\epsilon)$$

La riscrittura della suddetta diseguaglianza ci permette di inserire questa volta delle normali variabili di slack a valori positivi e quindi di avere una parte della matrice base ammissibile di partenza, a differenza del problema precedente, risparmiando altresì spazio in memoria per la memorizzazione del tableau che a questo punto diventa:

	S	V_{i+} | V_{i-}	x_{-i}	b	ε
		$(f_i - F_i * l_i(\epsilon))^T$			
	-I	F_i^T	U_i	0	$-U_i * l_{-i}(\epsilon)$
	0	0	F_{-i}	f_{-i}	$-F_{-i} * l_{-i}(\epsilon)$

Figura 14: Rappresentazione ottimizzata del tableau per un gioco in forma estesa a somma zero

Rispetto al precedente tableau vediamo un generale cambio di segno nella metà orizzontale superiore del tableau. Per ottenere un problema simile al precedente e quindi con valori della matrice U_i negativi possiamo scalare la stessa sottraendo ad ogni elemento della matrice il valore massimo delle utilità per il giocatore i-esimo meno uno senza altresì andare ad alterare la soluzione del sistema.

Con questa semplice operazione abbiamo bisogno dell'introduzione di variabili di slack per ottenere una base di partenza ammissibile solo per le righe della metà orizzontale inferiore del tableau rispetto alla precedente formulazione in cui avremmo dovuto introdurre tante variabili di slack quante il numero di righe del tableau stesso.

4.3.2 Rappresentazione

I giochi che andremo a risolvere attraverso l'algoritmo del simplesso prevedono una notevole quantità di variabili in base e quindi la memorizzazione di un tableau davvero enorme. Considerando che la maggior parte delle matrici che compongono il tableau nella sua interezza

sono sparse si è pensato, per risparmiare spazio in memoria, di memorizzare il tableau con il metodo a complex column delle matrici sparse. In pratica invece di avere una matrice che rappresenta l'intero tableau abbiamo tre vettori, uno per tener traccia di quanti elementi diversi da zero sono presenti nella i-esima colonna, uno per tenere traccia della riga nel tableau in cui è situato ogni elemento diverso da zero e l'ultimo vettore per tenere traccia dei valori stessi del tableau. Questo sistema viene utilizzato anche per la memorizzazione della matrice di perturbazione nel qual caso la risoluzione del problema ne preveda il suo utilizzo. Questo sistema permette di risparmiare una notevole quantità di memoria e non rappresenta un problema quando dobbiamo eseguire le operazioni di aggiornamento conseguente all'ingresso di una nuova variabile in base in quanto l'unica parte del tableau che verrà memorizzata come matrice ed aggiornata interamente è quella che contiene la base ammissibile del sistema.

Capitolo 5

Valutazioni sperimentali

In questo capitolo inseriremo le valutazioni sperimentali effettuate mostrando attraverso dei grafici i tempi di risoluzione dei giochi a somma zero. Nel capitolo 5.1 affronteremo come sono organizzati i grafici e come possono essere analizzati. Nel capitolo 5.2 metteremo i dati riguardanti il calcolo dell'equilibrio QPE. Nel 5.2 inseriremo i dati riguardanti il calcolo dell'equilibrio NE e nel capitolo 5.4 quelli del calcolo dell'equilibrio EFPE.

5.1 Organizzazione grafici

I grafici sono così organizzati:

esistono quattro tipologie di grafico per ogni tipo di equilibrio.

L'asse orizzontale mostra i livelli di profondità a cui arrivano l'albero di gioco e questo valore varia tra i due e i dodici.

L'asse verticale mostra invece i tempi di risoluzione dei giochi. All'interno di uno stesso grafo abbiamo sei valutazioni ognuna riguardante il livello di branching dell'albero di gioco.

Il livello di branching considerato varia tra due e sei.

Nei giochi da noi considerati ogni livello di branching è caratterizzato da un colore differente per renderne più facile la lettura.

Nel primo grafico dei quattro visualizzeremo i risultati del calcolo del corrispondente equilibrio avendo come information set dell'albero di gioco nullo, nel secondo grafico invece abbiamo un information set pari a 0.5, nel terzo grafico abbiamo un information set pari a 0.8 ed infine nell'ultimo grafico abbiamo un information set pari a 1.

5.2 QPE

Qui di seguito i grafici per il calcolo dell'equilibrio QPE:

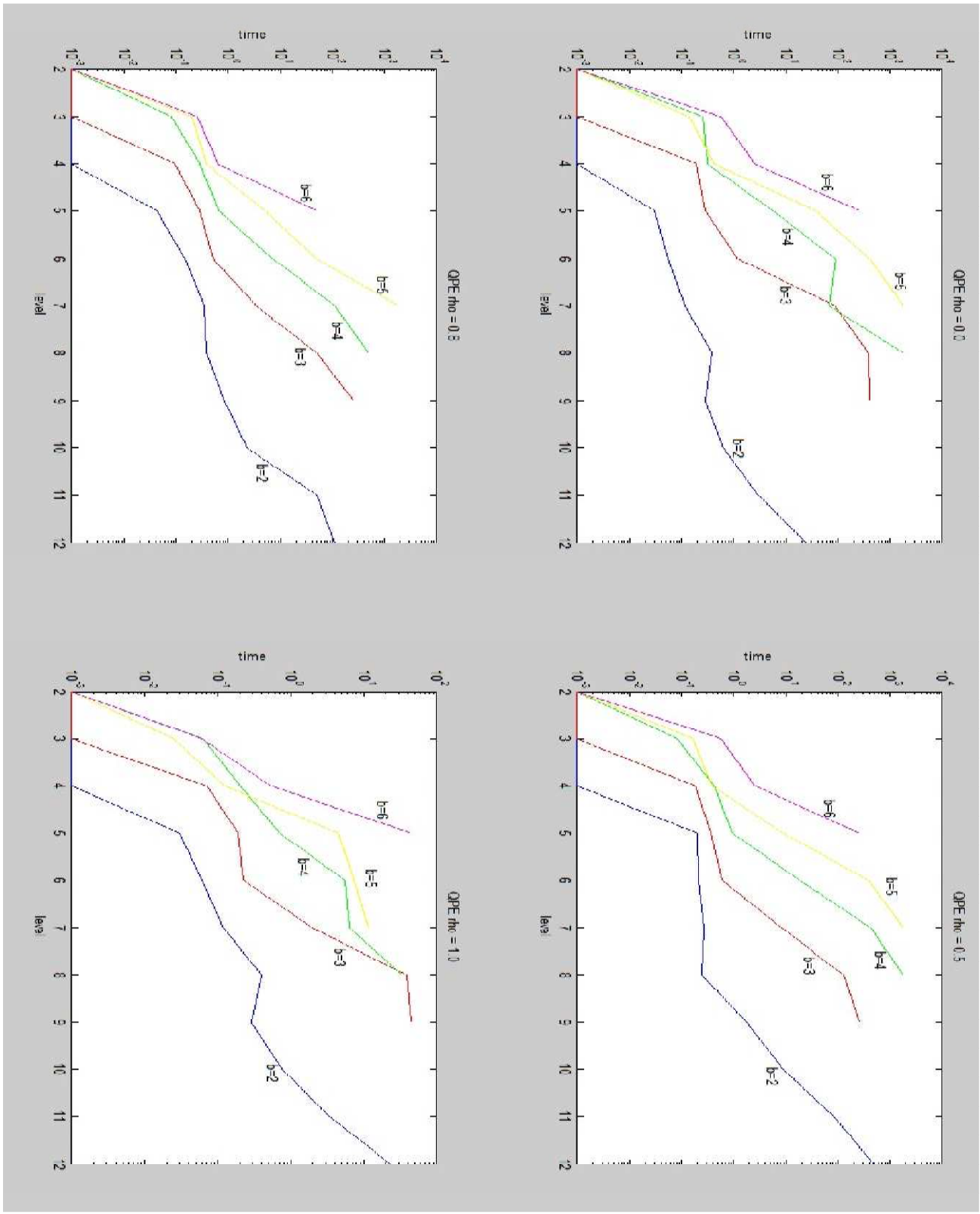


Figura 15: Grafici riguardanti il tempo di calcolo dell'equilibrio QPE

5.3 NE

Qui di seguito i grafici per il calcolo dell'equilibrio NE:

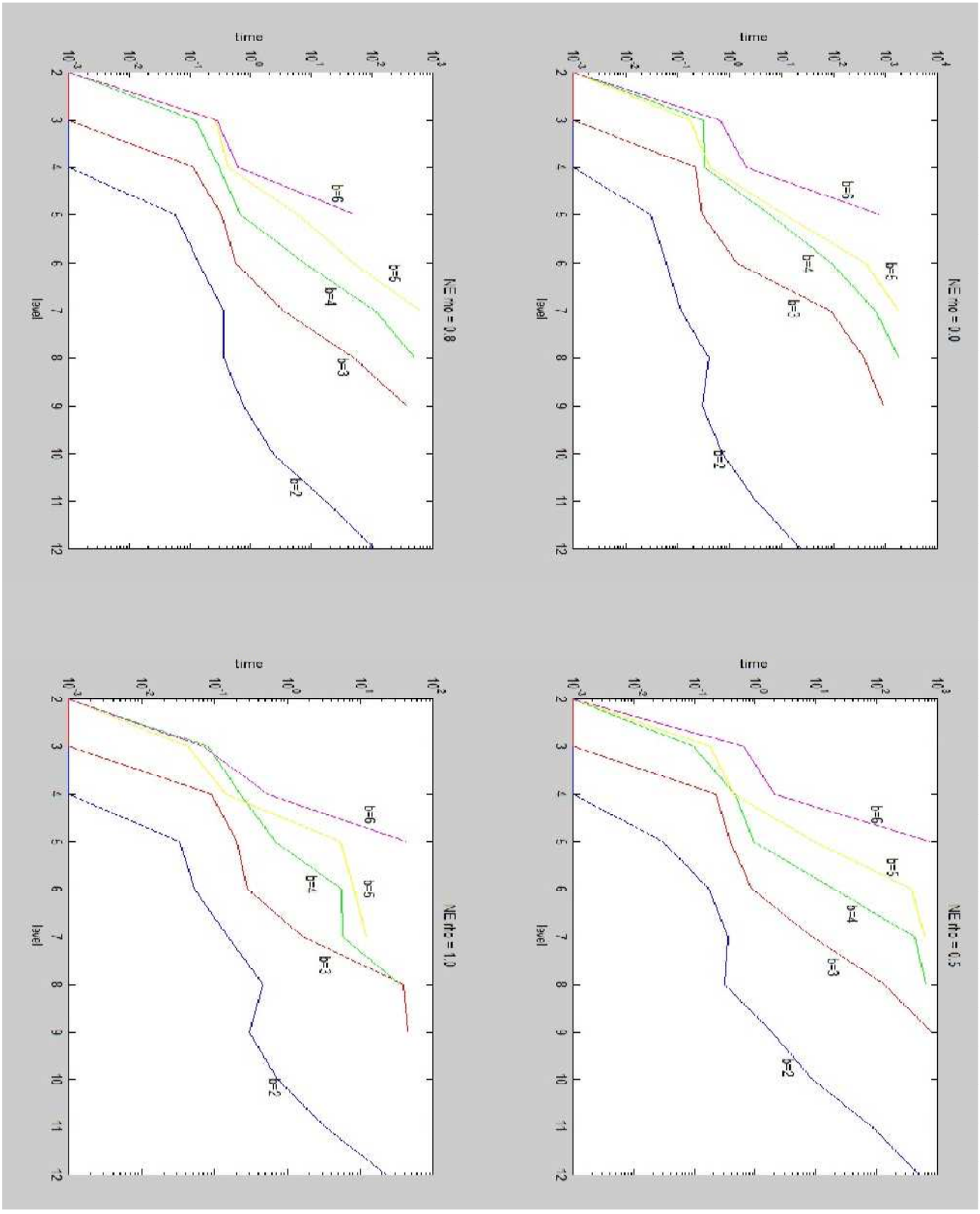


Figura 16: Grafici riguardanti il tempo di calcolo dell'equilibrio NE

5.4 EFPE

Qui di seguito i grafici per il calcolo dell'equilibrio EFPE:

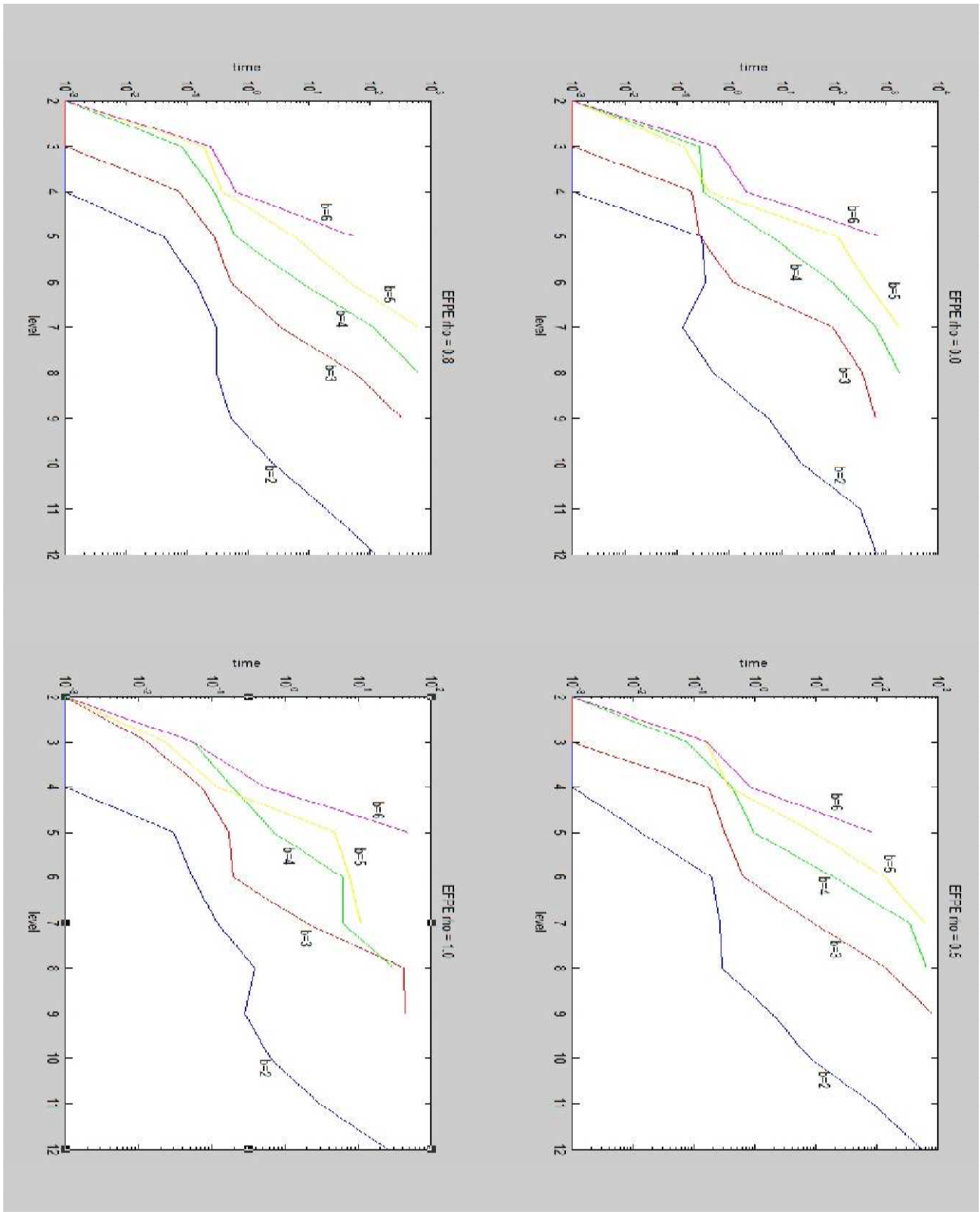


Figura 17: Grafici riguardanti il tempo di calcolo dell'equilibrio EFPE

Capitolo 6

Conclusioni e sviluppi futuri

A conclusione del lavoro, vengono ripercorsi le linee guida e i principali risultati ottenuti. Al termine alcuni spunti per lavori futuri.

6.1 Conclusioni

Nel corso degli anni molti ricercatori hanno proposto metodi ed algoritmi per automatizzare le ricerche di strategie che portassero a stati di equilibrio stabili nei giochi.

La Teoria dei Giochi, da ormai un secolo, offre concetti di equilibrio e modellazioni di giochi che possono rappresentare molte situazioni reali. Il principale concetto, ed anche il più studiato, è l'equilibrio di Nash: in tempi recentissimi sono state proposte diverse soluzioni computazionalmente efficienti per il calcolo di tale equilibrio.

Analizzando la realtà sembra che le assunzioni dell'equilibrio di Nash non sempre siano verificate; questo ha portato alla definizione di diversi equilibri stabili che rilassano le ipotesi alla base del Nash.

Il cuore del presente lavoro è stato proporre delle formulazioni e algoritmi finalizzati alla computazione di più di questi equilibri: l'equilibrio quasi perfetto(QPE) e l'equilibrio perfetto in giochi a forma estesa(EFPE) per giochi a somma zero.

Dopo aver introdotto il modello della struttura dei giochi e le definizioni propedeutiche, si è passati all'analisi teorica dettagliata degli equilibri e dei loro raffinamenti. Una volta inquadrato il problema dal punto di vista teorico, si sono sviluppati gli strumenti che hanno portato al calcolo dei vari tipi di equilibri per i giochi a somma zero.

Lo studio è stato completato con una serie di risultati pratici, circa la complessità ogni tipo di equilibrio: si è visto che l'algoritmo funzioni con tempistiche più basse per quanto riguarda il calcolo del QPE mentre per il calcolo di un EFPE possiede delle tempistiche più basse solo per giochi di piccole dimensioni mentre vanno ad aumentare i tempi per giochi a grandi dimensioni. Comunque le simulazioni dimostrano una buona efficienza nel calcolo di entrambi i tipi di equilibrio.

6.2 Sviluppi futuri

L'analisi di equilibri diversi dal Nash non ha ancora riscosso il meritato successo nell'ambito computer science; per questo il lavoro lascia molte porte aperte:

- studiare metodi di ricerca ed equilibri per giochi in cui c'è incertezza sui parametri.

- formulare la nozione di apprendimento in modo formale nei giochi ripetuti e sviluppare soluzioni computazionalmente efficienti di agenti evolutivi che giocano;

- analizzare soluzioni alternative ai vari tipi di equilibri considerati e progettare metodi per la ricerca computazionalmente efficiente di questi nuovi equilibri.

Bibliografia

[Shoam and Leyton-Brown, 2009] Shoam, Y. And Leyton-Brown, K. (2009). “Multiagent system”.

[Vidal, 2007] Vidal, J. M. (2007). “Fundamentals of multiagent system”.

[Nudelman, Wortman, Shoam and Leyton-Brown, 2004] Nudelman, E and Wortman, J. and Shoam, Y. and Leyton-Brown, K. (2004). “Run the GAMUT: A comprehensive Approach to Evaluating Game-Theoretic Algorithms”.

[Sandholm, Gilpin and Conitzer, 2005] Sandholm, T. and Gilpin, A. and Conitzer, V. (2005). “Mixed-Integer Programming Methods for finding Nash Equilibria”.

[Gatti and Iuliano, 2011] Gatti, N. and Iuliano, C. (2011). “Computing an Extensive-Form Perfect Equilibrium in Two-Player Games”.

[Koller and Pfeffer, 1997] Koller, D. and Pfeffer, A. (1997). “Representations and Solutions for Game-Theoretic Problems”.

[Forrest anf Goldfarb, 1992] Forrest, J. J. and Goldfarb, D. (1992). “Steepest-edge simplex algorithm for linear programming”.

Appendice A

	QPE(s)	EFPE(s)	NE(s)
• 2-2-2-0.0-0.0	<0.001	<0.001	<0.001
• 2-2-2-0.0-0.5	<0.001	<0.001	<0.001
• 2-2-2-0.0-0.8	<0.001	<0.001	<0.001
• 2-2-2-0.5-0.0	<0.001	<0.001	<0.001
• 2-2-2-0.5-0.5	<0.001	<0.001	<0.001
• 2-2-2-0.5-0.8	<0.001	<0.001	<0.001
• 2-2-2-0.8-0.0	<0.001	<0.001	<0.001
• 2-2-2-0.8-0.5	<0.001	<0.001	<0.001
• 2-2-2-1.0-0.0	<0.001	<0.001	<0.001
• 2-2-2-1.0-0.5	<0.001	<0.001	<0.001
• 2-2-2-1.0-0.8	<0.001	<0.001	<0.001
• 2-2-3-0.0-0.0	<0.001	<0.001	<0.001
• 2-2-3-0.0-0.5	<0.001	<0.001	<0.001
• 2-2-3-0.0-0.8	<0.001	<0.001	<0.001
• 2-2-3-0.5-0.0	<0.001	<0.001	<0.001
• 2-2-3-0.5-0.5	<0.001	<0.001	<0.001
• 2-2-3-0.5-0.8	<0.001	<0.001	<0.001
• 2-2-3-0.8-0.0	<0.001	<0.001	<0.001
• 2-2-3-0.8-0.5	<0.001	<0.001	<0.001
• 2-2-3-0.8-0.8	<0.001	<0.001	<0.001
• 2-2-3-1.0-0.0	<0.001	<0.001	<0.001
• 2-2-3-1.0-0.5	<0.001	<0.001	<0.001
• 2-2-3-1.0-0.8	<0.001	<0.001	<0.001
• 2-2-4-0.0-0.0	<0.001	<0.001	<0.001
• 2-2-4-0.0-0.5	<0.001	<0.001	<0.001
• 2-2-4-0.0-0.8	<0.001	<0.001	<0.001
• 2-2-4-0.5-0.0	<0.001	<0.001	<0.001
• 2-2-4-0.5-0.5	<0.001	<0.001	<0.001
• 2-2-4-0.5-0.8	<0.001	<0.001	<0.001
• 2-2-4-0.8-0.0	<0.001	<0.001	<0.001
• 2-2-4-0.8-0.5	<0.001	<0.001	<0.001
• 2-2-4-0.8-0.8	<0.001	<0.001	<0.001

• 2-2-4-1.0-0.0	<0.001	<0.001	<0.001
• 2-2-4-1.0-0.5	<0.001	<0.001	<0.001
• 2-2-4-1.0-0.8	<0.001	<0.001	<0.001
• 2-2-5-0.0-0.0	0.2	0.2	0.2
• 2-2-5-0.0-0.5	0.1	0.1	0.15
• 2-2-5-0.0-0.8	0.61	0.58	0.6
• 2-2-5-0.5-0.0	0.02	0.01	0.05
• 2-2-5-0.5-0.5	0.01	0.01	0.01
• 2-2-5-0.5-0.8	0.03	0.02	0.03
• 2-2-5-0.8-0.0	<0.001	<0.001	<0.001
• 2-2-5-0.8-0.5	0.04	0.04	0.05
• 2-2-5-0.8-0.8	0.09	0.08	0.12
• 2-2-5-1.0-0.0	<0.001	<0.001	<0.001
• 2-2-5-1.0-0.5	<0.001	<0.001	<0.001
• 2-2-5-1.0-0.8	0.09	0.09	0.1
• 2-2-6-0.0-0.0	0.6	0.6	0.6
• 2-2-6-0.0-0.5	0.218	0.167	0.15
• 2-2-6-0.0-0.8	0.25	0.25	0.2
• 2-2-6-0.5-0.0	0.21	0.164	0.18
• 2-2-6-0.5-0.5	0.204	0.188	0.12
• 2-2-6-0.5-0.8	0.228	0.22	0.219
• 2-2-6-0.8-0.0	0.1	0.1	0.1
• 2-2-6-0.8-0.5	0.22	0.2	0.2
• 2-2-6-0.8-0.8	0.123	0.108	0.112
• 2-2-6-1.0-0.0	<0.001	<0.001	<0.001
• 2-2-6-1.0-0.5	0.1	0.1	0.1
• 2-2-6-1.0-0.8	0.085	0.062	0.06
• 2-2-7-0.0-0.0	0.133	0.122	0.113
• 2-2-7-0.0-0.5	0.118	0.101	0.218
• 2-2-7-0.0-0.8	0.143	0.168	0.21
• 2-2-7-0.5-0.0	0.344	0.322	0.43
• 2-2-7-0.5-0.5	0.237	0.252	0.391
• 2-2-7-0.5-0.8	0.239	0.198	0.24
• 2-2-7-0.8-0.0	0.362	0.299	0.355
• 2-2-7-0.8-0.5	0.452	0.398	0.421
• 2-2-7-0.8-0.8	0.207	0.202	0.3

• 2-2-7-1.0-0.0	0.1	0.1	0.1
• 2-2-7-1.0-0.5	0.146	0.152	0.243
• 2-2-7-1.0-0.8	0.106	0.098	0.115
• 2-2-8-0.0-0.0	0.293	0.288	0.296
• 2-2-8-0.0-0.5	1.021	0.691	1.062
• 2-2-8-0.0-0.8	0.507	0.466	0.689
• 2-2-8-0.5-0.0	0.195	0.504	0.276
• 2-2-8-0.5-0.5	0.205	0.148	0.302
• 2-2-8-0.5-0.8	0.342	0.212	0.35
• 2-2-8-0.8-0.0	0.378	0.321	0.433
• 2-2-8-0.8-0.5	0.323	0.287	0.35
• 2-2-8-0.8-0.8	0.423	0.277	0.289
• 2-2-8-1.0-0.0	0.73	0.722	0.8
• 2-2-8-1.0-0.5	0.21	0.196	0.253
• 2-2-8-1.0-0.8	0.255	0.238	0.333
• 2-2-9-0.0-0.0	6.886	7.514	6.898
• 2-2-9-0.0-0.5	5.023	5.52	5.356
• 2-2-9-0.0-0.8	2.897	3.965	2.972
• 2-2-9-0.5-0.0	2.306	2.402	2.439
• 2-2-9-0.5-0.5	1.896	1.877	1.902
• 2-2-9-0.5-0.8	1.24	1.211	1.252
• 2-2-9-0.8-0.0	1.2	0.422	0.94
• 2-2-9-0.8-0.5	0.85	0.559	0.863
• 2-2-9-0.8-0.8	0.445	0.556	0.448
• 2-2-9-1.0-0.0	0.185	0.18	0.19
• 2-2-9-1.0-0.5	0.318	0.312	0.322
• 2-2-9-1.0-0.8	0.379	0.357	0.385
• 2-2-10-0.0-0.0	17.373	18.108	17.376
• 2-2-10-0.0-0.5	28.97	29.87	28.15
• 2-2-10-0.0-0.8	22.21	21.766	22.346
• 2-2-10-0.5-0.0	6.395	6.228	6.371
• 2-2-10-0.5-0.5	9.91	10.789	9.978
• 2-2-10-0.5-0.8	9.396	6.88	9.565
• 2-2-10-0.8-0.0	2.002	1.721	1.992
• 2-2-10-0.8-0.5	2.551	3.056	2.892
• 2-2-10-0.8-0.8	2.45	2.62	2.098

• 2-2-10-1.0-0.0	0.143	0.143	0.144
• 2-2-10-1.0-0.5	0.644	0.301	0.578
• 2-2-10-1.0-0.8	1.449	1.436	1.452
• 2-2-11-0.0-0.0	404.07	439.21	405.326
• 2-2-11-0.0-0.5	233.97	262.113	234.521
• 2-2-11-0.0-0.8	213.03	258.87	214.25
• 2-2-11-0.5-0.0	135.18	128.755	137.021
• 2-2-11-0.5-0.5	53.28	59.32	53.489
• 2-2-11-0.5-0.8	53.718	58.11	55.019
• 2-2-11-0.8-0.0	24.277	26.312	24.176
• 2-2-11-0.8-0.5	10.167	12.745	9.875
• 2-2-11-0.8-0.8	16.06	17.036	16.233
• 2-2-11-1.0-0.0	0.866	0.362	1.023
• 2-2-11-1.0-0.5	0.858	0.489	0.86
• 2-2-11-1.0-0.8	8.206	7.857	7.879
• 2-2-12-0.0-0.0	1120	1314	1090
• 2-2-12-0.0-0.5	532.765(30%)	562.21(30%)	534.64(30%)
• 2-2-12-0.0-0.8	148.723(50%)	188.61(50%)	150.41(50%)
• 2-2-12-0.5-0.0	368.31	388.231	370.155
• 2-2-12-0.5-0.5	667.92	712.91	670.134
• 2-2-12-0.5-0.8	456.1	488.321	458.267
• 2-2-12-0.8-0.0	89.15	93.715	90.232
• 2-2-12-0.8-0.5	110.78	133.248	113.358
• 2-2-12-0.8-0.8	126.2	146.11	128.621
• 2-2-12-1.0-0.0	3.85	3.784	4.466
• 2-2-12-1.0-0.5	3.124	3.582	3.40
• 2-2-12-1.0-0.8	63.133	68.532	61.956
• 3-3-2-0.0-0.0	<0.001	<0.001	<0.001
• 3-3-2-0.0-0.5	<0.001	<0.001	<0.001
• 3-3-2-0.0-0.8	<0.001	<0.001	<0.001
• 3-3-2-0.5-0.0	<0.001	<0.001	<0.001
• 3-3-2-0.5-0.5	<0.001	<0.001	<0.001
• 3-3-2-0.5-0.8	<0.001	<0.001	<0.001
• 3-3-2-0.8-0.0	<0.001	<0.001	<0.001
• 3-3-2-0.8-0.5	<0.001	<0.001	<0.001

• 3-3-2-0.8-0.8	<0.001	<0.001	<0.001
• 3-3-2-1.0-0.0	<0.001	<0.001	<0.001
• 3-3-2-1.0-0.5	<0.001	<0.001	<0.001
• 3-3-2-1.0-0.8	<0.001	<0.001	<0.001
• 3-3-3-0.0-0.0	<0.001	<0.001	<0.001
• 3-3-3-0.0-0.5	<0.001	<0.001	<0.001
• 3-3-3-0.0-0.8	<0.001	<0.001	<0.001
• 3-3-3-0.5-0.0	<0.001	<0.001	<0.001
• 3-3-3-0.5-0.5	<0.001	<0.001	<0.001
• 3-3-3-0.5-0.8	<0.001	<0.001	<0.001
• 3-3-3-0.8-0.0	<0.001	<0.001	<0.001
• 3-3-3-0.8-0.5	<0.001	<0.001	<0.001
• 3-3-3-0.8-0.8	<0.001	<0.001	<0.001
• 3-3-3-1.0-0.0	<0.001	<0.001	<0.001
• 3-3-3-1.0-0.5	<0.001	<0.001	<0.001
• 3-3-3-1.0-0.8	0.04	0.04	0.04
• 3-3-4-0.0-0.0	0.2	0.2	0.2
• 3-3-4-0.0-0.5	0.26	0.275	0.355
• 3-3-4-0.0-0.8	0.1	0.1	0.1
• 3-3-4-0.5-0.0	0.02	0.02	0.06
• 3-3-4-0.5-0.5	0.29	0.265	0.334
• 3-3-4-0.5-0.8	0.23	0.221	0.27
• 3-3-4-0.8-0.0	0.01	0.01	0.05
• 3-3-4-0.8-0.5	0.05	0.04	0.1
• 3-3-4-0.8-0.8	0.23	0.17	0.18
• 3-3-4-1.0-0.0	<0.001	<0.001	<0.001
• 3-3-4-1.0-0.5	<0.001	<0.001	<0.001
• 3-3-4-1.0-0.8	0.21	0.215	0.267
• 3-3-5-0.0-0.0	0.313	0.301	0.307
• 3-3-5-0.0-0.5	0.246	0.25	0.256
• 3-3-5-0.0-0.8	0.28	0.266	0.31
• 3-3-5-0.5-0.0	0.687	0.542	0.7
• 3-3-5-0.5-0.5	0.262	0.255	0.311
• 3-3-5-0.5-0.8	0.144	0.138	0.151
• 3-3-5-0.8-0.0	0.446	0.433	0.483
• 3-3-5-0.8-0.5	0.307	0.311	0.357

• 3-3-5-0.8-0.8	0.111	0.07	0.133
• 3-3-5-1.0-0.0	0.1	0.1	0.1
• 3-3-5-1.0-0.5	0.42	0.367	0.455
• 3-3-5-1.0-0.8	0.052	0.05	0.056
• 3-3-6-0.0-0.0	1.3	1.452	1.55
• 3-3-6-0.0-0.5	1.69	1.678	2.105
• 3-3-6-0.0-0.8	0.552	0.448	0.583
• 3-3-6-0.5-0.0	0.324	0.428	0.504
• 3-3-6-0.5-0.5	1.02	1.118	1.548
• 3-3-6-0.5-0.8	0.427	0.356	0.486
• 3-3-6-0.8-0.0	0.47	0.421	0.5
• 3-3-6-0.8-0.5	0.614	0.608	0.684
• 3-3-6-0.8-0.8	0.468	0.497	0.523
• 3-3-6-1.0-0.0	0.35	0.339	0.42
• 3-3-6-1.0-0.5	0.056	0.062	0.082
• 3-3-6-1.0-0.8	0.28	0.19	0.35
• 3-3-7-0.0-0.0	199.87	216.377	200.964
• 3-3-7-0.0-0.5	40.09	44.572	40.864
• 3-3-7-0.0-0.8	18.75	19.566	18.988
• 3-3-7-0.5-0.0	7.517	6.482	6.968
• 3-3-7-0.5-0.5	11.344	13.057	11.487
• 3-3-7-0.5-0.8	6.79	6.875	7.566
• 3-3-7-0.8-0.0	1.352	1.139	0.869
• 3-3-7-0.8-0.5	3.627	3.792	3.872
• 3-3-7-0.8-0.8	5.18	5.26	5.321
• 3-3-7-1.0-0.0	0.315	0.308	0.32
• 3-3-7-1.0-0.5	0.56	0.488	0.253
• 3-3-7-1.0-0.8	4.995	5.221	4.439
• 3-3-8-0.0-0.0	690.22	760.22	694.694
• 3-3-8-0.0-0.5	240.986(50%)	245.43(50%)	243.7(50%)
• 3-3-8-0.0-0.8	33.751(70%)	29.222(70%)	34.82(70%)
• 3-3-8-0.5-0.0	39.172	38.033	39.359
• 3-3-8-0.5-0.5	166.975	179.215	168.787
• 3-3-8-0.5-0.8	186.535	198.74	187.165
• 3-3-8-0.8-0.0	8.573	8.269	8.512
• 3-3-8-0.8-0.5	21.99	23.102	22.487

3-3-8-0.8-0.8	122.33	128.85	119.17
3-3-8-1.0-0.0	1.08	1.012	1.218
3-3-8-1.0-0.5	1.2	0.829	0.95
3-3-8-1.0-0.8	113.33	121.74	113.05
3-3-9-0.0-0.0	-	-	-
3-3-9-0.0-0.5	752.36(20%)	68.113(10%)	755.1(20%)
3-3-9-0.0-0.8	75.45(60%)	76.572(60%)	76.6(60%)
3-3-9-0.5-0.0	-	-	-
3-3-9-0.5-0.5	499.2(30%)	527.63(30%)	505(30%)
3-3-9-0.5-0.8	18.24(60%)	19.122(60%)	19.5(60%)
3-3-9-0.8-0.0	538.14	497.36	536.845
3-3-9-0.8-0.5	389.8(80%)	357.975(80%)	396.2(80%)
3-3-9-0.8-0.8	149.2(80%)	163.844(80%)	150.1(80%)
3-3-9-1.0-0.0	2.777	2.284	2.986
3-3-9-1.0-0.5	15.914	14.433	16.018
3-3-9-1.0-0.8	116.1	115.32	113.872
4-4-2-0.0-0.0	<0.001	<0.001	<0.001
4-4-2-0.0-0.5	<0.001	<0.001	<0.001
4-4-2-0.0-0.8	<0.001	<0.001	<0.001
4-4-2-0.5-0.0	<0.001	<0.001	<0.001
4-4-2-0.5-0.5	<0.001	<0.001	<0.001
4-4-2-0.5-0.8	<0.001	<0.001	<0.001
4-4-2-0.8-0.0	<0.001	<0.001	<0.001
4-4-2-0.8-0.5	<0.001	<0.001	<0.001
4-4-2-0.8-0.8	<0.001	<0.001	<0.001
4-4-2-1.0-0.0	<0.001	<0.001	<0.001
4-4-2-1.0-0.5	<0.001	<0.001	<0.001
4-4-2-1.0-0.8	<0.001	<0.001	<0.001
4-4-3-0.0-0.0	0.4	0.4	0.4
4-4-3-0.0-0.5	0.15	0.142	0.23
4-4-3-0.0-0.8	0.23	0.25	0.32
4-4-3-0.5-0.0	0.05	0.045	0.06
4-4-3-0.5-0.5	0.04	0.04	0.04
4-4-3-0.5-0.8	0.15	0.147	0.181
4-4-3-0.8-0.0	0.01	0.01	0.02

• 4-4-3-0.8-0.5	0.03	0.03	0.05
• 4-4-3-0.8-0.8	0.2	0.2	0.3
• 4-4-3-1.0-0.0	<0.001	<0.001	<0.001
• 4-4-3-1.0-0.5	0.04	0.04	0.05
• 4-4-3-1.0-0.8	0.15	0.13	0.19
• 4-4-4-0.0-0.0	0.19	0.19	0.22
• 4-4-4-0.0-0.5	0.305	0.302	0.31
• 4-4-4-0.0-0.8	0.455	0.462	0.468
• 4-4-4-0.5-0.0	0.55	0.523	0.62
• 4-4-4-0.5-0.5	0.353	0.362	0.38
• 4-4-4-0.5-0.8	0.356	0.348	0.394
• 4-4-4-0.8-0.0	0.19	0.19	0.19
• 4-4-4-0.8-0.5	0.26	0.249	0.284
• 4-4-4-0.8-0.8	0.389	0.395	0.402
• 4-4-4-1.0-0.0	0.05	0.05	0.06
• 4-4-4-1.0-0.5	0.24	0.233	0.27
• 4-4-4-1.0-0.8	0.323	0.289	0.356
• 4-4-5-0.0-0.0	13.32	14.887	13.587
• 4-4-5-0.0-0.5	2.5	2.701	2.757
• 4-4-5-0.0-0.8	1.193	1.175	1.096
• 4-4-5-0.5-0.0	1.15	1.155	1.158
• 4-4-5-0.5-0.5	0.56	0.548	0.552
• 4-4-5-0.5-0.8	1.117	1.115	1.131
• 4-4-5-0.8-0.0	0.5	0.51	0.511
• 4-4-5-0.8-0.5	0.242	0.239	0.248
• 4-4-5-0.8-0.8	1.21	1.119	1.215
• 4-4-5-1.0-0.0	0.6	0.6	0.6
• 4-4-5-1.0-0.5	0.5	0.5	0.5
• 4-4-5-1.0-0.8	1.024	1.002	0.913
• 4-4-6-0.0-0.0	52.83	56.957	52.581
• 4-4-6-0.0-0.5	189.19	196.218	191.015
• 4-4-6-0.0-0.8	28.05	26.782	28.675
• 4-4-6-0.5-0.0	4.85	5.427	5.2
• 4-4-6-0.5-0.5	33.213	34.877	34.363
• 4-4-6-0.5-0.8	18.5	18.469	18.624
• 4-4-6-0.8-0.0	1.05	0.683	0.785

• 4-4-6-0.8-0.5	5.1	4.833	5.132
• 4-4-6-0.8-0.8	15.62	16.248	16.714
• 4-4-6-1.0-0.0	0.109	0.109	0.109
• 4-4-6-1.0-0.5	1.21	1.19	1.218
• 4-4-6-1.0-0.8	15.35	16.846	15.272
• 4-4-7-0.0-0.0	-	-	-
• 4-4-7-0.0-0.5	68.23(40%)	67.256(40%)	70(40%)
• 4-4-7-0.0-0.8	68.96(60%)	66.436(60%)	68.5(60%)
• 4-4-7-0.5-0.0	1158	927.561(20%)	1152
• 4-4-7-0.5-0.5	77.87(60%)	74.318(60%)	78.9(60%)
• 4-4-7-0.5-0.8	18.45(70%)	17.225(70%)	19.6(70%)
• 4-4-7-0.8-0.0	123.99	113.395	124.523
• 4-4-7-0.8-0.5	161.28	166.87	160.788
• 4-4-7-0.8-0.8	50.1(90%)	51.236(90%)	51.3(90%)
• 4-4-7-1.0-0.0	1.544	0.84	0.453
• 4-4-7-1.0-0.5	15.9	16.328	15.724
• 4-4-7-1.0-0.8	1.456(90%)	1.515(90%)	1.52(90%)
• 4-4-8-0.0-0.0	-	-	-
• 4-4-8-0.0-0.5	-	-	-
• 4-4-8-0.0-0.8	-	-	-
• 4-4-8-0.5-0.0	-	-	-
• 4-4-8-0.5-0.5	39.578(20%)	42.011(20%)	40(20%)
• 4-4-8-0.5-0.8	40.65(40%)	43.1(40%)	42.9(40%)
• 4-4-8-0.8-0.0	1477	-	1448
• 4-4-8-0.8-0.5	7.211(20%)	8.147(20%)	8.2(20%)
• 4-4-8-0.8-0.8	10.123(40%)	9.875(40%)	11.7(40%)
• 4-4-8-1.0-0.0	37.57	37.221	38.495
• 4-4-8-1.0-0.5	61.48	52.548	62.155
• 4-4-8-1.0-0.8	2.992(90%)	2.39(90%)	3.1(90%)
• 5-5-2-0.0-0.0	<0.001	<0.001	<0.001
• 5-5-2-0.0-0.5	<0.001	<0.001	<0.001
• 5-5-2-0.0-0.8	<0.001	<0.001	<0.001
• 5-5-2-0.5-0.0	<0.001	<0.001	<0.001
• 5-5-2-0.5-0.5	<0.001	<0.001	<0.001
• 5-5-2-0.5-0.8	<0.001	<0.001	<0.001

• 5-5-2-0.8-0.0	<0.001	<0.001	<0.001
• 5-5-2-0.8-0.5	<0.001	<0.001	<0.001
• 5-5-2-0.8-0.8	<0.001	<0.001	<0.001
• 5-5-2-1.0-0.0	<0.001	<0.001	<0.001
• 5-5-2-1.0-0.5	<0.001	<0.001	<0.001
• 5-5-2-1.0-0.8	<0.001	<0.001	<0.001
• 5-5-3-0.0-0.0	0.22	0.21	0.254
• 5-5-3-0.0-0.5	0.12	0.113	0.151
• 5-5-3-0.0-0.8	0.085	0.082	0.102
• 5-5-3-0.5-0.0	0.301	0.295	0.353
• 5-5-3-0.5-0.5	0.121	0.114	0.111
• 5-5-3-0.5-0.8	0.06	0.06	0.08
• 5-5-3-0.8-0.0	0.12	0.132	0.155
• 5-5-3-0.8-0.5	0.041	0.046	0.086
• 5-5-3-0.8-0.8	0.45	0.387	0.53
• 5-5-3-1.0-0.0	<0.001	<0.001	<0.001
• 5-5-3-1.0-0.5	0.01	0.01	0.02
• 5-5-3-1.0-0.8	0.065	0.058	0.11
• 5-5-4-0.0-0.0	0.5	0.5	0.5
• 5-5-4-0.0-0.5	0.413	0.402	0.426
• 5-5-4-0.0-0.8	0.325	0.312	0.355
• 5-5-4-0.5-0.0	0.33	0.328	0.348
• 5-5-4-0.5-0.5	0.386	0.39	0.492
• 5-5-4-0.5-0.8	0.415	0.406	0.45
• 5-5-4-0.8-0.0	0.58	0.553	0.623
• 5-5-4-0.8-0.5	0.219	0.198	0.236
• 5-5-4-0.8-0.8	0.36	0.346	0.385
• 5-5-4-1.0-0.0	0.1	0.1	0.1
• 5-5-4-1.0-0.5	0.089	0.081	0.107
• 5-5-4-1.0-0.8	0.182	0.174	0.223
• 5-5-5-0.0-0.0	275.15	300.101	280.1
• 5-5-5-0.0-0.5	37.795	40.233	39.72
• 5-5-5-0.0-0.8	13.14	13.872	14.162
• 5-5-5-0.5-0.0	8.124	8.259	8.132
• 5-5-5-0.5-0.5	4.815	4.576	5.24
• 5-5-5-0.5-0.8	12.96	13.628	15.011

• 5-5-5-0.8-0.0	1.216	1.274	0.991
• 5-5-5-0.8-0.5	1.033	1.471	1.199
• 5-5-5-0.8-0.8	12.85	13.842	15.116
• 5-5-5-1.0-0.0	0.18	0.18	0.19
• 5-5-5-1.0-0.5	0.47	0.382	0.372
• 5-5-5-1.0-0.8	12.82	13.744	15.253
• 5-5-6-0.0-0.0	1202	1261	1190
• 5-5-6-0.0-0.5	5.769(30%)	5.841(30%)	6.1(30%)
• 5-5-6-0.0-0.8	7.892(60%)	8.796(60%)	8.2(60%)
• 5-5-6-0.5-0.0	58.245	54.695	57.826
• 5-5-6-0.5-0.5	979.8	268.464(60%)	975.133
• 5-5-6-0.5-0.8	89.23(90%)	72.258(90%)	91.1(90%)
• 5-5-6-0.8-0.0	10.35	10.746	10.725
• 5-5-6-0.8-0.5	114.6	106.81	114.682
• 5-5-6-0.8-0.8	24.166(90%)	20.058(90%)	25.4(90%)
• 5-5-6-1.0-0.0	1.137	1.114	1.21
• 5-5-6-1.0-0.5	13.14	13.127	13.258
• 5-5-6-1.0-0.8	8.12(90%)	8.267(90%)	9.7(90%)
• 5-5-7-0.0-0.0	-	-	-
• 5-5-7-0.0-0.5	-	-	-
• 5-5-7-0.0-0.8	-	-	-
• 5-5-7-0.5-0.0	-	-	-
• 5-5-7-0.5-0.5	37.94(20%)	38.613(20%)	38.3(20%)
• 5-5-7-0.5-0.8	33.41(40%)	35.335(40%)	36.2(40%)
• 5-5-7-0.8-0.0	-	-	-
• 5-5-7-0.8-0.5	5.339(30%)	5.388(30%)	6.1(30%)
• 5-5-7-0.8-0.8	9.843(50%)	11.176(50%)	10(50%)
• 5-5-7-1.0-0.0	4.21	4.257	4.321
• 5-5-7-1.0-0.5	17.261(90%)	16.138(90%)	17.3(90%)
• 5-5-7-1.0-0.8	13.119(90%)	11.776(90%)	14.7(90%)
• 6-6-2-0.0-0.0	<0.001	<0.001	<0.001
• 6-6-2-0.0-0.5	<0.001	<0.001	<0.001
• 6-6-2-0.0-0.8	<0.001	<0.001	<0.001
• 6-6-2-0.5-0.0	<0.001	<0.001	<0.001
• 6-6-2-0.5-0.5	<0.001	<0.001	<0.001

• 6-6-2-0.5-0.8	<0.001	<0.001	<0.001
• 6-6-2-0.8-0.0	<0.001	<0.001	<0.001
• 6-6-2-0.8-0.5	<0.001	<0.001	<0.001
• 6-6-2-0.8-0.8	<0.001	<0.001	<0.001
• 6-6-2-1.0-0.0	<0.001	<0.001	<0.001
• 6-6-2-1.0-0.5	<0.001	<0.001	<0.001
• 6-6-2-1.0-0.8	<0.001	<0.001	<0.001
• 6-6-3-0.0-0.0	0.9	0.9	1.1
• 6-6-3-0.0-0.5	0.488	0.476	0.535
• 6-6-3-0.0-0.8	0.29	0.282	0.31
• 6-6-3-0.5-0.0	0.12	0.132	0.16
• 6-6-3-0.5-0.5	0.151	0.145	0.185
• 6-6-3-0.5-0.8	0.22	0.218	0.25
• 6-6-3-0.8-0.0	0.27	0.27	0.27
• 6-6-3-0.8-0.5	0.185	0.172	0.202
• 6-6-3-0.8-0.8	0.31	0.284	0.333
• 6-6-3-1.0-0.0	<0.001	<0.001	<0.001
• 6-6-3-1.0-0.5	0.036	0.031	0.041
• 6-6-3-1.0-0.8	0.144	0.141	0.168
• 6-6-4-0.0-0.0	1.61	0.423	0.393
• 6-6-4-0.0-0.5	4.83	4.67	4.555
• 6-6-4-0.0-0.8	1.153	1.092	1.186
• 6-6-4-0.5-0.0	0.56	0.554	0.568
• 6-6-4-0.5-0.5	0.799	0.773	0.806
• 6-6-4-0.5-0.8	1.126	1.128	1.135
• 6-6-4-0.8-0.0	0.182	0.181	0.186
• 6-6-4-0.8-0.5	0.457	0.402	0.412
• 6-6-4-0.8-0.8	1.266	1.258	1.257
• 6-6-4-1.0-0.0	0.3	0.3	0.3
• 6-6-4-1.0-0.5	0.244	0.248	0.253
• 6-6-4-1.0-0.8	1.05	1.052	1.058
• 6-6-5-0.0-0.0	-	-	-
• 6-6-5-0.0-0.5	377.333	405.952	384.03
• 6-6-5-0.0-0.8	130.227	132.358	134.19
• 6-6-5-0.5-0.0	65.12	62.98	65.595
• 6-6-5-0.5-0.5	43.82	39.472	44.07

• 6-6-5-0.5-0.8	133.51	141.36	132.433
• 6-6-5-0.8-0.0	8.05	7.968	7.738
• 6-6-5-0.8-0.5	11.32	11.688	11.771
• 6-6-5-0.8-0.8	132.5	144.12	132.234
• 6-6-5-1.0-0.0	0.565	0.547	0.57
• 6-6-5-1.0-0.5	1.467	1.342	1.318
• 6-6-5-1.0-0.8	133.61	143.881	129.94