Politecnico di Milano
V Facoltà di Ingegneria
Corso di laurea magistrale in Ingegneria Informatica
Dipartimento di Elettronica, Informazione e Bioingegneria

UN ALGORITMO PER LA RISOLUZIONE DI GIOCHI STOCASTICI CON SPAZIO DEGLI STATI CONTINUO

Relatore: Prof. Nicola GATTI
Correlatore: Prof. Marcello RESTELLI

Tesi di Laurea di:
Antonio DIONISIO
Matr. 771151

Anno Accademico 2012-2013

Ai miei genitori, mio fratello, mio zio Domenico ed Antonella,
che mi hanno sempre sostenuto e spronato durante questo mio lungo ma
bellissimo percorso.

Sommario

La Teoria dei Giochi è definita come lo studio matematico dell'interazione tra decisori indipendenti; in particolare, nella cosiddetta teoria dei giochi non cooperativa, vengono trattate situazioni in cui gli interessi dei decisori sono in netto conflitto tra loro. Nel corso degli ultimi anni gli interessi di ricerca si sono concentrati sullo studio di giochi caratterizzati da uno spazio delle azioni continuo, in cui i decisori hanno accesso ad un numero infinito di azioni, e da uno spazio degli stati continuo, cioè i decisori possono trovarsi a decidere rispetto ad un numero infinito di situazioni o stati diversi, in quanto questi giochi permettono di modellizzare gran parte delle situazioni del mondo reale che si vorrebbero analizzare e risolvere.

In questa tesi verranno studiate alcune estensioni dei cosiddetti giochi polinomiali a due giocatori e a somma zero, giochi in cui si ha uno spazio delle azioni continuo e la funzione di reward è in forma polinomiale nelle variabili d'azione dei giocatori in gioco; in particolare ci si concentrererà sulla sottoclasse dei giochi stocastici chiamata MDP con proprietà di Switching Controller, studiando prima il caso più semplice in cui si ha uno spazio degli stati discreto, quindi passando al caso più complesso in cui si ha uno spazio degli stati continuo supportato nel generico intervallo $[c, d]$. In entrambi i casi verrà presentato il relativo algoritmo di risoluzione formulato in programmazione semidefinita positiva, quindi verranno discusse le diverse proprietà e verranno valutate sperimentalmente le prestazioni relativamente a questi algoritmi.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

1.1 Descrizione del progetto di tesi

La Teoria dei Giochi è quella branca della matematica che studia l'interazione tra diversi decisori, o giocatori, in situazioni modellizzate che rappresentano situazioni possibili nel mondo reale. Questa nasce nei primi anni '30 grazie alle ricerche effettuate dal matematico John Von Neumann, quindi nei dieci anni successivi, grazie al lavoro svolto dal matematico e premio Nobel John Nash, vengono definite le sue nozioni fondamentali.

Storicamente, la maggior parte della ricerca in questo ambito è stata diretta verso l'analisi di giochi finiti, cioè con uno spazio delle azioni discreto, dove i giocatori avevano a disposizione un numero limitato di azioni possibili, limitando invece la ricerca relativamente ai giochi infiniti, cioè con spazio delle azioni continuo, dove i giocatori avevano invece un numero potenzialmente infinito di azioni possibili: questo è dovuto al fatto che, fino a pochi anni fà, non esistevano metodi computazionali efficienti per poter risolvere la maggior parte dei giochi appartenenti a quest'ultima classe. Infatti, nonostante l'introduzione fin dagli anni '50 della classe dei giochi polinomiali, che fù da subito considerata come una possibile classe di collegamento tra i giochi finiti e quelli infiniti, si dovette aspettare più di cinquant'anni prima che furono fatti progressi significativi in quest'ultimo ambito di ricerca.

I progressi effettuati nel campo dei giochi polinomiali, che sono di fatto una sottoclasse dei giochi infiniti, si devono infatti al lavoro svolto da Pablo Parrillo [5] [10]: quest'ultimo nel 2006 presentò un algortimo di risoluzione per giochi polinomiali in forma normale, quindi nel 2007 estese la sue ricerche verso la risoluzione di giochi stocastici polinomiali, presentando un algoritmo per la risoluzione di giochi stocastici polinomiali a due giocatori e somma zero

con proprietà di Single Controller, dove c'è un unico giocatore, uguale per tutti gli stati, che governa le probabilità di transizione da uno stato all'altro del gioco stesso. Entrambi questi metodi risolutivi erano basati su tecniche di ottimizzazione in somma di quadrati e su formulazione del problema in programmazione semidefinita positiva.

Successivamente, nel 2011, Guido Bonomi [6] riprese in mano il lavoro di ricerca di Parrillo, estendendolo ai giochi stocastici polinomiali a due giocatori e somma zero con proprietà di Switching Controller, dove c'è sempre un unico giocatore che governa le probabilità di transizione di uno stato, però questo giocatore cambia da stato a stato. Relativamente a questi giochi, Bonomi ha poi presentanto un algortimo iterativo di risoluzione con garanzia di ottimalità della soluzione trovata e garanzia di convergenza.

Il nostro progetto di tesi partirà quindi dall'analisi del lavoro effettuato da Bonomi, estendendo inizialmente l'intervallo di supporto delle azioni dei giocatori dal caso limitato $[0, 1]$ ad un caso $[c, d]$ più generico, quindi presentando una versione ottimizzata del suo algoritmo di risoluzione per MDP polinomiali con Switching Controller.

Dopo questa prima parte, il nostro lavoro si concentrerà sull'estensione dello spazio degli stati del gioco: dato che è possibile modellare moltissime situazioni del mondo reale attraverso giochi con spazio degli stati continuo, cioè con un numero infinito e non numerabile di elementi, molta della ricerca di questi ultimi anni si è concentrata su questa particolare classe di giochi.

Sarà quindi discussa la classe dei giochi stocastici polinomiali con spazio degli stati conituno, infine verrà presentato un algoritmo di risoluzione per giochi stocastici polinomiali a due giocatori e somma zero con proprietà di Switching Controller e spazio degli stati continuo, di cui verrà discusso l'errore di approssimazione della soluzione restituita e verranno valutate sperimentalmente le prestazioni.

1.2 Organizzazione del Lavoro

Il nostro lavoro di tesi verrà organizzato come segue.

Nel Capitolo 2, presenteremo le principali nozioni teoriche relative alla teoria dei giochi non cooperativa. Inizialmente verranno descritti i giochi in forma normale, quindi verranno mostrati i diversi concetti di soluzione per questa classe di giochi. Successivamente verranno descritti i giochi stocastici, giochi in cui il gioco è rappresentato da un grafo, e verranno presentate alcune delle principali proprietà di questa classe. Poi verrà descritta nel dettaglio la sottoclasse dei giochi stocastici chiamata MDP: relativamente a questa, verranno mostrate prima soluzioni algoritmiche classiche per giochi

con spazio degli stati discreto, infine verranno mostrate soluzioni algoritmiche più recenti per giochi con spazio degli stati continuo.

Nel Capitolo 3, presenteremo la classe dei giochi polinomiali con spazio degli stati discreto e spazio delle azioni continuo, in cui i giocatori hanno accesso ad un numero infinito di azioni possibili. Inizialmente verranno descritti i giochi polinomiali in forma normale e verrà presentata tutta la teoria necessaria a definire vincoli su polinomi univariati e su sequenze valide di momenti in programmazione semidefinita positiva (SDP), quindi verrà proposta una soluzione di gioco basata sulla programmazione SDP del tutto equivalente alla soluzione classica in programmazione lineare ottimizzata. Verrà poi estesa la precedente trattazione attraverso la presentazione dei giochi stocastici polinomiali, in particolare degli MDP, con proprietà di Single Controller, mostrando anche il relativo algoritmo di risoluzione basato su programmazione SDP. Successivamente, verrà presentato il caso particolare di un MDP polinomiale che valuta la miglior risposta di un giocatore rispetto ad un profilo di strategie fissato del giocatore avversario. Quindi, verrà mostrata la classe dei giochi stocastici polinomiali con proprietà di Switching Controller, che rappresentano una superclasse di quelli Single Controller, e verrà mostrato un algoritmo iterativo in grado di calcolarne una soluzione d'equilibrio ϵ-Nash. Infine, verrà presentato un metodo per l'approssimazione di funzioni continue non in forma polinomiale in funzioni in forma polinomiale, e verrà analizzato l'errore introdotto nella soluzione d'equilibrio restituita dall'algoritmo per MDP con Switching Controller a causa di questa approssimazione sulle funzioni.

Nel Capitolo 4, verrà estesa la classe dei giochi polinomiali ad uno spazio delle azioni e degli stati entrambi continui, dove i giocatori possono trovarsi a decidere rispetto ad un numero infinito di situazioni o stati diversi. Inizialmente verranno presentati i giochi stocastici polinomiali con Single Controller con spazio degli stati continuo, introducendo la fondamentale nozione di funzione Lipschitz-continua che permetterà una campionatura finita di stati per la risoluzione dei problemi di questa classe di giochi; relativamente a questi giochi, verrà mostrato un metodo di risoluzione basato su una coppia di problemi primale e duale formulati in programmazione SDP. Successivamente, l'analisi dei giochi stocastici polinomiali con spazio degli stati continuo verrà estesa anche ai giochi con proprietà di Switching Controller, mostrando, anche in questo caso, un algoritmo iterativo di risoluzione basato su problemi in programmazione SDP. Infine, verrà discussa l'analisi dell'errore teorico introdotto dalla campionatura dello spazio degli stati e dall'assunzione di Lipschitz-continuità delle funzioni caratteristiche del gioco.

Nel Capitolo 5, verranno invece presentati i dati e le valutazioni dei test effettuati durante il corso del nostro lavoro. Inizialmente verrà analizzato

l'algoritmo Switching Controller con spazio degli stati discreto, comparando diverse implementazioni dello stesso e diverse versioni ottimizzazioni dei vincoli effettuate. Poi, verrà analizzato il metodo di risoluzione per MDP con Single Controller e spazio degli stati continuo, comparando le diverse prestazioni di questo metodo, in termini di accuratezza della soluzione e tempi di computazione, con quelle dell'algoritmo per MDP con spazio degli stati discreti, avendo come benchmark la soluzione in forma chiusa di un problema LQR. Infine, verrà presentata la valutazione sperimentale delle prestazioni, in termini di numero di iterazioni e di tempi di computazione, dell'algoritmo per MDP con Switching Controller con spazio degli stati continuo al variare della dimensione del gioco in ingresso.

Infine, nel Capitolo 6 presenteremo le conclusioni derivanti dal lavoro svolto, presentando anche alcuni spunti per future ricerche basate su questo lavoro.

Capitolo 2

Stato dell'Arte

In questo capitolo presenteremo inizialmente una breve introduzione alla Teoria dei Giochi, cioè il campo scientifico su cui lavoreremo per il nostro progetto di tesi.

Dopo questa breve introduzione, nelle Sezioni 2.2 e 2.3, inizieremo a fare un'ampia panoramica sui Giochi in forma Normale, cioè i giochi definiti nella forma canonica di rappresentazione, soffermandoci soprattutto sui vari metodi di risoluzione tra quelli più noti ed usati in letteratura.

Dopo questa panoramica, nella Sezione 2.4, si passerà all'introduzione di una particolare classe di giochi, i Giochi Stocastici, che rappresentano tutti quei giochi in cui è presente almeno un elemento di incertezza e che includono al loro interno una sottoclasse di giochi chiamata MDP, che è uno degli argomenti principali della nostra trattazione.

Infine, nelle Sezioni 2.5 e 2.6, parleremo dei giochi MDP, illustrando le varie soluzioni presenti nello stato dell'arte attuale per quanto riguarda giochi in cui lo stato di gioco ha un numero prima finito (discreto) poi infinito (continuo) di elementi.

2.1 Introduzione alla Teoria dei Giochi

La Teoria dei Giochi è definita come lo studio matematico dell'interazione tra agenti/giocatori indipendenti, razionali e self-interested, cioè egoisti, ed è comunemente applicata a diverse discipline quali economia, scienze politiche, biologia, psicologia e informatica. Delle sue tante applicazioni e ramificazioni, sicuramente la più importante, nonché quella direttamente attinente al nostro progetto, è quella chiamata teoria dei giochi non cooperativa: quest'ultima tratta, con un approccio prettamente individualistico, cioè concentrato sul

singolo agente, situazioni in cui gli interessi di agenti diversi sono in conflitto tra loro. All'interno di questo ramo della teoria dei giochi ci soffermeremo in particolare sulla definizione e sulla risoluzione dei giochi in forma Normale, cioè scritti secondo la rappresentazione canonica della teoria dei giochi che andrò ad illustrare nel prossimo paragrafo.

Tornando alla definizione appena data di teoria dei giochi, quando si parla di agenti egoisti, o self-interested, si vuole indicare agenti che hanno una propria descrizione degli stati del gioco con le proprie preferenze/interessi relative a quest'ultimi, più o meno indipendetemente da cosa fà l'avversario, e che agiscono quindi al fine di raggiungere gli stati di maggiore interesse, in modo da ottenere il meglio per sé in termini di utilità; per quanto riguarda invece la definizione di agente razionale, questa indica il fatto che l'agente è capace di ordinare coerentemente i valori del gioco, così da poter decidere quale azione fare per massimizzare la propria utilità. La teoria che tratta la modellizzazione degli interessi di un agente è chiamata teoria dell'utilità: questo tipo di approccio ha l'obiettivo di quantificare il grado di preferenza di un agente valutando tutte le possibili alternative a sua disposizione, anche in presenza di incertezza, e di modellarlo su di una specifica funzione di utilità associata all'agente. Quest'ultima quindi risulta essere una funzione che associa ad ogni stato del gioco un valore reale, detto anche outcome, che indica la preferenza dell'agente rispetto allo stato stesso; in situazioni di incertezza, l'utilità dell'agente è invece definita tramite il valore atteso della funzione d'utilità rispetto alla distribuzione di probabilità sugli stati. Ovviamente, data una specifica funzione d'utilità, l'obiettivo dell'agente sarà quello di massimizzarla: se l'agente quindi ha a disposizione tutte le informazioni sul gioco, questo obiettivo risulta concettualmente facile da raggiungere, anche in presenza di incertezza, semplicemente agendo razionalmente. Tuttavia, se il gioco prevede la presenza di 2 o più giocatori/agenti razionali, raggiungere l'obiettivo per l'agente risulta subito più difficile, in quanto l'azione degli altri giocatori in gioco può influenzare positivamente o negativamente la sua utilità, e viceversa.

2.2 Giochi in forma Normale

Nella Teoria dei Giochi, la forma Normale, nota anche come forma Strategica, è la rappresentazione canonica delle interazioni strategiche tra giocatori di un dato gioco: grazie a questa è possibile rappresentare l'utilità di ogni giocatore per ogni stato del mondo nel caso in cui gli stati del gioco dipendano direttamente dalle azioni combinate dei giocatori in gioco. Lo studio di questa particolare forma di giochi è fondamentale in quanto moltissime delle

altre forme di gioco più interessanti, tra cui citiamo i giochi in forma estesa e i giochi Bayesiani (che contengono elementi di incertezza nell'ambiente di gioco), possono esser ricondotte ad equivalenti giochi in forma Normale. Le nozioni che presenteremo in questa sezione si basano su quelle presenti in [25], [26], [27].

Definizione 1 (Gioco in Forma Normale)**.** Un gioco in forma Normale, finito e ad n giocatori, è rappresentato dalla tupla (N, A, u), dove:

- N è un insieme finito di n giocatori, indicizzati da i;
- $A = A_1 \times \ldots \times A_n$, dove A_i è l'insieme finito delle azioni disponibili per l'i-esimo giocatore, ed ogni vettore $a = (a_1, \ldots, a_n) \in A$ è definito come profilo d'azione;
- $u = (u_1, \ldots, u_n)$, dove $u_i : A \mapsto \mathbb{R}$ è la funzione di utilità a valori reali per l'i-esimo giocatore.

Qui si assume implicitamente che il set degli outcome corrisponde al set delle azioni. La classica rappresentazione di questi giochi avviene tramite l'uso di matrici ad n dimensioni: nel caso particolare di gioco a 2 giocatori, la matrice bidimensionale che si forma prevede che ad ogni riga corrisponda una possibile azione per il giocatore 1, ad ogni colonna corrisponda una possibile azione per il giocatore 2, e che ad ogni cella corrisponda uno dei possibili outcome del gioco in base alle azioni combinate in riga e colonna dei 2 giocatori. L'utilità di ogni giocatore verrà quindi scritta nella cella corrispondente all'outcome relativo, con le utilità ordinate partendo dal primo giocatore.

I giochi in forma Normale possono quindi essere suddivisi in diverse classi di gioco in base ad alcune caratteristiche che può avere la funzione d'utilità dei giocatori, cioè in giochi ad utilità comune oppure giochi a somma costante. La prima classe di giochi, relativa ai cosiddetti giochi ad utilità in comune, raggruppa tutti quei giochi in cui, per ogni profilo d'azione, tutti i giocatori hanno la stessa utilità, cioè i giocatori non sono in conflitto tra loro ma cercano di coordinarsi insieme in modo da fare l'azione che massimizza l'utilità di tutti; questi giochi vengono anche chiamati giochi di squadra o giochi di puro coordinamento.

La seconda classe di giochi invece, relativa ai cosiddetti giochi a somma costante, raggruppa tutti quei giochi in cui, per ogni profilo d'azione, la somma delle utilità dei diversi giocatori equivale sempre ad un valore c costante.

Definizione 2 (Gioco a somma costante)**.** Un gioco a due giocatori in forma normale è a somma costante se esiste una costante c tale che per ogni profilo di strategie $a \in A_1 \times A_2$ si ha $u_1(a) + u_2(a) = c$.

La sottoclasse più conosciuta ed usata di questi giochi è chiamata giochi a somma zero, dove cioè il valore della costante c è pari a zero. Questi giochi sono caratterizzati anche dalla presenza di 2 soli giocatori, i quali quindi si trovano in una situazione di pura concorrenza, in netto contrasto con la prima classe di giochi già presentata. Classici esempi di questa classe di giochi sono il gioco chiamato Sasso, Carta Forbice oppure il gioco di Matching Penny, in cui i 2 giocatori hanno una moneta che mostrano all'avversario girata come testa o croce e in base alla combinazione delle 2 monete mostrate uno dei due giocatori vince (quindi ottiene come utilità +1), mentre l'altro perde (ed ottiene come utilità -1).

2.2.1 Strategie nei giochi in Forma Normale

Definiamo ora i 2 tipi di strategia che un giocatore può scegliere di adottare. La strategia ovviamente più semplice è quella di scegliere un'azione tra quelle a sua disposizione ed effettuarla: questo tipo di strategia è detto strategia pura, mentre la scelta di una stretegia pura per ogni giocatore è detta profilo di strategie pure. Il secondo tipo di strategia invece, sicuramente meno immediato, è quello di randomizzare la scelta di un'azione sull'insieme delle azioni disponibili secondo una qualche distribuzione di probabilità: questo tipo di strategia è detto strategia mista, di cui si riporta la definizione formale di seguito.

Definizione 3 (Strategia Mista)**.** Sia (N, A, u) un gioco in forma Normale, e per qualsiasi insieme X sia $\prod(X)$ l'insieme di tutte le distribuzioni di probabilità rispetto ad X. L'insieme delle strategie miste per l'i-esimo giocatore è quindi $S_i = \prod(A_i)$.

Definizione 4 (Mixed-Strategy Profile)**.** L'insieme dei profili di strategie miste è il Prodotto Cartesiano dei diversi insiemi di strategia miste $S_1 \times \ldots \times S_n$.

Dalla definizione appena data di strategia mista si può intuire che una strategia pura non è altro che un caso particolare di strategia mista, dove un'azione ha probabilità uguale ad 1 di esser eseguita, e tutte le altre hanno probabilità nulla.

Date le precedenti definizioni, si può facilmente intuire che di solito, dell'intero set di azioni a disposizione del giocatore, solo un'azione o un piccolo sottoinsieme di queste verrà selezionato per esser eseguito, da qui la necessità di introdurre la seguente nozione di Supporto di una data strategia.

Definizione 5 (Supporto di una Strategia)**.** Sia $s_i(a_i)$ la probabilità che l'i-esima azione sia giocata rispetto alla strategia mista s_i. Il supporto di una data strategia mista s_i per l'i-esimo giocatore è l'insieme di tutte le strategie pure $\{a_i | s_i(a_i) > 0\}$.

Per completare il discorso relativo alla nozione di strategia, passiamo ora a definire l'utilità dei giocatori dato un particolare profilo di strategie. Nel caso di un profilo di strategie pure, la matrice caratteristica del gioco definisce direttamente l'utilità dei giocatori; nel caso invece di un profilo di strategie miste, l'utilità dei giocatori è definita tramite il concetto di utilità attesa, cioè si calcola la media delle utilità delle azioni supportate dalle strategie dei giocatori, media pesata in base alla distribuzione delle probabilità sulle stesse azioni. Diamo ora una definizione più formale di utilità attesa, data una strategia mista.

Definizione 6 (Utilità attesa)**.** Dato un gioco in forma Normale (N, A, u), l'utilità attesa u_i per l'i-esimo giocatore relativamente al profilo di strategie miste $s = (s_1, \ldots, s_n)$ è definita come

$$u_i(s) = \sum_{a \in A} u_i(a) \prod_{j=1}^{N} s_j(a_j). \tag{2.1}$$

2.3 Concetti di Soluzione per giochi in forma Normale

Dopo aver introdotto tutte le definizioni utili ad una buona rappresentazione del gioco e dei comportamenti dei giocatori al suo interno, possiamo ora iniziare a parlare di come trovare la strategia ottimale per ogni giocatore in gioco. Se il gioco presenta, nel suo caso più semplice e basilare, solo un giocatore, come già detto questo applicherà come sua strategia ottimale la strategia che semplicemente massimizza la sua utilità diretta o attesa in base all'ambiente di gioco. Ovviamente, quando il gioco presenta 2 o più agenti al suo interno, questa semplice idea di strategia ottimale non è più

utile, in quanto quest'ultima non può non dipendere anche dalle strategie adottate dagli altri giocatori presenti, i quali hanno anche loro come obiettivo quello di massimizzare la propria utilità. La teoria dei giochi affronta la ricerca di strategie ottime nel caso multiagente dividendola per sottoinsiemi in base alla tipologia di outocomes che si vogliono ottenere dal gioco, definiti come Concetti di Soluzioni: tra questi i più importanti e conosciuti sono l'Ottimalità di Pareto, che non tratteremo perché non inerente al nostro progetto, e l'equilibrio di Nash, in tutte le sue varianti, di cui ora parleremo ampiamente. Le nozioni che presenteremo in questa sezione si basano su quelle presenti in [25], [26], [27], [28].

2.3.1 Equilibrio di Nash

Passiamo subito alla descrizione dell'Equilibrio di Nash, il concetto di soluzione più conosciuto ed usato nella Teoria dei Giochi. Prima di trattare questo argomento, è però necessaria una breve parentesi sui giochi in generale in modo da introdurre poi più facilmente questo concetto di soluzione. Se consideriamo il punto di vista del singolo agente, se si sapesse come giocherebbero tutti i giocatori a parte l'i-esimo, si potrebbero formalmente definire lo strategy profile $s_{-i} = (s_1, \ldots, s_{i-1}, s_{i+1}, \ldots, s_n)$, che rappresenta uno strategy profile senza la strategia dell'i-esimo giocatore, e $s = (s_i, s_{-i})$; date queste definizioni, se tutti i giocatori diversi dall'i-esimo si impegnassero a giocare effettivamente un dato profilo di strategie s_{-i}, allora l'i-esimo giocatore dovrebbe solo affrontare il problema di scegliere la sua Best Response, o miglior risposta, rispetto a s_{-i}. Definiamo formalmente cosa vuol dire applicare la Best Response per un dato agente.

Definizione 7 (Best Response)**.** La miglior risposta, o Best Response, dell'i-esimo giocatore ad un dato strategy profile s_{-1} è una strategia mista $s_i^* \in S_i$ tale che, per tutte le strategie $s_i \in S_i$, $u_i(s_i^*, s_{-i}) = u_i(s_i, s_{-i})$.

La best response non è necessariamente unica: tranne nel caso particolare in cui sia un'unica strategia pura la nostra best response, il numero di best response è potenzialmente infinito. Infatti, quando il supporto di una best response s^* include due o più azioni, l'agente deve essere indifferente tra di loro randomizzando, altrimenti se non lo fosse vorrebbe dire che preferirebbe ridurre a zero la probabilità di giocare almeno una delle azioni; così facendo, almeno una combinazione di queste azioni deve essere quindi anche una best response, non solo la combinazione ottenuta in particolare in s^*. Analogamente, se ci sono due strategie pure che sono singolarmente best response,

una qualsiasi combinazione delle due è necessariamente anch'essa una best response.

Tuttavia, nei giochi in generale, un agente non sà a priori quali saranno le strategie che gli altri giocatori in gioco decideranno di adottare: la nozione di best response, pertanto, non può esser definita come un concetto di soluzione. Partendo però dall'idea di best response possiamo definire quello che è probabilmente il più importante concetto di soluzione nella teoria dei giochi non cooperativi, cioè l'equilibrio di Nash, di cui diamo subito una definizione formale.

Definizione 8 (Equilibrio di Nash)**.** Un profilo di strategie $s = (s_1, \ldots, s_n)$ è un Equilibrio di Nash se, per tutti gli agenti i, s_i è una best response rispetto a s_{-i}.

Dalla definizione appena data si può intuire che l'equilibrio di Nash è un profilo di strategie stabile: ogni agente infatti, prendendo per certo il fatto che gli altri giocheranno una determinata strategia in base al gioco, non ha nessun incentivo a deviare dalla strategia scelta tramite best response. Gli equilibri di Nash possono essere suddivisi in due categorie, stretti o deboli: nel primo caso la strategia di ogni agente è costituita da un'unica best response rispetto alle strategie degli altri agenti, mentre nel secondo la strategia può non essere unica.

Definizione 9 (Equilibrio Stretto di Nash)**.** Un profilo di strategie $s = (s_1, \ldots, s_n)$ è un equilibrio stretto di Nash se, per ogni agente i e per ogni strategia $s'_i \neq s_i$, si ha $u_i(s_i, s_{-i}) > u_i(s'_i, s_{-i})$.

Definizione 10 (Equilibrio Debole di Nash)**.** Un profilo di strategie $s = (s_1, \ldots, s_n)$ è un equilibrio debole di Nash se, per ogni agente i e per e ogni strategia $s'_i \neq s_i$, si ha $u_i(s_i, s_{-i}) \geq u_i(s'_i, s_{-i})$, ed s non è un equilibrio stretto di Nash.

Da queste 2 definizioni si capisce che gli equilibri stretti sono meno stabili di quelli stretti, perché nel primo caso c'è almeno un giocatore che ha una best response rispetto alle strategie degli altri giocatori che non è la sua strategia d'equilibrio. Possiamo quindi affermare che gli equilibri di Nash in strategie miste sono sempre di tipo debole, mentre quelli in strategie pure possono essere sia deboli che stretti. Per rafforzare l'importanza di questo concetto di soluzione, presentiamo ora il teorema che ne caratterizza l'esistenza, cioè il Teorema di Esistenza dell'Equilibrio di Nash, basato sul teorema di Esistenza dei Punti Fissi di Kakutani.

Teorema 1. *(Nash, 1951) Ogni gioco con un numero finito di giocatori e numero finito di profili di azione ha almeno un equilibrio di Nash.*

A livello di computazione, il problema di ricerca di equilibri di Nash nel caso di nostro interesse, cioè quello di un gioco a 2 giocatori a somma zero, è facilmente risolvibile esprimendo questo problema di ricerca sotto forma di problema di programmazione lineare (LP), di cui parleremo approfonditamente più avanti; questa particolare forma di problema ci garantisce nel caso peggiore la computazione dell'equilibrio in tempo polinomiale. Se però passiamo ad un problema di ricerca di equilibri di Nash in un gioco a 2 giocatori e somma costante diversa da zero, questo richiederà nel caso peggiore un tempo di computazione che è esponenziale nel numero di azioni di ogni giocatore.

Per riassumere tutte le nozioni citate sui giochi in forma normale, presentiamo ora un esempio classico di gioco con un'unico equilibrio di Nash, cioè il Dilemma del Prigioniero. Questo gioco presenta 2 giocatori, in questo caso 2 prigionieri condannati alla galera per un reato, che sono di fronte ad un dilemma: decidere se non confessare (azione A) oppure confessare (azione B) il reato. Le righe e le colonne della matrice presentano le 2 possibili scelte, rispettivamente, per il primo giocatore e per il secondo giocatore:

- nel caso entrambi non confessino il reato, avrebbero ambedue un valore di utilità pari a -1 (cioè passerebbero solo un anno in carcere);
- nel caso uno solo dei 2 confessi, quello che confessa viene premiato con la libertà, cioè con un corrispondente valore di utilità pari a 0, mentre l'altro riceverebbe un valore di -4;
- nel caso entrambi confessino il reato, avrebbero ambedue un valore di utilità pari a -3.

Guardando la matrice del gioco, descritta dalla matrice bidimensionale in Figura 2.1, si nota come la Best Response di entrambi i giocatori sia scegliere l'azione D, cioè confessare: entrambi infatti ricevono un valore di utilità attesa superiore nel caso confessino, a prescindere dall'azione dell'avversario. Prendendo ad esempio il primo giocatore, nel caso giochi l'azione C riceverebbe come valore di utilità -1 o -4, entrambi peggiori dei rispettivi valore di utilità nel caso scegliesse l'azione D, cioè 0 e -3; ovviamente lo stesso ragionamento vale per il secondo giocatore. Così facendo, si otterrà un unico equilibrio di Nash, dato dalle strategie pure (D,D).

Poiché il nostro lavoro tratterà principalmente giochi a somma zero con 2 giocatori, presentiamo ora quello che può esser definito un corollario del Teorema di Nash, cioè il Teorema Minmax di Von Neumann.

	C	D
C	$-1, -1$	$-4, 0$
D	$0, -4$	$-3, -3$

Figura 2.1: Dilemma del Prigioniero.

Teorema 2. *(Teorema Minimax (von Neumann, 1928)) Un gioco a somma zero a 2 giocatori finito, descritto da una matrice dei valori di utilità, ha almeno un equilibrio di Nash in strategie miste.*

Dal nome di questo teorema prendiamo spunto per parlare di un concetto di soluzione che, nel caso di giochi a 2 giocatori a somma zero è fortemente collegato a quello di equilibrio di Nash, cioè le strategie Minmax e Maxmin.

2.3.2 Strategie MINMAX e MAXMIN

Partendo dal caso più generico, possiamo definire la maxmin-strategy dell'i-esimo giocatore, in un gioco a n giocatori e somma costante diversa da zero, una strategia mista che massimizza il peggior valore dell'utilità dell'i-esimo giocatore, nel caso in cui tutti gli altri giocatori decidano di giocatore le strategie che causano le peggiori perdite, in termini di utilità attesa, per l'i-esimo giocatore. Il valore maxmin del gioco, detto anche soglia di sicurezza, per l'i-esimo giocatore sarà quindi il valore minimo di utilità garantito da una strategia maxmin attuata dallo stesso.

Definizione 11 (Maxmin)**.** La strategia maxmin per il giocatore i è $\underset{s_i}{argmax}\ \underset{s_{-i}}{min}\ u_i(s_i, s_{-i})$, ed il valore maxmin per il giocatore i è $\max\limits_{s_i} \min\limits_{s_{-i}} u_i(s_i, s_{-i})$.

Data la definizione di cui sopra, possiamo subito intuire come questo concetto di soluzione abbia senso solo in caso di giochi in cui tutti i giocatori giocano in modo simultaneo: la strategia maxmin infatti risulta essere una scelta razionale solo nel caso in cui si voglia massimizzare la propria utilità attesa in modo conservativo, cioè senza fare nessuna ipotesi riguardo le possibili azioni, razionali o meno, degli altri giocatori.

Equivalentemente, la minmax-strategy e il valore minmax risultano essere la controparte degli stessi per Maxmin: nel semplice caso di un gioco a 2 giocatori, per esempio, la strategia minmax per il primo giocatore sarà infatti quella gli consente di minimizzare il massimo valore di utilità ottenibile dal suo avversario; questa strategia quindi ha senso solo se si considera di voler punire un altro giocatore senza badare al proprio valore di utilità.

Definizione 12 (Minmax a 2 giocatori)**.** La strategia minmax per il giocatore i contro il giocatore $-i$ è $argmin_{s_i} \max_{s_{-i}} u_{-i}(s_i, s_{-i})$, ed il valore minmax per il giocatore $-i$ è $\min_{s_i} \max_{s-i} u_{-i}(s_i, s_{-i})$.

Ovviamente, nel caso i giocatori in gioco siano più di 2, ricavare la strategia minmax dell'i-esimo giocatore contro il j-esimo è un pò più complicato in quanto ora, essendoci altri giocatori che possono influire sulle utilità del gioco, non è più garantito il fatto che l'i-esimo giocatore possa limitare al minimo il massimo valore d'utilità ottenibile dal j-esimo agente.

Tornando però al caso di nostro interesse, cioè i giochi a due giocatori, possiamo affermare 2 cose molto importanti:

- il valore minmax di un giocatore è sempre uguale al suo valore maxmin;
- dato che la strategia minmax e quella maxmin di un dato giocatore non dipendono dalle strategie del suo avversario, allora possiamo considerarli come già detto dei concetti di soluzione.

Essendo quindi dei concetti di soluzione, possiamo definire dei profili di strategie miste $s = (s_1, s_2)$ come dei profili di strategie maxmin tali che s_1 è una strategia maxmin per il primo giocatore ed s_2 è una strategia maxmin per il secondo giocatore; equivalentemente possiamo definire i profili di strategie minmax. Definiti questi profili di strategie, è possibile riformulare il Teorema di MinMax di Von Neumann come segue.

Teorema 3 (Teorema Minimax (von Neumann, 1928))**.** *In un gioco finito a due giocatori a somma zero, in ogni equilibrio di Nash ogni giocatore riceve un valore d'utilità che è equivalente sia al suo valore maxmin sia al suo valore minmax.*

Da questo teorema quindi possiamo ricavare le seguenti conclusioni nel caso di giochi a 2 giocatori a somma zero:

- per entrambi i giocatori, il proprio valore minmax è equivalente al proprio valore maxmin, e questo valore viene anche chiamato Valore del Gioco;

- per entrambi i giocatori, il set delle proprie strategie minmax è equivalente al set delle proprie strategie maxmin;

- ogni profilo di strategie maxmin, o minmax, è anche un Equilibro di Nash.

Questo dimostra il già citato collegamento tra questi 2 concetti di soluzione, infatti le strategie minmax e maxmin risultano essere un caso particolare di Equilibrio di Nash nel caso di giochi a 2 giocatori a somma zero, il ché implica anche che, sotto quest'ultime ipotesi, queste strategie possono essere computate, come gli equilibri di Nash, in tempo polinomiale.

2.3.3 Equilibrio di Nash Approssimato

Se l'equilibrio di Nash significa nessun incentivo a deviare, allora l'equilibrio di Nash approssimato indica un basso incentivo a deviare: infatti, impostando ϵ come un valore positivo solitamente abbastanza piccolo, possiamo definire un ϵ-equilibrio di Nash come un profilo di strategie miste tali che ogni giocatore può variare la sua utilità attesa fino ad un massimo di ϵ cambiando la sua attuale strategia. Questo concetto di soluzione, cioè l'Equilibrio Approssimato di Nash (ϵ-Nash), parte quindi dall'idea che ci potrebbero essere in gioco uno o più agenti che non cambierebbero una loro determinata strategia per giocare una best response se la differenza di utilità attesa tra le due fosse uguale o inferiore ad un certo valore ϵ. Si assuma che la matrice che rappresenta la funzione di reward del gioco sia normalizzata, cioè sia formata da tutti e soli valori compresi nell'intervallo $[0, 1]$.

Definizione 13. [ϵ-Nash] Si fissi un $\epsilon > 0$. Un profilo di strategie $s = (s_1, \ldots, s_n)$ è un ϵ-Equilibrio di Nash (ϵ-NE) se, per ogni agente i e per ogni strategia $s'_i \neq s_i$, si ha $u_i(s_i, s_{-i}) \geq u_i(s'_i, s_{-i}) - \epsilon$.

Per quanto sembri poco utile, l'assunzione fatta prima di questa definizione ci fornisce una garanzia sul fatto che ϵ appartenga ad un certo intervallo.

Questo concetto possiede alcune importanti caratteristiche:

- dato un gioco con le stesse ipotesi di base di esistenza dell'equilibrio di Nash, in questo esisterà sempre almeno un ϵ-equilibrio di Nash;

- per ogni Equilibrio di Nash, si può idendificare una regione di ϵ-Nash che lo circonda, per un qualsiasi valore di ϵ;
- è computazionalmente utile, infatti per identificare un ϵ-Nash non serve prendere in considerazione tutto lo spazio di gioco, ma basta considerare anche solo un insieme finito di profili di strategie miste.

Oltre a queste caratteristiche positive, però, possiede anche diversi svantaggi:

- un ϵ-Nash non è necessariamente circondato, o comunque vicino, ad altri equilibri di Nash;
- la definizione di ϵ-Nash non implica che uno o più degli altri giocatori non possa guadagnare più di ϵ applicando la propria Best Response rispetto alla strategia scelta dall'i-esimo giocatore.

Si può quindi concludere che, se si ha come obiettivo la ricerca di un punto fisso di un problema, questo concetto di soluzione può risultare meno stabile, o affidabile, rispetto ad un equilibrio di Nash per gli svantaggi appena citati, mentre se il nostro scopo è l'ottimizzazione di un singolo obiettivo di un dato problema, allora questo concetto di soluzione può risultare computazionalmente molto utile per le sue specifiche caratteristiche positive.

2.4 Giochi Stocastici

Nella vita reale possono capitare molti eventi fuori dal nostro controllo, che quindi possono causare situazioni impreviste. Alcune tipologie di giochi prevedono al loro interno situazioni impredicibili, e rappresentano questa cosa introducendo nel gioco stesso un elemento casuale, come può essere il lancio di una moneta o di un dado: questi vengono chiamati Giochi Stocastici (o Markoviani), di cui un classico esempio è il gioco del Backgammon. L'introduzione di questa nuova variabile rende più difficile la ricerca della mossa migliore da fare: in questo caso infatti, non è più possibile definire a priori dei valori di minmax esatti, ma si possono solo calcolare dei valori attesi, cioè fare la media pesata di tutti i possibili esiti relativi alla variabile casuale. I giochi stocastici possono essere visti anche come una collezione di più giochi: si può decidere infatti che ad ogni turno si giochi un particolare gioco dalla collezione in modo tale che la scelta di quest'ultimo dipenda probabilisticamente dal gioco giocato e dalle azioni fatte da tutti i giocatori al turno precedente. Fanno parte di questa vasta categoria di giochi i Markov Decision Processes (MDP), di cui parleremo approfonditamente più avanti,

ed i Giochi Ripetuti, che, da come si può facilmente intuire dal nome, sono caratterizzati dal fatto che i giocatori continuano a giocare iterativamente su un singolo gioco. Le nozioni che presenteremo in questa sezione si basano su quelle presenti in [25], [27].

Diamo ora una definizione formale di gioco stocastico.

Definizione 14. [Gioco Stocastico] Un gioco stocastico è definito dalla tupla (Q, N, A, P, R), dove:

- Q è un set finito di giochi (stati);
- N è un set finito di n giocatori;
- $A = A_1 \times \cdots \times A_n$, dove A_i è un set finito di azioni possibili per l'i-esimo giocatore;
- $P : Q \times A \times Q \longmapsto [0, 1]$ è la funzione di probabilità di transizione; più precisamente, $P(q_0, a, q_1)$ è la probabilità di transizione dallo stato q_0 allo stato q_1 dato il profilo d'azione a;
- $R = r_1, \ldots, r_n$, dove $r_i : Q \times A \longmapsto \mathbb{R}$ è una funzione di reward a valori reali per l'i-esimo giocatore.

Nella definizione appena data si suppone che lo spazio delle strategie degli agenti sia sempre uguale in tutti i giochi, e che quindi la differenza tra i diversi giochi sia solo nelle funzioni di reward. Nel caso si volesse rimuovere questa assunzione, cambierebbe solo la notazione con cui viene descritto il gioco, e non la sua difficoltà di risoluzione; se invece si rimuovessero, come poi faremo nel corso del nostro progetto di tesi, le restrizioni relative al fatto di avere gli insiemi Q e A finiti, allora si avrebbe un'effettiva complicazione nella ricerca della soluzione del gioco.

Poiché dalla precedente definizione possiamo distinguere i diversi giochi di una collezione solo dalla funzione di reward, è utile fornire anche un modo per aggregare i vari reward dei giochi in modo da avere l'idea del reward cumulativo di un giocatore dato un gioco stocastico. A tal proposito, esistono principalmente 2 metodi di aggregazione: la cosiddetta Ricompensa Media e la cosiddetta Ricompensa Futura Scontata.

Definizione 15 (Ricompensa Media)**.** Data una sequenza infinita di reward r_i^1, r_i^2, ... per l'i-esimo giocatore, la sua ricompensa media sarà

$$\lim_{k \longmapsto \infty} \sum_{j=1}^{k} \frac{r_i^{(j)}}{k}. \tag{2.2}$$

In questa definizione si può trovare subito un possibile problema di questo metodo, cioè che il limite da calcolare potrebbe non esistere: infatti, nonostante il fatto che i payoff siano limitati per ogni stato del gioco, il calcolo della loro media potrebbe portare alla creazione di cicli, e quindi non convergere.

Definizione 16 (Ricompensa Futura Scontata)**.** Data una sequenza infinita di payoff r_i^1, r_i^2, ... per l'i-esimo giocatore, ed un discount factor β tale che $0 \leq \beta \leq 1$, la ricompensa futura scontata per il giocatore i sarà

$$\sum_{j=1}^{\infty} \beta^i r_i^{(j)}. \tag{2.3}$$

Questo metodo quindi rappresenta il fatto che l'agente si preoccupa di più di quello che può accumulare nei turni più vicini temporalmente che in quelli lontani, notando anche il fatto che 1-β rappresenta la probabilità che il gioco possa terminare in un qualsiasi turno, senza che l'agente possa prevedere ciò.

Una volta definiti formalmente i giochi stocastici, si può passare alla fase di definizione di strategie ed equilibri all'interno di quest'ultimi. Per prima cosa però definiamo lo spazio di strategia di un agente.
Sia $h_t = (q_0, a_0, q_1, a_1, ..., a_{t-1}, q_t)$ lo storico di t turni di gioco relativo ad un dato gioco stocastico, cioè la sequenza alternata di stati di gioco e profili d'azione dei giocatori dal turno 0 al turno t, e sia H_t l'insieme di tutti i possibili storici di lunghezza t-turni: l'insieme delle strategie deterministiche è il prodotto Cartesiano $\prod_{t,H_t}$, il quale richiede una scelta per ogni possibile storico ad ogni possibile turno di gioco. La strategia di un agente potrebbe quindi consistere in una qualsiasi miscela di strategie deterministiche; tuttavia esiste una gerarchia ristretta di classi di strategie realmente interessanti al fine di determinare una soluzione a questi giochi.

La prima classe è quella delle strategie cosiddette Comportamentali, che hanno come requisito il fatto che la miscela di strategie per ogni storico sia scelta sempre in modo indipendente da quella degli altri possibili storici.

Definizione 17 (Strategia Comportamentale)**.** Una strategia comportamentale $s_i(h_t, a_{ij})$ restituisce la probabilità di giocare l'azione a_{ij} per lo storico h_t.

La seconda classe è quella delle strategie di Markov, che ha come prerequisito il fatto di essere una strategia Comportamentale, quindi aggiunge un nuovo requisito tale per cui la distribuzione relative alle azioni, in un dato turno t, dipenda solo dallo stato corrente.

Definizione 18 (Strategia Markoviana)**.** Una strategia Markoviana s_i è una strategia comportamentale nella quale $s_i(h_t, a_{ij}) = s_i(h'_t, a_{ij})$, se $q_t = q'_t$, dove q_t e q'_t sono rispettivamente gli stati finali di h_t e h'_t.

La terza ed ultima classe è quella delle cosiddette strategie Stazionarie, che ha come prerequisito il fatto di essere una strategia Markoviana, poi aggiunge un ulteriore requisito tale per cui viene rimossa ogni possibile dipendenda della strategia da un dato turno t.

Definizione 19 (Strategia Stazionaria)**.** Una strategia stazionaria s_i è una strategia Markoviana nella quale $s_i(h_{t_1}, a_{ij}) = s_i(h'_{t_2}, a_{ij})$, se $q_{t_1} = q'_{t_2}$, dove q_{t_1} e q'_{t_2} sono rispettivamente gli stati finali di h_{t_1} e h'_{t_2}.

Fornite le precedenti definizioni di classi di strategie, possiamo ora considerare i possibili equilibri nei giochi stocastici. Innanzitutto, se consideriamo il caso di giochi con calcolo della ricompensa futura scontata, allora possiamo dire che in quest'ultimi esiste sempre un Equilibrio di Nash: in particolare possiamo definire uno specifico profilo di strategie chiamato Equilibrio Perfetto di Markov se questo è formato da sole strategie Markoviane, e se è un Equilibrio di Nash indipendentemente dallo stato di partenza.

Teorema 4. *Ogni gioco stocastico ad n giocatori, somma costante, con ricompensa futura scontata ha un Equilibrio Perfetto di Markov.*

Passando invece al caso di giochi con calcolo della ricompensa media, le cose si complicano per il problema già citata della possibile non esistenza del limite stesso. Se però consideriamo una particolare sottoclasse di giochi, chiamata giochi stocastici irriducibili, allora non abbiamo più questo problema. Un gioco stocastico è detto irriducibile se ogni profilo di strategie dà luogo ad una catena di Markov irriducibile sull'insieme di stati del gioco, cioè ogni stato di gioco può essere raggiungo con probabilità positiva indipendentemente dalla strategia adottata: in questi giochi quindi i limiti delle medie sono tutti ben definiti, ed è possibile definire il seguente teorema.

Teorema 5. *Ogni gioco stocastico irriducibile a 2 giocatori, a somma costante, con ricompensa media ha un Equilibrio di Nash.*

Passando quindi alla computazione di questi equilibri, possiamo subito dire che non esiste una formulazione di programmazione lineare, cioè che garantisce una computazione in tempo polinomiale della soluzione, per l'intera classe dei giochi stocastici, ma solo per alcune sue sottoclassi. La prima sottoclasse è definita dall'insieme dei giochi stocastici a 2 giocatori, somma costante, con calcolo della ricompensa futura scontata e con condizione di single controller, cioè tali per cui le probabilità di transizione sono determinate sempre da un unico giocatore.

Definizione 20 (Gioco Stocastico Single-controller)**.** Un gioco stocastico è detto Single Controller se esiste un giocatore i tale che $\forall q, q' \in Q$, $\forall a \in A$, $P(q, a, q') = P(q, a', q')$, se $a_i = a'_i$.

La seconda sottoclasse per cui esiste una formulazione di programmazione lineare è quella data dall'insieme dei giochi stocastici a 2 giocatori, a somma costante, con calcolo della ricompensa futura scontata e con una coppia di condizioni cosiddetta di separable reward state independent transition (SR-SIT), cioè lo stato di gioco e il profilo delle azioni hanno effetti indipendenti sulla ricompensa ottenuta da ogni agente, e la funzione di transizione dipende solo dal profilo delle azioni.

Se il nostro problema non ricade in una di queste sottoclassi di giochi, ma comunque è della tipologia a Ricompensa Futura Scontata, si possono trovare soluzioni altrettando valide alla programmazione lineare: se siamo in un caso a somma costante, è possibile applicare una versione modificata del metodo di Newton attraverso una formulazione di programmazione non lineare al problema, mentre se siamo in caso più specifico come i giochi a somma zero, è possibile applicare un algoritmo di tipo value-iteration.

Nel caso in cui la tipologia del nostro problema preveda invece una Ricompensa Media, allora dovremmo aggiungere ulteriori restrizioni al nostro problema rispetto al caso di Ricompensa Futura Scontata se vogliamo usare la programmazione lineare come tipo di formulazione: infatti, i giochi stocastici di questo tipo devono essere a 2 giocatori, single controller e a somma zero (e non più a somma costante generale). Nel caso in cui alcune di queste condizioni vengano a mancare, in generale non si può più applicare la programmazione lineare; tuttavia se si è in presenza di un gioco stocastico Irriducibile, allora è ancora possibile trovare una soluzione grazie ad un algoritmo, con garanzia di convergenza, che è formato dalla combinazione tra un metodo di soluzione detto Policy-Iteration ed alcune tecniche di approssimazione successive.

2.5 Soluzione classica per MDP con spazio degli stati discreto

Come abbiamo già detto, gli MDP, o Markov Decision Processes, sono una particolare sottoclasse dei giochi stocastici in cui i vari giochi della collezione sono rappresentati sotto forma di stati, e i giocatori (in un MDP classico c'è solitamente un unico giocatore) si muovono da uno stato all'altro secondo modelli di transizione probabilistici legati alle azioni fatte dagli stessi giocatori. Nel caso valga la condizione di Single-controller (SC), è possibile calcolare per questi giochi una soluzione di equilibrio minmax attraverso la programmazione lineare, quindi con garanzia di risoluzione in tempi polinomiali. Le nozioni che presenteremo in questa sezione si basano su quelle presenti in [3], [5], [29].

Prima di passare alla descrizione di questo tipo di formulazione, diamo ora una definizione formale di MDP:

Definizione 21 (Markov Decision Process)**.** Un Markov Decision Process è un gioco stocastico definito dalla tupla (S, A, P, R, β) dove:

- $S = 1, \ldots, S$ è lo spazio finito degli stati del gioco;
- $A = 1, \ldots, m$ è l'insieme finito delle azioni possibili negli stati del gioco;
- $R : S \times A \longmapsto \mathbb{R}$ è il modello di Ricompensa immediato, tale che $r(s, a)$ rappresenta il Reward ottenuto nello stato s facendo l'azione a;
- $P : S \times A \times S \longmapsto [0, 1]$ è il modello di transizione, tale che $p(s', s, a)$ rappresenta la probabilità condizionata di spostarsi dallo stato s allo stato s' facendo l'azione a;
- β è il discount factor, o fattore di sconto, con $0 \leq \beta < 1$.

Dato un MDP, il nostro obiettivo sarà quindi quello di trovare la policy ottima $\Pi^* : S \longmapsto A$ che massimizza la funzione d'utilità per ogni stato del gioco. Il valore della policy ottima in ogni stato del gioco, denotato da $v(s)$, per l'MDP sopra formalizzato dovrà soddisfare la seguente equazione, conosciuta come l'Equazione dei Punti Fissi di Bellman:

$$v(s) = \max_a [r(s, a) + \beta \sum_{s' \in S} p(s', s, a) V(s')]. \tag{2.4}$$

In altri termini, data la funzione d'utilità $v(s)$, la policy ottima $\Pi^*(s)$ sarà definita dall'azione che ottimizza l'equazione di Bellman. L'equazione (2.4) rappresenta il massimo reward accumulabile dall'agente partendo da un dato stato s, sotto l'ipotesi di un orizzonte infinito di turni opportunamenti pesati dal fattore di sconto β.

I metodi per risolvere un MDP rispettando la condizione dettata dall'equazione di Bellman sono molteplici: value iteration, policy iteration e programmazione lineare sono tra quelli più conosciuti ed usati; per la nostra trattazione, come già detto, ci concentreremo soprattutto sull'ultimo di questi. Questa scelta è giustificata principalmente dalle motivazioni seguenti:

- un problema formulato in programmazione lineare produce una soluzione esatta, senza dover specificare nessun criterio di termine, come invece succede con gli altri 2 approcci citati;
- molti algoritmi che sfruttano metodi di approssimazione della funzione d'utilità degli stati, che verranno citati più avanti nel corso del nostro lavoro, sono basati su formulazione in programmazione lineare;
- la formulazione in programmazione lineare, diversamente dagli altri, offre un modo per risolvere problemi in cui la massimizzazione della ricompensa futura scontata è soggetta a vincoli addizionali sui reward attesi, come mostrato in [1] e [2].

La formulazione in programmazione lineare per risolvere un MDP classico, cioè con un unico giocatore e con conseguente condizione di SC implicita, è data dal seguente problema:

$$\min \sum_{s=1}^{S} v(s), \quad \textit{subject to}$$

$$v(s) - r(s,a) - \beta \sum_{s'=1}^{S} p(s',s,a) v(s') \geq 0, \quad \forall s \in S, \forall a \in A \tag{2.5}$$

dove i valori di $v(s)$ for ogni stato $s \in S$ sono trattati come variabili primali del problema.

Tuttavia nel caso non classico, ma di maggior interesse per il nostro progetto, in cui ci siano 2 giocatori invece che uno solo all'interno del gioco, tali che influenzano entrambi il reward in ogni stato ma di cui solo uno influenza le probabilità di transizione, non è possibile formulare il problema sotto forma di programmazione lineare come appena fatto senza aggiungere la condizione di somma zero, oltre a quella di Single Controller. Ovviamente però l'aggiunta di un secondo giocatore modifica la definizione già vista di MDP, anche

se solo in piccola parte, ed arricchisce la formulazione del problema di nuovi elementi. Se si assume, per semplicità, che gli insiemi delle azioni dei 2 giocatori siano uguali, cioè $A_1 = A_2 = A$, l'unica cosa che cambia nella definizione del problema è la funzione di Reward, che diventa $R : S \times A \times A \longmapsto \mathbb{R}$, tale che $r(s, a_1, a_2)$ rappresenta il reward immediato ottenuto per convenzione dal primo giocatore nello stato s se i 2 giocatori giocano rispettivamente le azioni a_1 ed a_2; il reward ottenuto dal secondo giocatore è facilmente ricavabile negando il valore del reward ottenuto dal primo giocatore, essendoci l'ipotesi di somma zero.

Passando invece alla formulazione del problema, è necessario descriverne prima i nuovi elementi presenti. Innanzitutto, una strategia mista per il primo giocatore è rappresentata da una funzione $f : S \times A_1 \longmapsto [0, 1]$ soggetta al vincolo di normalizzazione $\sum_{a_1} f(s, a_1) = 1$, $\forall s \in S$, in modo tale che $f(s) = [f(s, 1), \ldots, f(s, m)]$ sia una distribuzione di probabilità sullo spazio delle azioni A_1; in modo molto simile, si definisce una strategia per il secondo giocatore in un particolare stato come $g(s) = [g(s, 1), ..., g(s, m)]$. Le collezioni di strategie miste, indicizzate per stato, saranno quindi rispettivamente denotate da $f = [f(1), ..., f(S)]$ e $g = [g(1), ..., g(S)]$. Definite queste, si può formalizzare sia una matrice di probabilità indotta dalla strategia f come

$$P(f) = \sum_{a_1 \in A_1} p(s', s, a_1) f(s, a_1),$$

sia una funzione di reward indotta da entrambe le strategie f e g come

$$r(s, f(s), g(s)) = \sum_{a_1 \in A_1, a_2 \in A_2} r(s, a_1, a_2) f(s, a_1) g(s, a_2).$$

Il reward accumulabile su un orizzonte ipoteticamente infinito di turni partendo dallo stato s, dati i nuovi elementi introdotti, sarà quindi così definito:

$$v(s, f(s), g(s)) = r(s, f(s), g(s)) + \beta \sum_{s' \in S} P(s', s, a_1) v(s', f(s'), g(s'))] \quad (2.6)$$

La soluzione di questo problema, essendo a 2 giocatori, sarà trovare delle strategie f^0 e g^0 che rappresentino un equilibrio del problema, quindi tali da render sempre valida la seguente proprietà dell'Equilibrio di Nash, conosciuta anche come saddle-point condition:

$$v(f, g^0) \leq v(f^0, g^0) \leq v(f^0, g), \quad (2.7)$$

per tutte le possibili strategie miste f e g.

Possiamo ora presentare un teorema che formalizza il problema sotto forma di programmazione lineare, e che può essere trovato in [3].

Teorema 6. *Si consideri la coppia primale (P1) e duale (D1) di programmi lineari seguente:*

$$\min_{v(s)} \sum_{s=1}^{S} v(s), \quad \textit{subject to}$$

$$v(s) \geq \sum_{a_2 \in A_2} r(s, a_1, a_2) g(s, a_2) + \beta \sum_{s'=1}^{S} P(s', s, a_1) v(s'), \quad \forall s \in S, \forall a_1 \in A_1 \tag{2.8a}$$

$$\sum_{a_2 \in A_2} g(s, a_2) = 1, \quad \forall s \in S \tag{2.8b}$$

$$g(s, a_2) \geq 0, \quad \forall s \in S, \forall a_2 \in A_2 \tag{2.8c}$$

$$\max_{z(s)} \sum_{s=1}^{S} z(s), \quad \textit{subject to}$$

$$z(s) \leq \sum_{a_1 \in A_1} x(s, a_1) r(s, a_1, a_2), \quad \forall s \in S, \forall a_2 \in A_2 \tag{2.9a}$$

$$\sum_{s=1}^{S} \sum_{a_1 \in A_1} [\delta(s, s') - \beta P(s's, a_1)] x(s, a_1) = 1, \quad \forall s' \in S \tag{2.9b}$$

$$x(s, a_1) \geq 0, \quad \forall s \in S, \forall a_1 \in A_1 \tag{2.9c}$$

Sia p^* la soluzione ottima al problema primale, e d^* quella relativa al duale, e siano $x^*(s, a_1)$ i valori ottimi delle variabili $x(s, a_1)$ ottenuti dalla soluzione del problema duale. Si definisca

$$f^* = \frac{x^*(s, a_1)}{\sum_{a_1 \in A_1} x^*(s, a_1)} \tag{2.10}$$

e sia $g^*(s, a_2)$ la distribuzione ottenuta dalla soluzione ottima del problema primale.

Allora valgono le seguenti affermazioni:

- $p^* = d^*$.
- *sia $V^* = [V^*(1), ..., V^*(S)]$ la soluzione ottima al problema primale; allora $V^* = V(f^*, g^*)$.*

- $V(f^*, g^*)$ *soddisfa la saddle-point condition (2.7).*

Dimostrazione. La dimostrazione di questo teorema è presente a pag. 93 di [3]. □

Possiamo facilmente osservare che nel teorema appena esposto la terza affermazione sottolinea il fatto che le distribuzioni calcolate nel problema sono di fatto ottime per i 2 giocatori in termini di Equilibrio di Nash: quindi questo teorema ci fornisce la garanzia dell'ottimalità della soluzione trovata tramite la formulazione in programmazione lineare presentata.

2.6 Algoritmi per MDP con spazio degli stati continuo

Dopo aver descritto gli MDP nella loro versione più classica e presentato proposte per la loro risoluzione, passiamo ora a descrivere una versione diversa, probabilmente più interessante, di quest'ultimi che stà diventanto un florido campo di ricerca in questi ultimi anni, cioè i cosiddetti CMDP, cioè MDP con spazio degli stati Continuo. L'interesse verso questa particolare sottoclasse di giochi stocastici è dovuto al fatto che molte applicazioni reali lavorano su uno spazio di stati infinito, non numerabile, cioè devono gestire variabili che possono assumere valori continui in un determinato intervallo.

Il passaggio da soluzioni per problemi a stati discreti a soluzioni per problemi a stati continui non è così facile ed immediato come può sembrare: la maggior parte degli algoritmi classici, che hanno ottime performance rispetto a problemi discreti, ha infatti notevoli difficoltà se applicato a problemi continui, ed è appunto per questo motivo che non si hanno soluzioni cosiddette classiche ma solo soluzioni approssimate. Un esempio di soluzione che non può essere applicata direttamente ai CMDP è la classica formulazione in programmazione dinamica. Questo è l'approccio generalmente più usato per risolvere MDP con spazio degli stati discreto, tuttavia soffre notoriamente di un problema legato all'aumento della dimensionalità con cui si descrive il gioco da risolvere: infatti, poiché lo spazio degli stati cresce esponenzialmente con il numero di dimensioni usate per descriverlo, il numero di variabili di stato da gestire risulterà potenzialmente infinito, quindi computazionalmente non calcolabile con questo tipo di approccio. Oltre a ciò, c'è un problema che affligge in molti casi i CMDP e che non permette l'uso degli algoritmi classici: la policy ottima potrebbe non avere un supporto finito, quindi non sarebbe computabile.

Prima di passare alla descrizione di vari metodi di risoluzione per i CMDP, vediamo quali sono le principali differenze con gli MDP classici. La prima differenza sostanziale è appunto nello spazio degli stati S, che da discreto diventa continuo, quindi con un numero di elementi non numerabile; gli altri elementi della tupla (S, A, P, R, γ) con cui è stato definito nella precedente sezione un MDP rimangono invece inalterati. L'altra differenza con gli MDP classici riguarda il vincolo da soddisfare per la ricerca della soluzione ottima del gioco: il valore della policy ottima in ogni stato di un CMDP deve ora soddisfare una versione modificata dell'Equazione dei Punti Fissi di Bellman (2.4), cioè:

$$v(s) = \max_a [r(s,a) + \gamma \int_{s' \in S} p(s', s, a) v(s')]. \tag{2.11}$$

Per quanto riguarda la soluzione dei CMDP, presentiamo le 3 principali classi di metodi di risoluzione per questo tipo di problemi.

La prima classe è definita dal cosiddetto discretization approach, cioè un approccio che prima discretizza lo spazio degli stati continuo poi risolve il risultante MDP discretizzato con metodi classici. Fanno parte di questa classe, per esempio, i metodi grid-based, cioè basati su metodi di grigliatura dello spazio continuo: lo spazio degli stati da un insieme di infiniti elementi diventa in un insieme finito di punti (grid-point), quindi viene approssimato il valore della funzione d'utilità su questi punti. Nonostante l'idea della discretizzazione possa sembrare valida ed interessante, questo approccio soffre notevoli problemi legati alla relazione tra accuratezza della soluzione e problemi di dimensionalità: un elevato livello di accuretezza infatti viene pagato con una richiesta di un numero esponenziale di elementi per descrivere lo spazio degli stati del gioco, quindi si torna al problema dei metodi classici già citato in precedenza.

La seconda classe è definita dal cosiddetto discretization-free approach, e comprende tutti quei metodi che non fanno un uso esplicito della discretizzazione sullo spazio degli stati per poter calcolare la soluzione dei CMDP, ma usano metodi di approssimazione sul modello del gioco. Gli algoritmi più conosciuti appartenenti a questa classe sono i cosiddetti Uniformization methods, cioè approcci che trasformano il modello di gioco in uno più semplice attraverso passaggi di uniformazione della distribuzione delle probabibilità di transizione. Algoritmi che fanno parte di questa classe sono poi il Lazy Approximation (LA), che è l'algoritmo attualmente considerato leader per la risoluzione dei CMDP, e l'approccio cosiddetto CPH, 2 metodi che presenteremo nel dettaglio più avanti. Su questa tipologia di approccio ai CMDP non ci sono in generale particolari problemi da menzionare, in quanto la classe stessa risulta molto vasta ed eterogenea, quindi senza problemi

comuni attualmente riscontrati; questo però non significa che non ci sia nessun problema particolare relativo al singolo algoritmo, come per esempio emergerà dall'analisi che farò successivamente dei 2 algoritmi citati.

La terza ed ultima classe è definita dal cosiddetto value function approximation approach, cioè un approccio che prevede la sostituzione della funzione d'utilità originale con una sua versione approssimata, spesso molto più semplice e definita tramite alcuni parametri caratteristici, quindi risolve il problema usando alcuni metodi classici riadattati per sfruttare le caratteristiche della nuova funzione approssimata. Questa classe può essere vista come una classe a metà tra le prime 2 presentate, in quanto presenta sia fasi di discretizzazione dello spazio di gioco che fasi di approssimazione del modello della funzione d'utilità. Gli algoritmi più usati appartenenti a questa classe ricevono in ingresso il modello della funzione d'utilità, quindi ne valutano i parametri caratteristici applicando metodi come quello dei Minimi Quadrati, infine valutano iterativamente la funzione approssimata tramite programmazione dinamica su punti dello spazio degli spazi scelti tramite metodi come il MonteCarlo Sampling. Fanno parte di questa classe anche algoritmi con approcci più sperimentali chiamati l'Approximate Linear Programming (ALP) e Non-Parametric Approximate Linear Programming (NP-ALP), 2 metodi che andremo successivamente a presentare meglio nel dettaglio.
Relativamente a questa tipologia di approcci, infine, è bene ricordare che spesso gli algoritmi più usati, cioè quelli che abbinano programmazione dinamica e metodi a Minimi Quadrati con le funzioni approssimate, possono soffrire problemi di instabilità e di divergenza della soluzione del gioco, oltre a problemi intrinsechi dovuti alla difficile analisi dei parametri del modello per una buona approssimazione.

Dopo aver descritto i diversi tipi di approccio ad un problema CMDP, passiamo ora alla descrizione in dettaglio di alcuni algoritmi tra i più rilevanti e promettenti attualmente noti.

2.6.1 Lazy Approximation Algorithm

Il primo metodo di risoluzione che presentiamo è quello che viene considerato l'algoritmo di riferimento attuale per MDP con spazio degli stati continuo, e il suo nome è Lazy Approximation (LA) [30]: questo metodo è definito come discretization-free approach, in quanto riesce a gestire funzioni rappresentanti distribuzioni di probabilità di transizione continue, e può risolvere MDP aventi un orizzonte finito di mosse.

Rispetto ai classici metodi di discretizzazione in cui viene prima approssimato/discretizzato l'MDP originale quindi risolto il modello discreto ricavato

con i metodi per MDP discreti, questo approccio posticipa la fase di approssimazione del modello finché non diventa streettamente necessaria (da qui il nome Lazy, cioè pigra) e permette di controllare il tradeoff tra accuratezza e compattezza delle funzioni di utilità calcolate. Questo metodo presenta anche l'uso di 2 particolari modelli di funzioni chiamati piecewise constant (PWC), cioè funzioni costanti a tratti, e piecewise linear (PWL), cioè funzioni lineari a tratti.

Passando alla descrizione di questo approccio, si considererà il caso semplice di avere un problema definito in uno spazio degli stati monodimensionale, per facilità di esposizione.
Siano $R(s,a)$ e $P(s',s,a)$ definite come funzioni PWC; se consideriamo l'integrale presente nell'equazione di Bellman con stati continui come la convoluzione di $V(s)$ e $P(s',s,a)$, ed ipotizziamo di avere la funzione d'utilità espressa anch'essa tramite una funzione PWC ed indicizzata da n, che corrisponde al numero di iterazioni dell'algoritmo di risoluzione già calcolati, cioè $V^n(s)$, allora possiamo affermare che $V^{n+1}(s)$, cioè la funzione d'utilità dopo un'iterazione di programmazione dinamica sull'equazione suddetta, sarà una funzione del tipo PWL. Se ripetiamo lo stesso procedimento passando da $V^{n+1}(s)$ PWL a $V^{n+2}(s)$, quest'ultima sarà una funzione piecewise quadratic, cioè a tratti quadratica: l'ordine di $V(s)$ quindi cresce con n, rendendo una computazione di questo tipo in pratica insostenibile.
Per rendere quindi possibile la computazione di $V(s)$ in modo efficiente per un orizzonte molto ampio di turni sarà necessario un passaggio intermedio di approssimazione: prima di calcolare $V^{n+2}(s)$ si approssima $V^{n+1}(s)$ con una funzione $\overline{V}^{n+1}(s)$ PWC, quindi si procede all'iterazione sull'equazione di Bellman in modo da ottenere $\hat{V}^{n+2}(s)$, che è una ragionevole approssimazione di $V^{n+2}(s)$ se $\overline{V}^{n+1}(s)$ è abbastanza simile a $V^{n+1}(s)$. Per fare questo passaggio di approssimazione intermedio da una funzione PWL ad una PWC viene usato una schema di trasformazione che migliora la qualità della policy calcolata minimizzado l'errore in norma infinito tra le 2 funzioni, cioè

$$\epsilon_n = \|\hat{V}^n - \overline{V}^n\|_\infty = \sup_x |\hat{V}^n(x) - \overline{V}^n(x)|.$$

Si definisca ora anche il massimo errore di approssimazione di V^n con $\overline{V}^n$, cioè

$$\varepsilon_n = \|\overline{V}^n - V^n\|_\infty = \sup_x |\overline{V}^n(x) - V^n(x)|.$$

Date queste definizioni, e dato il fatto che sia l'operatore di max che quello di somma applicati ad un set di funzioni PWC producono entrambi un'altra funzione PWC, possiamo affermare quanto segue:

$$\varepsilon_{n+1} \leq \epsilon_{n+1} + \varepsilon_n.$$

Questo ci dice una cosa molto importante sul metodo LA: questo può controllare direttamente l'errore di approssimazione ε_{n+1} tenendo ϵ_n piccolo a piacere, facendolo specificare direttamente all'utente, cosa che non è possibile fare nel caso dei metodi di discretizzazione tradizionale.

Passiamo ora alla descrizione dello schema algoritmico usato in LA. Innanzitutto, consideriamo l'intervallo relativo allo spazio degli stati definito come $[L, U]$, con L che indica il Lower-bound, cioè il limite inferiore, e U che indica l'Upper-bound, cioè il limite superiore; prendiamo questo intervallo e lo dividiamo uniformemente in sottointervalli più piccoli, ognuno dei quali sarà approssimato da una funzione costante. Il numero totale di sottointervalli è definito come

$$p_n \;=\; \left\lceil \frac{|a|(U-L)}{2\epsilon} \right\rceil$$

ed ognuno di questi sottointervalli avrà lunghezza

$$\Delta x \;=\; \frac{U-L}{p_n}.$$

Dato questo schema, si può partizionare l'intervallo iniziale $[L, U]$ in una collezione di sottointervalli del tipo:

$$[L_1, U_1) \;=\; [L, L+\Delta x)$$

$$[L_2, U_2) \;=\; [L+\Delta x, L+2\Delta x)$$

$$\ldots$$

$$[L_{p_n}, U_{p_n}) \;=\; [L+(p_n-1)\Delta x, L+p_n\Delta x).$$

Di conseguenza, la funzione $\hat{V}^n$ relativa all'i-esimo sottointervallo $[L_i, U_i)$ potrà essere approssimata come segue:

$$\overline{V}^n(x) \;=\; a(L+(i-1/2)\Delta x)+b, \quad \textit{con}\, x \in [L_i, U_i).$$

Dato lo schema di approssimazione di cui sopra, e data una tolleranza d'errore ϵ definita direttamente dall'utente, è possibile garantire che $\epsilon_n \leq \epsilon$, cioè è possibile controllare l'errore di approssimazione in modo diretto, come già affermato in precedenza.

Effettuando infine le stesse prove sperimantali con LA e con metodi di discretizzazione classici, si vede come l'errore di approssimazione di quest'ultimi cresca molto più velocemente rispetto a LA, e come la soluzione finale prodotta sia meno accurata, meno compatta e richieda più tempo rispetto appunto a quella fornita da LA.

2.6.2 CPH Algorithm

Parlando sempre di approccio del tipo discretization-free, è stato da poco presentato un nuovo algoritmo che cerca di migliorare i risultati ottenuti con quello di Lazy Approximation: la sua sigla è CPH, e sta per "continuous (= C) state MDP through phase-type (= PH) distributions" [31]. Questo nome infatti deriva dal fatto che, invece di discretizzare la funzione di transizione o di usare una sua approssimazione con funzioni PWC, questo algoritmo utilizza distribuzioni phase-type, le quali permettono di approssimare funzioni di transizione arbitrarie usando delle catene di Markov con transizioni di tipo esponenziale; l'uso di distribuzioni esponenziali per creare i blocchi basilari per le distribuzioni Phase-type è dovuto alle loro particolari proprietà, che permettono di calcolare in modo molto efficiente le varie iterazioni di risoluzione sull'Equazione di Bellaman tramite soluzione analitica del processo di convoluzione all'interno dell'equazione stessa.

Passando alla descrizione del problema MDP risolvibile tramite CPH, si assumerà per semplicità di esposizione che lo spazio degli stati abbia un'unica dimensione continua, solitamente la durata temporale dell'azione eseguita dall'agente nello stato, il cui valore associato decresce esponenzialmente una volta che l'azione viene eseguita. Questo algoritmo quindi, rispetto ad LA, permette l'uso di variabili $t > 0$ di tipo temporale, che potremmo vedere come delle deadline per l'esecuzione di una determinata azione in un dato stato. Impostando poi il time-to-deadline come $t - t'$, si possono ridefinire la probabilità di transizione come $P(s', s, a) * p(t')$ e il reward associato all'esecuzione, con successo, dell'azione come $R(s, a) * p(t')$, con t' che indica il tempo di esecuzione effettivo di un'azione in accordo con la funzione di densità di probabilità esponziale $p(t') = \lambda e^{-\lambda t'}$. In questo caso quindi l'agente raggiungerà il suo obiettivo eseguendo l'azione tale che

$$arg \max_{a \in A} \sum_{s' \in S} P(s', s, a) \int_0^t p(t')((V_{s'}^*(t - t') + R(s, a, s'))dt')$$

Un'altra novità di questo algoritmo è data dal fatto che le funzioni d'utilità V_s^n sono del tipo piecewise Gamma (PWG), cioè funzioni Gamma a tratti, e sono rappresentate con il simbolo Υ_s; l'uso di questo tipo di funzioni sarà ora mostrato nel caso in cui si cerchi di valutare delle Policy Incondizionate, cioè tali per cui la scelta dell'azione non dipende dal tempo corrente.

Si assuma che le transizioni tra i vari stati $s_1, ..., s_n$ siano deterministiche; sia r_i il reward accumulato dall'agente nel caso si esegua a_i rispettando la deadline e si denoti con il simbolo $\diamond$ l'operazione di convoluzione, tale che $(p \diamond V_s)(t) = \int_0^t p(t-y)V_s(y)dy$, e quindi $(p \diamond r_i)(t) = \int_0^t r_i p(t-y)dy$ Si riscriva

quindi l'equazione di Bellman associata cercando di srotolarla partendo dallo stato iniziale s_1:

$$\begin{aligned} V_{s_1}^{n-1} &= p\diamond r_1 + p\diamond V_{s_2}^{n-2} \\ &= p\diamond r_1 + p\diamond p\diamond r_2 + p\diamond V_{s_3}^{n-3} \\ &= \dots \\ &= p\diamond r_1 + p\diamond p\diamond r_2 + \dots + \underbrace{p\diamond \dots \diamond p}_{n-1} \diamond r_{n-1} \end{aligned}$$

Poiché $p\diamond \dots \diamond p$ risulta essere molto simile alla funzione Gamma incompleta di Eulero, per una qualche costante c avremo che:

$$(\underbrace{p\diamond \dots \diamond p}_{n} \diamond c)(t) \;=\; c(1 - e^{-\lambda t}(1 + \frac{(\lambda t)^1}{1!} + \dots + \frac{(\lambda t)^{n-1}}{(n-1)!}))$$

Grazie a questa somiglianza possiamo riscrivere $V_{s_1}^{n-1}$ in un modo più compatto:

$$V_{s_1}^{n-1} = c_1 - e^{-\lambda t}(c_2 + c_3(\lambda t) + \dots + c_n \tfrac{(\lambda t)^{n-2}}{(n-2)!}) = \Gamma_{s_1}$$

con

$$[c_1, c_2, ..., c_n] = [\textstyle\sum_{t=1}^{n-1} r_i,\ \sum_{t=1}^{n-1} r_i,\ \sum_{t=2}^{n-1} r_i,\ \dots,\ \sum_{t=n-1}^{n-1} r_i]$$

Possiamo quindi affermare che $V_{s_1}^{n-1}$ risulta formulabile come funzione PWG con un'unica componente Γ_{s_1}; quest'ultima infine può esser salvata sotto forma di vettore del tipo:

$$\overline{\Gamma}_{s_1} \;=\; \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_n \end{pmatrix} \;=\; \begin{pmatrix} r_1 + r_2 + \dots + r_{n-1} \\ r_1 + r_2 + \dots + r_{n-1} \\ r_2 + \dots + r_{n-1} \\ \dots \\ r_{n-1} \end{pmatrix}$$

Grazie alle nozioni appena presentate è possibile creare una procedura ricorsiva molto veloce che ricava le funzioni d'utilità usando semplici trasformazioni vettoriali, di cui parleremo direttamente all'interno della descrizione specifica dell'algoritmo.

Un altro vantaggio di questa procedura è che permette di controllare l'errore di approssimazione in modo da limitarlo al di sotto di una certa soglia voluta dall'utente: questa cosa è possibile perché si può calcolare il minimo numero di iterazioni (n) dell'algoritmo tali da garantire che l'errore

di approssimazione reale sia inferiore l'errore massimo voluto (ϵ). La relazione tra questi è espressa come segue:

$$n^* \geq \log_{\frac{e^{\lambda\Delta}-1}{e^{\lambda\Delta}}} \frac{\epsilon}{R_{\max}(e^{\lambda\Delta}-1)}$$

Prima di passare al cuore dell'algoritmo però abbiamo bisogno di presentare un'ulteriore parte, più precisamente un teorema, che sarà fondamentale per la descrizione dell'algoritmo iterativo.

Teorema 7. *Se* $\overline{\Gamma} = [c_1, c_2, ..., c_n]$, *allora possiamo associare alla funzione* $p\diamond\Gamma$ *il vettore dei coefficienti* $\overline{(p\diamond\Gamma)} = [c_1, c_2, ..., c_n]$.

Possiamo ora iniziare a descrivere l'algoritmo caratterizzante l'approccio CPH. Questo inizia ricevendo in ingresso sia la struttura del problema MDP continuo sia il livello d'errore d'approssimazione ϵ tollerato dall'utente, poi lancia una funzione che calcola il valore di λ che uniformerà le transizioni con funzioni di probabilità esponenziali ed un'altra funzione che determina il valore n^* di iterazioni minime per garantire che sia rispettato il vincolo sull'errore di approssimazione.

Dopo questi passaggi, e superata una fase di preparazione di dati e template, si passa al cuore dell'algoritmo, dove è implementato il metodo di calcolo del valore delle funzioni d'utilità formulato tramite programmazione dinamica, diviso in 3 passaggi principali. Sapendo che, per costruzione prima derivata, $V_{\bar{s}}^n$ sarà composto da segmenti $\Gamma_{\bar{s},i}|_{i\in I_{\bar{s}}}$, si imposteranno i segmenti $\Gamma_{s,i}|_{i\in I_{\bar{s}}}$ relativi a V_s^{n+1} attraverso i seguenti tre passaggi:

1. per ogni intervallo, si aggiungono le funzioni d'utilità convolute future tramite la tecnica vista nel Teorema 7, tali che:

   $$\overline{\Gamma}_{s,i} \longleftarrow [\overline{\Gamma}_{\bar{s},i}[1] \,|\, \overline{\Gamma}_{\bar{s},i}]$$

   dove $[A|B]$ indica la concatenazione di A e B, e $\overline{\Gamma}_{\bar{s},i}[1]$ accede al primo coefficiente c_1 di $\overline{\Gamma}_{\bar{s},i}$.

2. Per tutti gli intervalli con $i > 1$ si aggiunge a $\overline{\Gamma}_{\bar{s},i}$ un valore di breakpoint regolarizzante per l'i-esimo intervallo; $\overline{\Gamma}_{\bar{s},i}$ sarà quindi modificata come segue:

   $$\overline{\Gamma}_{s,i} \longleftarrow \overline{\Gamma}_{s,i} - [0, constant, 0, \ldots, 0].$$

3. Infine per ogni intervallo si aggiunge $p\diamond r$; grazie al Teorema 7 avremo che $\overline{(p\diamond r)} = [r, r]$, quindi esprimeremo quanto segue:

   $$\overline{\Gamma}_{s,i} \longleftarrow \overline{\Gamma}_{s,i} + [r, r, 0, \ldots, 0].$$

Per concludere la trattazione si può dire che, oltre alle varie proprietà dell'algoritmo già presentate, le prove sperimentali mostrano che si ha un miglioramento prestazionale dei tempi di computazione pari ad un paio di ordini di grandezza rispetto al metodo di Lazy Approximation.

2.6.3 Approximate Linear Programming

Presentiamo ora un approccio algoritmico che fà parte della categoria dei metodi di value function approximation e che si chiama Approximate Linear Programming (ALP) [32]. Questo metodo permette di risolvere particolari tipologie di MDP, chiamati a fattori complessi, sia con spazio degli stati discreto sia con spazio degli stati continuo, ovviamente con un opportuno cambio di formulazione, sfruttando una combinazione lineare di funzioni definite da feature locali per modellare le funzioni d'utilità e metodi di programmazione lineare per adeguare opportunamente i coefficienti del modello.

Gli MDP a fattori complessi sono caratterizzati da uno spazio degli stati X che è definito in termini di variabili di stato $X_1, ..., X_n$: ovviamente, senza opportune parametrizzazioni di compattezza, questo spazio diventerebbe esponenziale nel numero di variabili al suo interno.

Per risolvere questo problema viene usato il seguente modello lineare per approssimare la funzione d'utilità:

$$f(x) = \sum_i w_i f_i(x_i),$$

dove w_i sono i coefficienti lineari, in numero finito, da adeguare opportunamente al modello, e f_i sono le funzioni con feature locali definite su sottospazi x_i delle variabili di stato.

Questo modello lineare permette quindi la computazione della funzione obiettivo in modo efficiente, ma il numero teorico di vincoli totali del problema da soddisfare rimane esponenziale: se consideriamo però che nella risoluzione pratica solo un sottoinsieme di questi vincoli è realmente attivo durante la computazione ed influisce direttamente sulla soluzione, ci basta trovarlo per poter risolvere il tutto.

Si assuma che le rappresentazioni a fattori complessi prevedano probabilità di transizione definite in termini di densità sui sottospazi delle variabili di stato; data questa e le nozioni precedenti, possiamo ora formulare il problema MDP a fattori complessi con stati continui come segue:

$$\min_w \sum_i w_i \int_{x_i} f_i(x_i) dx_i, \quad subject\, to$$

$$\sum_i w_i \left[f_i(x_i) - \gamma \int_{x'_i} \left(\prod_{x'_j \in x'_i} p(x'_j, x_{j,a}, a) \right) f_i(x'_i) dx'_i \right] - R(x,a) \geq 0, \quad \forall x, a$$

Questo tipo di formulazione, nonostante le premesse fatte, presenta intrinsecamente due possibili problemi: il primo riguarda gli integrali presenti, che potrebbero essere impropri e non calcolabili, mentre il secondo, già citato, riguarda la ricerca dei vincoli attivi del problema, per non avere una formulazione con infiniti vincoli da soddisfare. Per risolvere entrambi sono state identificate delle classi coniugate di modelli di transizione e funzioni di base che permettono di avere integrali propri nella formulazione e di calcolare soluzioni in forma chiusa.

Partendo dal modello di transizione, viene usato quello basato sulle densità di distribuzione β, che rappresentano le probabilità di un processo bernoulliano a posteriori dell'osservazione di $\alpha - 1$ successi e $\beta - 1$ fallimenti; questo modello viene quindi definito come segue:

$$p(x'_j, x_{j,a}, a) = Beta(x'_j, g^1_{j,a}(x_{j,a}), g^2_{j,a}(x_{j,a}))$$

con $x_{j,a}$ insieme generatore di una data variabile x_j, data l'azione a, mentre $g^1_{j,a}x_{j,a}$,$g^2_{j,a}x_{j,a} > 0$, con $x_{j,a} \in [0,1]^{|x_{j,a}|}$, rappresentano rispettivamente i parametri (α, β) del modello di transizione Beta.

Parlando invece delle funzioni di base, viene usata una forma di funzioni con feature locali data dal prodotto di potenze di funzioni, che ben si accoppia alle transizioni beta già presentate; queste funzioni saranno definite quindi come segue:

$$f_i(x_i) = \prod_{x_j \in x_i} x_j^{m_{j,i}}$$

Grazie a questi accorgimenti, si può quindi riscrivere la formulazione ALP, ora finalmente computabile, del problema MDP come segue:

$$\min_w \sum_i w_i (\frac{1}{2})^{|x_i|}, \quad subject\ to$$

$$\sum_i w_i \left[\prod_{x_j \in x_i} x_j - \gamma \prod_{x_j \in x_i} \frac{g^1_{j,a}(x_{j,a})}{g^1_{j,a}(x_{j,a}) + g^2_{j,a}(x_{j,a})} \right] - R(x,a) \geq 0, \quad \forall x, a$$

Per concludere la presentazione del metodo ALP, possiamo ora elencare i vantaggi di quest'ultimo, verificati anche nelle prove sperimentali, rispetto ai metodi di discretizzazione grid-based ed a quelli di approssimazione della funzione d'utilità classici:

- riceve vantaggi intrinsechi dati dalla struttura del processo di approssimazione impostata;
- permette di definire modelli di funzioni d'utilità non lineari e di evitare al contempo i problemi di instabilità classici associati alle tecniche di approssimazione a minimi quadrati;
- produce una soluzione più robusta rispetto ai metodi di discretizzazione grid-based, soprattutto in casi in cui l'insieme di sample usato è molto piccolo;
- il tempo di esecuzione scala meglio rispetto ai metodi di discretizzazione grid-based.

2.6.4 Non-Parametric Approximate Linear Programming

Nella categoria dei metodi di value-function approximation l'approccio ALP presentato in precedenza è stato la base per molti metodi successivi in cui si è andato man mano raffinando e migliorando l'approccio generale agli MDP con stati continui. Infatti, ALP è un metodo di approssimazione della funzione d'utilità di tipo parametrico in cui si rappresenta appunto la funzione d'utilità come combinazione lineare di feature locali, feature che però sono scelte a priori: la fase di scelta rappresenta però una difficile sfida per chi usa questo approccio.

Scegliere un buon set di feature è però fondamentale per una buona approssimazione della soluzione del problema: infatti se questo risulta troppo restrittivo, ALP è soggetto a molti bias, cioè potrebbe non riuscire a rappresentare alcune strutture fondamentali della funzione d'utilità, mentre se il set risulta troppo espressivo, ALP può presentare problemi di varianza nella soluzione, in quanto la funzione d'utilità approssimata potrebbe overfittare i dati/esempi di training. Per sistemare, o perlomeno attenuare, questi problemi si sono sperimentate successivamente alcune tecniche di Regolarizzazione e di Selezione delle feature, tuttavia queste continuano ad avere come base dei modelli parametrici, quindi non riescono a risolvere del tutto i problemi di cui sopra. La formulazione che deriva da questi modelli parametrici infine non permette di ricavare facilmente la policy ottima dal problema duale.

Una possibile soluzione a questi problemi quindi si dovrebbe basare su un approccio non parametrico: da questa idea nasce Non-Parametric Approximate Linear Programming (NP-ALP) [33], cioè un approccio di programmazione lineare completamente non parametrico che bypassa l'uso e la selezione delle feature usando solo una particolare assunzione di regolarità

sul modello. Solitamente le assunzioni di regolarità sono rappresentate in questi modelli tramite vincoli sui pesi presenti nell'approssimazione lineare della funzione d'utilità: l'assunzione che verrà usata in NP-ALP prevede che la funzione d'utilità ottimale sia Lipschitz-continua.

Affinché si possa soddisfare quest'assunzione deve essere valido il seguente vincolo, per ogni possibile coppia di stati s ed s' del gioco:

$$|V^*(s) - V^*(s')| \leq L_{V^*} d(s, s') \tag{2.15}$$

dove

$$d(s, s') = \|k(s) - k(s')\| \tag{2.16}$$

e dove k(s) rappresenta la mappatura dallo spazio degli stati ad uno spazio di vettori normalizzato ed L_V è la costante di Lipschitz associata alla funzione d'utilità.

Si denoti quindi con $\mathcal{M}_L$ l'insieme di funzioni aventi L come costante di Lipschitz, e con $\widetilde{V}$ la funzione d'utilità approssimata; la formulazione del problema MDP a stati continui sarà quindi definita come segue:

$$\min_{V(s)} \sum_s \widetilde{V}(s), \quad subject\, to$$

$$\widetilde{V}(s) \geq R(s, a) + \gamma \sum_{s' \in \widetilde{S}} P(s', s, a) \widetilde{V}(s'), \quad \forall (s \in \widetilde{S}, a) \tag{2.17a}$$

$$\widetilde{V} \in \mathcal{M}_{L_{\widetilde{V}}} \tag{2.17b}$$

con $\widetilde{V} \in \mathcal{M}_{L_{\widetilde{V}}}$ implementato come segue

$$|\widetilde{V}(s) - \widetilde{V}(s')| \leq L_{\widetilde{V}} d(s, s'), \forall (s, s').$$

Se ad una prima osservazione può sembrare che il numero di vincoli del problema, seppur campionati, sia teoricamente ingestibile, nella pratica si nota come prima cosa che tutti i vincoli di continuità coinvolgono esattamente 2 variabili alla volta, cosicché la relativa matrice di vincoli risulta molto sparsa, caratteristica che viene gestita in modo efficiente dai moderni calcolatori e permette la computazione del problema, quindi come seconda cosa che per ogni sample del problema sarà attivo uno ed uno solo dei vincoli presenti, cioè o il vincolo di Bellman o quello di Lipschitz.

In questo approccio è possibile avere nell'implementazione anche solo dei vincoli campionati in quanto, quando vogliamo calcolare l'utilità di uno stato t non presente direttamente nei vincoli suddetti, possiamo darne un valore

stimato che viene ottenuto identificando uno stato vincolato s tale che sia massimizzata la relazione seguente:

$$V(t) \geq V(s) - L_{\widetilde{V}} d(s,t)$$

Analizzando meglio i parametri della formulazione si nota che, se $L_{\widetilde{V}} = L_{V^*}$, allora la soluzione sarebbe $\widetilde{V} = V^*$; tuttavia in pratica è molto difficile che $L_{\widetilde{V}}$ sià esattamente uguale ad L_{V^*}, sia per motivi di design dell'algoritmo, in modo da evitare problemi di overfit dei dati, sia per motivi di scarsa conoscenza del valore reale di L_{V^*}.

Come tutti i metodi non parametrici, anche questo ha la sua maggior forza, come si può facilmente notare dalla formulazione, nella grande flessibilità di selezione della funzione di distanza per definire l'assunzione di regolarità, tuttavia questa risulta anche essere la sua grande debolezza, in quanto la sua scelta ha importanti effetti sui tempi di computazione della soluzione, quindi sulle performance dell'algoritmo in generale.

Per quanto riguarda il limite all'errore nella soluzione del problema, si può ricavare che in NP-ALP questo sarà:

$$\|\widetilde{V} - V^*\|_{1,\rho} \leq \epsilon + 2\frac{\epsilon_p(L_{\widetilde{V}})}{1-\gamma}. \tag{2.18}$$

Questo errore è quindi composto principalmente da 2 componenti:

$$\epsilon \leq \frac{R_{\max} - R_{\min}}{(1-\gamma)^2}(1 - \frac{L_{\widetilde{V}}}{L_{V^*}})$$

che rappresenta la componente d'errore intrinseca al problema, dovuta all'uso della funzione d'utilità approssimata, limitata da alcuni valori caratteristici del modello stesso in uso;

$$\epsilon_p(L_{\widetilde{V}}) \leq d_{\max}(L_{V^*} + L_{\widetilde{V}})$$

che rappresenta la componente d'errore relativa alla possibile violazione dei vincoli di Bellman in tutti quei stati che non fanno parte dell'insieme di sample del problema, componente limitata da $d_{\max}$ che identifica la massima distanza da uno stato non presente nell'insieme dei sample allo stato a lui più vicino che però risulti presente nell'insieme dei sample.

Per concludere la trattazione su questo approccio, elenchiamo i principali vantaggi di NP-ALP rispetto all'approccio ALP standard:

- l'implementazione di NP-ALP risulta più semplice, permette una computazione più efficiente del problema primale e soprattutto del problema duale ed ha un limite d'errore computabile più facilmente, in quanto possiamo relazionarlo direttamente alle proprietà del modello in uso;

- NP-ALP permette di trattare problemi che presentano uno spazio delle azioni molto largo, o addirittura infinito (continuo);
- essendo un approccio non parametrico, NP-ALP non presenta problemi di tuning di ρ, cioè dei pesi relativi alla rilevanza degli stati nel problema, in quanto non si hanno cambiamenti nella soluzione del problema se anche si assegnano pesi diversi agli stati
- NP-ALP assicura una soluzione limitata anche se i vincoli sono campionati, mentre in ALP la mancanza anche solo di un unico vincolo può causare, nel caso peggiore, una soluzione del programma lineare che è illimitata.

Capitolo 3

Giochi Polinomiali con Spazio delle Azioni Continuo

In questo capitolo presenteremo una classe di giochi chiamata Giochi Polinomiali: grazie a questa potremo descrivere, analizzare e risolvere problemi aventi come caratteristica comune uno spazio delle azioni di gioco continuo in un intervallo chiuso $[c, d]$. L'uso di questa particolare classe di giochi è anche giustificato dal fatto che può esser vista come un possibile collegamento tra forme di giochi finiti ed infiniti.

Entrando nel dettaglio di questo capitolo, nella Sezione 3.1 introdurremo la classe dei giochi polinomiali e mostreremo tutte le nozioni necessarie per risolvere un gioco in forma normale appartenente a questa classe.

Nella Sezione 3.2, estenderemo la nostra trattazione sui giochi polinomiali ai giochi stocastici, in particolare proponendo soluzioni per problemi relativi alla sottoclasse degli MDP aventi la proprietà di Single Controller.

Nella Sezione 3.3, invece, tratteremo una particolare versione di gioco stocastico polinomiale in cui viene valutata la soluzione di Best Response di un dato giocatore a fronte di un profilo noto di strategie miste dell'avversario.

Nella Sezione 3.4, ci concentreremo sulla classe dei giochi stocastici definita dagli MDP polinomiali con proprietà di Switching Controller, che rappresenta una superclasse degli MDP polinomiali Single Controller, e descriveremo un algoritmo, con garanzie di convergenza ed ottimalità della soluzione, che risolve problemi relativi a questa particolare classe di giochi.

Infine, per concludere questo capitolo, nella Sezione 3.5 presenteremo un'analisi teorica dell'errore massimo di approssimazione della soluzione che si ha quando si risolve un MDP con funzioni in forma polinomiale che in

realtà rappresenta l'approssimazione di un MDP con funzioni in forma non polinomiale.

3.1 Giochi Polinomiali in forma Normale

I giochi polinomiali sono un'importante sottoclasse dei giochi infiniti caratterizzata da una funzione di reward di tipo polinomiale le cui variabili sono le azioni dei giocatori. Poiché si tratta di una tipologia di giochi infiniti, queste variabili non sono altro che gli elementi di un insieme infinito e non numerabile di azioni possibili per ogni giocatore.
Questa particolare sottoclasse di giochi è ritenuta molto interessante principalmente per 2 motivi: primo perché può esser vista come un possibile collegamento tra forme di giochi finiti ed infiniti, secondo perché prevede, per sua proprietà, che esista sempre una soluzione d'equilibrio con supporto finito dimensionalmente proporzionata al grado della funzione di reward. Questa particolare proprietà è descritta e dimostrata in [4], mentre un possibile algoritmo per calcolare le strategie ottime relative alla soluzione d'equilibrio, con uno spazio delle azioni ristretto in $[0, 1]$ e $[-1, 1]$, è mostrato in [5] e [6]. L'obiettivo di questa sezione sarà quello di riprendere le nozioni trattate in [5] e [6] ed estenderle in un caso più ampio, con limiti $[c, d]$ generici per quanto riguarda lo spazio delle azioni, sfruttando tecniche di programmazione semidefinita positiva ed ottimizzazione in somma di quadrati, di cui parleremo ampiamente più avanti in questa sezione.

Iniziamo dando ora una definizione formale di gioco polinomiale.

Definizione 22 (Gioco Polinomiale)**.** Un gioco a 2 giocatori a somma zero è detto Gioco Polinomiale se la sua funzione di reward è di tipo polinomiale nella forma

$$R(a_1, a_2) = \sum_{i=0}^{n} \sum_{j=0}^{m} r_{ij} {a_1}^i {a_2}^j \tag{3.1}$$

dove $a_1 \in X$ e $a_2 \in Y$ sono rispettivamente le azioni del primo e del secondo giocatore, con X ed Y set di strategie rappresentanti sottoinsiemi limitati e compatti di spazi euclidei.

Si definiscano ora i valori reali a_1 ed a_2, appartenenti entrambi all'intervallo chiuso e limitato $[c, d]$ e rappresentanti le azioni pure di ogni giocatore. Dato che la classe in cui operiamo è formata da giochi a somma zero, possiamo tranquillamente usare il Minmax come concetto di soluzione, in quanto

questo equivale al concetto di equilibrio di Nash per questa classe di giochi. La nozione di equilibrio Minmax per questi giochi però si ottiene allargando lo spazio delle strategie da pure a miste, quindi definiamo ora le strategie miste tramite le corrispondenti misure di probabilità μ e ν sull'insieme delle strategie pure, cioè l'intervallo $[c, d]$ su cui variano a_1 ed a_2. L'uso di strategie miste rende poi necessario esprimere la funzione d'utilità sotto forma di Utilità attesa date le strategie μ e ν dei 2 giocatori; la sua forma sarà del tipo:

$$\mathbb{E}_{\mu\times\nu}[R(a_1, a_2)] = \int_c^d \int_c^d (R(a_1, a_2)\mu(a_1)\nu(a_2))da_1da_2$$

con $\mu(a_1)$ che rappresenta la probabilità, data la strategia μ, che il primo giocatore esegua l'azione pura a_1, e con $\nu(a_2)$ che rappresenta la probabilità, data la strategia ν, che il secondo giocatore esegua l'azione pura a_1. Sostituendo poi la funzione di reward generica con quella espressa in forma polinomiale, separando gli integrali con le relative variabili ed infine applicando alcune sotituzioni si otterrà la seguente espressione:

$$\mathbb{E}_{\mu\times\nu}[R(a_1, a_2)] = \sum_{i=0}^{n}\sum_{j=0}^{m} r_{ij}\mu_i\nu_j \tag{3.2}$$

con μ_i che rappresenta i momenti di ordine i della misura di probabilità $\mu(a_1)$ e ν_j che rappresenta i momenti di ordine j della misura di probabilità $\nu(a_2)$, cioè si definiscono le seguenti sostituzioni:

$$\mu_i = \int_c^d a_1^i\mu(a_1)da_1, \quad \nu_j = \int_c^d a_2^j\nu(a_2)da_2.$$

Notando poi che gli spazi dei momenti sono insiemi compatti e convessi rispettivamente in $\mathbb{R}^{n+1}$ e $\mathbb{R}^{m+1}$ e che, essendo i payoff caratterizzati da una struttura molto modulare, possiamo definire le strategie ottime di ogni giocatore rispettivamente tramite i primi m ed n momenti, relativi a μ e ν, presentiamo ora il seguente teorema che ci permette di caratterizzare la soluzione di un gioco polinomiale:

Teorema 8. *Si consideri un gioco a due giocatori a somma zero nello spazio* $[c, d] \times [c, d]$*, con funzione di reward data dall'espressione (3.1). Allora il valore del gioco è ben definito ed esistono le strategie ottime miste* μ^* ν^* *che soddisfano la saddle-point condition (2.7). Inoltre, senza perdita di generalità, i supporti delle misure ottime sono finiti, con al massimo* $\min(n, m) + 1$ *atomi.*

Sapendo quindi che il valore del gioco, che rappresenteremo con il simbolo γ, è ben definito, illustriamo ora il problema di ottimizzazione che permette di calcolarlo:

$$\min_{\gamma} \gamma, \qquad subject\,to$$

$$\mathbb{E}_\nu[R(a_1, a_2)] \leq \gamma, \quad \forall a_1 \in [c, d] \tag{3.3a}$$

$$\int_c^d \nu(a_2) da_2 = 1 \tag{3.3b}$$

Analizziamo ora i 2 vincoli che compongono il problema, in modo tale da poterlo riscrivere in una forma più conveniente per la successiva elaborazione. Nel primo vincolo possiamo applicare dei semplici passaggi in modo da poter riscrivere l'utilità attesa in termini dei primi m momenti della misura di probabilità ν come segue:

$$\mathbb{E}_\nu[R(a_1, a_2)] = \sum_{i=0}^{n} \sum_{j=0}^{m} r_{ij} \nu_j a_1^i \tag{3.4}$$

Ora l'utilità attesa è rappresentata da un un polinomio univariato nell'azione a_1 del primo giocatore, i cui coefficienti però dipendono dai momenti ν_j della strategia mista che adotterà il secondo giocatore. Così facendo, il primo vincolo può essere visto come una condizione di non negatività sull'intervallo $[c, d]$ imposta sul polinomio univariato $\gamma - \sum_{i=0}^{n} \sum_{j=0}^{m} r_{ij} \nu_j a_1^i$. Prima di passare al secondo vincolo, diamo una definizione formale di polinomio univariato, che ci sarà utile anche più avanti nella trattazione, seguito da un teorema che definisce una delle sue proprietà pirncipali:

Definizione 23 (Polinomio Univariato)**.** Un polinomio $p(x) \in \mathbb{R}$ di grado n si dice univariato se ha la seguente forma:

$$p(x) = p_n x^n + p_{n-1} x^{n-1} + \cdots + p_1 x^1 + p_0$$

dove i coefficienti p_k assumono valori reali.

Teorema 9 (Teorema Fondamentale dell'Algebra)**.** *Ogni polinomio univariato (non zero) di grado n ha esattamente n radici complesse (contate con la loro molteplicità). Inoltre si ha l'unica fattorizzazione*

$$p(x) = p_n \prod_{k=1}^{n} (x - x_k)$$

dove $x_k \in \mathbb{C}$ *sono le radici di* $p(x)$*.*

Per quanto riguarda il secondo vincolo, applichiamo un cambio di variabile, passando dalla variabile di decisione ν, che rappresenta una misura di probabilità, alla variabile $\nu_{j\,j=0}^{m}$, che rappresenta i momenti. Ora questo vincolo può esser visto come una condizione tale per cui $\nu_{j\,j=0}^{m}$ deve essere una sequenza di momenti valida per una misura di probabilità supportata nell'intervallo $[c, d]$; questa condizione così riscritta sarà espressa tramite 2 vincoli e non più uno solo.

Ora possiamo riscrivere il precedente problema di ottimizzazione nel seguente modo:

$$\min_{\gamma} \gamma, \qquad \textit{subject to}$$

$$\gamma - \sum_{i=0}^{n}\sum_{j=0}^{m} r_{ij}\nu_j a_1^i \in \mathcal{P}_n \tag{3.5a}$$

$$\nu \in \mathcal{M}_m \tag{3.5b}$$

$$\nu_0 = 1 \tag{3.5c}$$

dove $\mathcal{P}_n$ è l'insieme dei polinomi univariati di grado n non negativi nell'intervallo $[c, d]$ e $\mathcal{M}_m$ è l'insieme dei primi $m + 1$ momenti di una misura di probabilità non negativa con supporto sempre nell'intervallo $[c, d]$. In questa nuova forma il primo e il terzo vincolo equivalogono ai 2 vincoli del problema scritto precedentemente, mentre il secondo vincolo è stato aggiunto per rendere valida la sequenza di momenti introdotta con il cambio di variabile. Nonostante la riformulazione però il problema rimane in una forma astratta, cioè non è ancora possibile risolverlo direttamente; bisognerà quindi prima rappresentare i due insiemi $\mathcal{P}_n$ e $\mathcal{M}_m$ in una forma più concreta e definita, poi passare ad una formulazione direttamente computabile.

3.1.1 Definizione di Polinomi Univariati mediante Programmazione SDP

Per prima cosa analizziamo il primo vincolo: questo rappresenta una condizione di non negatività su un polinomio univariato, e per poterlo esprimere introdurremo in seguito un particolare tipo di formulazione, chiamata programmazione semidefinita positiva (SDP). Iniziamo definendo meglio la condizione di non negatività sul polinomio:

Definizione 24 (Polinomio Semidefinito Positivo o Non Negativo)**.** Un polinomio univariato $p(x)$ è Semidefinito Positivo o Non Negativo se $p(x) \geq 0$ per ogni valore reale di x.

Ora introduciamo il concetto di polinomio in Somma di Quadrati (SOS), il quale ci permetterà di collegare successivamente la non negatività del polinomio alla formulazione in programmazione semidefinita.

Definizione 25 (Polinomio in Somma di Quadrati)**.** Un polinomio univariato $p(x)$ è in Somma di Quadrati (SOS) se esistono $q_1, \ldots, q_m \in \mathbb{R}$ tali che

$$p(x) = \sum_{k=1}^{m} q_k^2(x).$$

Confrontando le 2 definizioni appena enunciate, si può subito notare come la condizione di SOS sia una condizione sufficiente per la non negatività di un polinomio; questa condizione poi diventa anche necessaria nel caso si tratti di polinomi univariati. Queste osservazioni sono quindi riunite nel seguente teorema:

Teorema 10. *Un polinomio univariato è semidefinito positivo o non negativo se e solo se è in somma di quadrati.*

Poiché, come verrà dimostrato a breve, si può verificare la condizione di SOS risolvendo un problema di programmazione semidefinita, allora si potrà anche verificare la condizione di non negatività di un polinomio univariato mendiante la stessa tecnica. Introduciamo ora alcuni elementi caratterizzanti questa formulazione. Sia $\mathcal{S}^n$ l'insieme delle matrici reali simmetriche di dimensione $n \times n$.

Definizione 26 (Matrice Semidefinita Positiva e Definita Positiva)**.** Una matrice $Q \in \mathcal{S}^n$ si dice Semidefinita Positiva se $x^T Q x \geq 0$ per ogni $x \in \mathbb{R}^n$ e si dice Definita Positiva se $x^T Q x > 0$ per ogni $x \in \begin{bmatrix} q_1(x) \\ q_2(x) \\ \vdots \\ q_m(x) \end{bmatrix}$ non nullo.

Per convenzione si denoterà con $\mathcal{S}^n_+$ l'insieme delle matrici reali, simmetriche, $n \times n$ semidefinite positive e con $\mathcal{S}^n_{++}$ l'insieme delle matrici reali, simmetriche, $n \times n$ definite positive. Date tutte queste nozioni, possiamo ora presentare il seguente lemma, che caratterizza la forte relazione tra condizione SOS e programmazione SDP:

Lemma 1. Sia $p(x)$ un polinomio univariato di grado $2d$. Allora $p(x)$ è non negativo (o in somma di quadrati) se e solo se esiste $Q \in \mathcal{S}_+^{d+1}$ che soddisfano

$$p(x) = [x]_d^T Q [x]_d.$$

La condizione appena espressa non è altro che un sistema di equazioni lineari di dimensione $2d+1$ tra gli elementi di Q ed i coefficienti di $p(x)$, cioè equivale alla seguente relazione:

$$p_i = \sum_{j+k=i} Q_{jk}, \quad con\, i = 0, \ldots, 2d. \tag{3.6}$$

Di conseguenza, il Lemma 1 può essere riformulato come segue:

Lemma 2. Un polinomio univariato $p(x) = \sum_{i=0}^{2d} p_i x_i$ è in somma di quadrati se e solo se esiste $Q \in \mathcal{S}_+^{d+1}$ che soddisfa la relazione espressa in (3.6). Questo è quindi definito come un problema di programmazione semidefinita positiva (SDP).

Prima di procedere ulteriormente, presentiamo ora 2 operatori lineari che saranno fondamentali per la formulazione del nostro problema. Il primo operatore lineare è $\mathcal{H} : \mathbb{R}^{2n-1} \to \mathcal{S}^n$, il quale riceve in ingresso un vettore e ne restituisce in uscita la matrice di Hankel associata, cioè una matrice costante lungo le antidiagonali che segue il seguente schema:

$$\mathcal{H} : \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_{2n-1} \end{bmatrix} \to \begin{bmatrix} a_1 & a_2 & \ldots & a_n \\ a_2 & a_3 & \ldots & a_{n+1} \\ \vdots & \vdots & \ddots & \vdots \\ a_n & a_{n+1} & \ldots & a_{2n-1} \end{bmatrix}$$

Il secondo operatore lineare è $\mathcal{H}^* : \mathcal{S}^n \to \mathbb{R}^{2n-1}$, il quale riceve in ingresso una matrice e restituisce in uscita un vettore i cui elementi sono le somma degli elementi lungo le antidiagonali della matrice, cioè opera secondo il seguente schema:

$$\mathcal{H}^* : \begin{bmatrix} m_{11} & m_{12} & \ldots & m_{1n} \\ m_{12} & m_{22} & \ldots & m_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ m_{1n} & m_{2n} & \ldots & m_{nn} \end{bmatrix} \to \begin{bmatrix} m_{11} \\ 2m_{12} \\ m_{22} + 2m_{13} \\ \vdots \\ m_{nn} \end{bmatrix}$$

Definiti questi 2 nuovi operatori, si può riformulare il Lemma 2 come segue:

Lemma 3. Si consideri il polinomio univariato $p(x) = \sum_{i=0}^{2d} p_i x_i$, e sia $\overline{p} = [p_0, \ldots, p_{2d}]^T$ il relativo vettore dei coefficienti. Allora $p(x)$ è in è in somma di quadrati se e solo se esiste $Q \in \mathcal{S}_+^{d+1}$, con $Q \succeq 0$ tale che

$$\overline{p} = \mathcal{H}^*(Q). \tag{3.7}$$

Dopo aver caratterizzato la condizione di non negatività di un polinomio univariato nel più ampio intervallo (-∞ , +∞), passiamo ora alla definizione di quest'ultima in un intervallo $[c, d]$ più specifico, che sarà quello considerato per tutto il resto della nostra tesi; questa caratterizzazione è espressa tramite il seguente teorema, associato spesso ai nomi di Pólya-Szegö, Fekete o Markov-Lukacs.

Teorema 11. *Siano c e d due numeri reali tali che $c < d$. Allora un polinomio univariato $p(x)$ è non negativo nell'intervallo $[c, d]$ se e solo se può essere scritto come*

$$\begin{cases} p(x) = z(x) + (x-c)(d-x)w(x), & \text{se } deg(p) \text{ è pari} \\ p(x) = (x-c)z(x) + (d-x)w(x), & \text{se } deg(p) \text{ è dispari} \end{cases} \tag{3.8}$$

dove $z(x)$ e $w(x)$ sono in SOS, e dove, nel primo caso, $deg(p) = 2d$, $deg(z) \leq 2d$ e $deg(w) \leq 2d-2$, mentre nel secondo caso $deg(p) = 2d-1$, $deg(z) \leq 2d-2$ e $deg(w) \leq 2d-2$.

Per tutto il resto della trattazione di questa tesi si farà specificatamente riferimento a polinomi univariati di grado pari.
Si definiscano ora 2 matrici di supporto L_1 ed L_2 come segue:

$$L_1 = \begin{bmatrix} I_{n\times n} \\ 0_{1\times n} \end{bmatrix} \quad , \quad L_2 = \begin{bmatrix} 0_{1\times n} \\ I_{n\times n} \end{bmatrix} \tag{3.9}$$

con $I_{n\times n}$ che rappresenta la matrice Identità di dimensioni $n \times n$.
Date le nozioni appena presentate, è ora possibile caratterizzare, con il seguente lemma, la condizione di non negatività tramite la formulazione di un problema SDP nell'intervallo specifico $[c, d]$:

Lemma 4. Si consideri il polinomio univariato $p(x) = \sum_{i=0}^{2d} p_i x_i$. Allora $p(x)$ è non negativo nell'intervallo $[c, d]$ se e solo se esistono le matrici $Z \in \mathcal{S}^{d+1}$ e

$W \in \mathcal{S}^{d+1}$, con $Z \succeq 0$, $W \succeq 0$, tali che

$$\begin{bmatrix} p_0 \\ \vdots \\ p_2 d \end{bmatrix} = \mathcal{H}^*(Z + (c+d)\frac{1}{2}(L_1 W L_2^T + L_2 W L_1^T) - (c * d) L_1 W L_1^T - L_2 W L_2^T) \tag{3.10}$$

Dimostrazione. La dimostrazione segue dalla caratterizzazione della non negatività di un polinomio univariato sull'intervallo $[c, d]$ con $deg(p)$ pari data nel Teorema 11; infatti, si ha che

$$p(x) \geq 0 \, \forall x \in [c, d] \iff p(x) = z(x) + (x - c)(d - x)w(x)$$

dove $z(x)$ e $w(x)$ sono SOS. Applicando semplicemente quanto enunciato nel Lemma 3 otteniamo la condizione richiesta. □

3.1.2 Definzione dei Momenti mediante Programmazione SDP

Ora diamo una definizione più precisa dei momenti, che erano stati prima introdotti nella notazione relativa alla soluzione di un gioco in forma polinomiale.

Definizione 27 (Momento)**.** Sia X una variabile casuale a valori reali. Il momento di ordine k μ_k è la media delle potenze k-esime di X, cioè

$$\mu_k := \mathbb{E}[X^k] = \int x^k dx \tag{3.11}$$

Si definisca quindi $\overline{\mu} = [\mu_0, \ldots, \mu_m]$ come vettore di momenti in $\mathbb{R}^{m+1}$: questo vettore è una sequanza valida di momenti di lunghezza $m+1$ se corrisponde alla sequenza dei primi $m+1$ momenti di una misura non negativa μ supportata sull'insieme delle azioni A. Per quanto riguarda μ, una misura non negativa è una misura di probabilità se il suo momento di ordine zero è uguale ad 1. Infine si denoti con $\mathcal{M}(A)$ lo spazio dei momenti supportato sull'insieme delle azioni A, e con $\mathcal{M}_m(A)$ l'insieme delle sequenze di momenti di lunghezza $m+1$ corrispondenti a misure di probabilità valide. Queste nozioni appena presentate rappresentano l'insieme dei vincoli definiti precedentemente nella soluzione del problema polinomiale, tuttavia dobbiamo ora riscriverli in modo che possano esser computati.

Iniziamo l'analisi dei vincoli partendo da quello più semplice, cioè quello necessario per definire μ come misura di probabilità valida, indicato in (3.5c). Si definisca $e_1 \in \mathbb{R}^{m+1}$ come vettore di lunghezza $m+1$ composto dal primo elemento che vale 1, e da tutti gli altri che valgono 0; il vincolo (3.5c) può essere ora riscritto come

$$e_1^T[\mu_0, \ldots, \mu_m] \;=\; 1. \tag{3.12}$$

Per quanto riguarda invece il vincolo (3.5b), l'analisi richiede alcuni passaggi preliminari prima di presentare la nuova formulazione. Poiché per definizione μ è una misura non negativa, dovrà valere $\mu_k \geq 0$ per ogni valore di k, con $0 \leq k \leq m+1$. Per ricavare questa condizione in modo ricorsivo, si può sfruttare la relazione tra il primo ed il secondo momento di una variabile casuale X e la sua varianza. Dato che la varianza è sempre positiva, cioè $var(X) \geq 0$ per ogni valore di X, e che è definita come $var(X) = \mathbb{E}[X^2] - \mathbb{E}[X]^2 = \mu_2 - \mu_1^2$, allora si potrà scrivere una condizione di non negatività del tipo $\mu_2 - \mu_1^2 \geq 0$. Questa disuguaglianza può essere quindi riscritta in modo più generale come segue:

$$0 \;\leq\; \mathbb{E}[(a+bX)^2] \;=\; a^2 + 2ab\mathbb{E}[X] + b^2\mathbb{E}[X^2] \;=\; \begin{bmatrix} a \\ b \end{bmatrix}^T \begin{bmatrix} \mu_0 & \mu_1 \\ \mu_1 & \mu_2 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix}$$

con a,b $\in \mathbb{R}$. Successivamente verrà usato l'operatore $\mathcal{H}$ per semplificare la notazione relativa alle matrici dei momenti. Così riscritta, questa disuguaglianza implica che la matrice dei momenti al suo interno deve essere semidefinita positiva. Per estendere in modo sistematico la procedura appena presentata ai momenti di grado superiore a 2, si può considerere la seguente disuguaglianza:

$$\mathbb{E}[(a_0 + a_1 x + a_2 x^2 + \cdots + a_m x^m)^2] \;\geq\; 0.$$

Date queste nozioni preliminari, possiamo ora passare alla caratterizzazione dei momenti su un'intervallo generico $[c, d]$ presentando i 2 lemmi che formano il cosiddetto Hausdorff moment problem, mostrato in [7].

Lemma 5. I valori $\mu_0, \ldots, \mu_{2m}$ sono i momenti algebrici di una misura positiva supportata su un intervallo finito $[c, d]$ se e solo se le matrici A e B, definite come

$$A \;=\; \mathcal{H}([\mu_0, \ldots, \mu_{2m}]^T)$$

$$B \;=\; (c{+}d)\mathcal{H}([\mu_1, \ldots, \mu_{2m-1}]^T) - (a{*}b)\mathcal{H}([\mu_0, \ldots, \mu_{2m-2}]^T) - \mathcal{H}([\mu_2, \ldots, \mu_{2m}]^T)$$

sono semidefinite positive.

Lemma 6. I valori $\mu_0, \ldots, \mu_{2m+1}$ sono i momenti algebrici di una misura positiva supportata su un intervallo finito $[c, d]$ se e solo se le matrici A e B, definite come

$$A = -c * H([\mu_0, \ldots, \mu_{2m}]^T) + H([\mu_1, \ldots, \mu_{2m+1}]^T) \tag{3.13}$$

$$B = d * H([\mu_0, \ldots, \mu_{2m}]^T) - H([\mu_1, \ldots, \mu_{2m+1}]^T) \tag{3.14}$$

sono semidefinite positive.

Di questi 2 lemmi, noi useremo il secondo, cioè quello che lavora con una sequenza di momenti di lunghezza $m + 2$, cioè presenta un momento in più rispetto alla nostra formulazione attuale. Il calcolo del valore di questo momento è reso necessario dal fatto che la fase finale di ricostruzione dei supporti delle strategie dei giocatori e dei relativi pesi, che presenteremo alla fine di questa sezione, richiede una sequenza valida di momenti equivalente ad un vettore di lunghezza $m + 2$.

Ora possiamo finalmente dare una caratterizzazione computabile di $\mathcal{M}([c, d])$ e $\mathcal{M}_{m+1}([c, d])$:

Lemma 7. Il vettore $\mu = [\mu_0, \ldots, \mu_m + 1]^T$ è una sequenza valida di momenti per una misura di probabilità nell'intervallo finito $[c, d]$ se e solo se

$$e_1^T[\mu_0, \ldots, \mu_m] = 1. \tag{3.15a}$$

$$-c * \mathcal{H}([\mu_0, \ldots, \mu_m]^T) + \mathcal{H}([\mu_1, \ldots, \mu_{m+1}]^T) \succeq 0 \tag{3.15b}$$

$$d * \mathcal{H}([\mu_0, \ldots, \mu_m]^T) - \mathcal{H}([\mu_1, \ldots, \mu_{m+1}]^T) \succeq 0 \tag{3.15c}$$

3.1.3 Risoluzione di un Gioco Polinomiale mediante SDP

Grazie alle nozioni ed alle caratterizzazioni date nelle precedenti sezioni, è ora possibile riformulare il problema (3.5) mediante programmazione semidefinita positiva (SDP), cioè è possibile rappresentarlo in una forma risolvibile e non più astratta. Per semplificare la formulazione del problema, che qui sotto andremo ad esporre nella sua interezza, si denoti con $R \in \mathbb{R}^{(n+1)\times(m+1)}$ la matrice che contiene i coefficienti r_{ij} del polinomio $R(a_1, a_2)$, e con $\nu \in \mathbb{R}^{m+1}$ il vettore dei primi $m+1$ momenti di una misura non negativa ν. Si definisca poi il polinomio $t(a_1) = \gamma - \sum_{i=0}^{n} \sum_{j=0}^{m} r_{ij}\nu_j a_1^i$, quindi si denoti con $\mathbf{t}$ il relativo vettore dei coefficienti, tale per cui $\mathbf{t} = \gamma e_1 - R\nu$. Sfruttando il

Lemma 4, si può esprimere la non negatività del polinomio univariato $t(a_1)$ sull'intervallo $[c, d]$ tramite il vincolo

$$\mathbf{t} = \gamma e_1 - R\nu = \mathcal{H}^*(Z + (c+d)\frac{1}{2}(L_1 W L_2^T + L_2 W L_1^T) - (c*d)L_1 W L_1^T - L_2 W L_2^T). \tag{3.16}$$

con le matrici $Z \in \mathcal{S}^{n+1}$ e $W \in \mathcal{S}^n$ semidefinite positive. Grazie a quest'ultima rifinitura della condizione sul polinomio univariato, si può ora presentare il problema di ottimizzazione polinomiale astratto (3.5) sull'intervallo finito $[c, d]$ riformulato come un problema SDP:

$$\min_{\gamma} \gamma,$$

subject to

$$\mathcal{H}^*(Z + (c+d)\frac{1}{2}(L_1 W L_2^T + L_2 W L_1^T) - (c*d)L_1 W L_1^T - L_2 W L_2^T) =$$

$$= \gamma e_1 - R\nu \tag{3.17a}$$

$$e_1^T \nu = 1 \tag{3.17b}$$

$$-c * \mathcal{H}([\nu_0, \ldots, \nu_m]^T) + \mathcal{H}([\nu_1, \ldots, \nu_{m+1}]^T) \succeq 0 \tag{3.17c}$$

$$d * \mathcal{H}([\nu_0, \ldots, \nu_m]^T) - \mathcal{H}([\nu_1, \ldots, \nu_{m+1}]^T) \succeq 0 \tag{3.17d}$$

$$Z, W \succeq 0 \tag{3.17e}$$

La soluzione di questo problema ci restituisce il valore del gioco γ, ed anche i primi $m + 2$ momenti delle misure di probabilità da cui poi ricostruiremo, a fine sezione, le strategie ottime relative al giocatore 2.

Ora ci manca solo ricavare le misure di probabilità relative alle strategie ottime del giocatore 1 per avere una soluzione completa del gioco polinomiale. Per fare ciò, iniziamo ricordardo che nei giochi a 2 giocatori a somma zero c'è una relazione diretta tra i ruoli degli stessi giocatori e le proprietà di dualità convessa del problema di ottimizzazione primale; il corrispondente problema duale infatti si ricava semplicemente scambiando il ruolo dei 2 giocatori. Si definisca ora con $\mu \in \mathbb{R}^{n+1}$ il vettore dei primi $n + 1$ momenti di una misura non negativa μ, e con $e_2 \in \mathbb{R}^{n+1}$ un vettore di lunghezza $n + 1$ composto dal primo elemento che vale 1 e da tutti gli altri che valgono 0. Infine si definiscano le matrici $J \in \mathcal{S}^{n+1}$ e $K \in \mathcal{S}^n$. Data la precedente osservazione e date queste definizioni, possiamo formulare il problema Duale relativamente all'SDP (3.17), che ci permette di ottenere le misure di probabilità relative alle strategie ottime del giocatore 1:

$$\max_{\gamma} \gamma,$$

subject to

$$\mathcal{H}^*(J + (c+d)\frac{1}{2}(L_1 K L_2^T + L_2 K L_1^T) - (c*d) L_1 K L_1^T - L_2 K L_2^T) =$$

$$= R^T \mu - \gamma e_2 \tag{3.18a}$$

$$e_2^T \mu = 1. \tag{3.18b}$$

$$-c * \mathcal{H}([\mu_0, \ldots, \mu_n]^T) + \mathcal{H}([\mu_1, \ldots, \mu_{n+1}]^T) \succeq 0 \tag{3.18c}$$

$$d * \mathcal{H}([\mu_0, \ldots, \mu_n]^T) - \mathcal{H}([\mu_1, \ldots, \mu_{n+1}]^T) \succeq 0 \tag{3.18d}$$

$$J, K \succeq 0 \tag{3.18e}$$

Possiamo ora affermare che, risolvendo la coppia di problemi primale (3.17) e duale (3.18), è possibile ottenere sia il valore del gioco che i momenti delle strategie ottime di entrambi i giocatori. Quindi questa coppia di problemi SDP sopra descritti fornisce una naturale generalizzazione della soluzione di giochi finiti a somma zero ottenibile tramite la classica formulazione in programmazione lineare, condividendone tutte le fondamentali proprietà di dualità.

3.1.4 Ricostruzione delle strategie miste dai momenti

Per concludere questa sezione, presentiamo ora il procedimento usato per ricavare una misura univariata atomica dato un determinato insieme di momenti, che nel nostro caso vuol dire ricavare le strategie ottime dei 2 giocatori dati i nostri 2 vettori di momenti μ e ν. Questo procedimento può essere trovato nella sua forma standard in [8] e [9]. Si definisca l'insieme dei momenti $(\mu_0, \mu_1, \ldots, \mu_{2n-1})$ relativi ad una misura non negativa supportata in $\mathbb{R}$. Quest'ultima sarà discreta, cioè formata da un numero finito di elementi, e caratterizzata dalla forma $\sum_{i=1}^{n} w_i \delta(x - x_i)$, con $w_i = Prob(x = a_i)$ per ogni i. Si imposti quindi il seguente sistema lineare:

$$\begin{bmatrix} \mu_0 & \mu_1 & \ldots & \mu_{n-1} \\ \mu_1 & \mu_2 & \ldots & \mu_n \\ \vdots & \vdots & \ddots & \vdots \\ \mu_{n-1} & \mu_n & \ldots & \mu_{2n-2} \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-1} \end{bmatrix} = - \begin{bmatrix} \mu_n \\ \mu_{n+1} \\ \vdots \\ \mu_{2n-1} \end{bmatrix} \tag{3.19}$$

E' facilmente osservabile come la matrice di Hankel presente nel sistema possa essere equivalente alle matrici dei momenti presenti nei problemi primale

e duale, cioè $H([\mu_0, \ldots, \mu_n]^T)$ oppure $H([\nu_0, \ldots, \nu_m]^T)$. Poiché la matrice di Hankel che useremo nel sistema sarà semidefinita positiva per costruzione, la soluzione di questo sistema lineare, cioè il vettore $[c_0, \ldots, c_{n-1}]$, è unica. Sfruttiamo ora quest'ultimo come vettore dei coefficienti del seguente polinomio univariato:

$$x^n + c_{n-1}x^{n-1} + \cdots + c_1 x + c_0 \;=\; 0 \tag{3.20}$$

Le radici x_i di questo polinomio saranno tutte reali e distinte, e corrisponderanno ai punti di supporto della misura discreta sopra descritta; nel nostro caso equivarranno ai supporti delle strategie ottime dei 2 giocatori. Per ottenere infine i pesi w_i relativi ai vari supporti, cioè la probabilità con cui ogni supporto verrà scelto da un determinato giocatore, si risolverà il seguente sistema, detto Sistema non Singolare di Vandermonde:

$$\sum_{i=1}^{n} w_i x_i^j \;=\; \mu_j, \quad \textit{con } 0 \leq j \leq n-1. \tag{3.21}$$

3.2 MDP Polinomiali Single Controller

Nella Sezione 2.5 è stata presentata una particolare sottoclasse dei giochi stocastici, cioè i Markov Decision Processes (MDP), e ne è stata proposta la soluzione tramite programmazione lineare nel caso valga la condizione di Single Controller. In questa sezione riprenderemo la trattazione di questi giochi per analizzare il caso in cui lo spazio delle azioni del gioco passi da un numero finito ad un numero infinito e non numerabile di elementi in un intervallo continuo $[c, d]$. Per fare ciò, partiremo dal lavoro presentato nella sezione precedente riguardante i giochi polinomiali in forma normale per estenderlo agli MDP, rappresentando ogni stato del gioco con una funzione di Reward polinomiale. I risultati di questa sezione estendono ad un intervallo generico continuo delle azioni $[c, d]$ i risultati di analisi precedenti trattati in [6] e [10].

3.2.1 Presentazione del problema astratto

Per iniziare questa trattazione, partiamo dalla descrizione del problema, recuperando alcune nozioni già citate in altre sezioni.
Si consideri un MDP come quello definito nella Definizione 21 in cui valga la condizione di Single Controller, definita in Definizione 20 e in cui siano presenti nel gioco $N = 2$ giocatori, come nel caso analizzato in Sezione 2.5. Si mantenga lo spazio degli stati, S, come un insieme finito di elementi s, mentre si consideri lo spazio delle azioni A di gioco come un insieme infinito di elementi in un determinato intervallo finito, cioè $A = [c, d] \in \mathbb{R}$; per semplicità di trattazione si assuma che $A = A_1 = A_2$, e che lo spazio delle azioni sia uguale per ogni stato del gioco. Si consideri la funzione di reward R a coefficienti reali di tipo polinomiale nelle variabili a_1 ed a_2 come quella definita nell'espressione (3.1), e si consideri il fatto che il gioco sia a somma zero. Infine si consideri la probabilità di transizione $p(s', s, a_1)$ a coefficienti reali di tipo polinomiale nella sola variabile a_1, quindi indipendente dall'azione a_2 del secondo giocatore in quanto vale la proprietà già citata di Single Controller, definita come segue:

$$p(s', s, a_1) = \sum_{i=0}^{d_{ss'}} p_i(s, s') a_1^i$$

Analizzando questa particolare tipologia di giochi si può osservare che, essendo il processo potenzialmente infinito in termini di orizzonte decisionale, risulta una scelta assulutamente naturale quella di concentrarsi soprattutto sulla ricerca delle strategie stazionarie dei diversi giocatori, strategie definite nella Definizione 19 e che prevedono la dipendenza delle stesse solo dallo

stato del gioco e non dal tempo. La ricerca di queste specifiche strategie è supportato dal fatto che in [11] viene mostrato come esistano sempre equilibri stazionari per giochi stocastici a 2 giocatori, a somma zero, con spazio delle azioni e degli stati finito; in [10] invece viene provata la loro esistenza ed unicità, indipendentemente dalla condizione di Single Controller, anche in giochi stocastici in cui lo spazio delle azioni è infinito e la funzione di reward è polinomiale.

Parlando sempre di strategie, risulterà naturale anche considerare l'insieme delle strategie miste dei giocatori, in modo da potersi ricondurre alla nozione di equilibrio minmax. A tal proposito, definiamo una strategia mista per il primo giocatore come l'insieme finito delle misure di probabilità $\mu = [\mu(1), \ldots, \mu(\mathrm{S})]$ supportato in A_1. In modo analogo si definisca una strategia mista per il secondo giocatore come $\nu = [\nu(1), \ldots, \nu(\mathrm{S})]$ supportato in A_2.

Per chiarire la notazione appena usata e che verrà mantenuta per il resto della tesi, si noti che le lettere greche $\mu\ \nu\ \epsilon\ \varepsilon$ verranno utilizzate per indicare le misure di probabilità, i loro pedici indicherranno l'ordine del momento a cui si riferiscono, gli indici nelle parentesi che le seguono denoteranno lo stato di gioco a cui si riferiscono, e un'eventuale linea sopra queste indicherà che non si tratta di un unico momento ma di una sequenza di momenti, definita in lunghezza dal contesto in cui verrà usata; per esempio, $\mu_j(\mathrm{i})$ indicherà il momento j-esimo relativo la strategia mista adottata dal primo giocatore quando si trova nell'i-esimo stato di gioco, mentre $\overline{\mu}(\mathrm{i}) = [\mu_0(i), \ldots, \mu_n(i)]$.

Definite quindi le strategie miste dei 2 giocatori, μ e ν, si possono riformulare le nozioni di funzione di Reward R e di probabilità di transizione P. La prima, nel caso si consideri il primo giocatore nello stato di gioco s, sarà una funzione del tipo:

$$r(s, \mu(s), \nu(s)) \ = \ \int_{A_1} \int_{A_2} r(s, a_1, a_2) d\mu(s) d\nu(s).$$

La seconda invece, essendo dipendente solo dall'azione del primo giocatore, sarà una matrice di probabilità di transizione $P(\mu)$ definita come:

$$P_{ss'}(\mu) \ = \ \int_{A_S} p(s', s, a_1) d\mu(s).$$

Date queste nozioni, si può scrivere il reward $v_\beta(s, \mu(s), \nu(s))$ accumulabile in un orizzonte infinito di turni partendo da un dato stato s, in modo simile a quanto fatto nell'equazione (2.6), come segue:

$$v_\beta(s, \mu(s), \nu(s)) \ = \ r(s, \mu(s), \nu(s)) + \beta \sum_{s' \in S} \Big(\int_{A_1} p(s', s, a_1) d\mu(s) \Big) v_\beta(s', \mu(s'), \nu(s')) \tag{3.22}$$

Grazie a questa formulazione, possiamo ora definire il concetto di strategia d'equilibrio (stazionario) per la sottoclasse di giochi che stiamo trattando, a cui si collega anche la definizione di vettore dei valori del gioco:

Definizione 28 (Strategie di Equilibrio e Vettore dei Valori)**.** Due vettori di strategie miste (indicizzati dallo stato) μ^0 e ν^0 che soddisfano la saddle-point condition definita dall'equazione (2.7) per ogni vettore di strategie miste μ e ν sono chiamati Strategie di Equilibrio. Il corrispondente vettore $v_\beta(\mu^0, \nu^0)$ è chiamato Vettore dei Valori del gioco.

Sempre grazie ai risultati delle ricerche presentate in [10], sappiamo che, in caso di giochi stocastici a 2 giocatori, a somma zero, con spazio degli stati finito, spazio delle azioni infinito e funzione di reward polinomiale, il vettore dei valori associato alla strategia stazionaria d'equilibrio esiste sempre ed è unico.

Sapendo quindi che, per i giochi che stiamo analizzando, esiste sempre un'unica soluzione d'equilibrio, a cui è sempre associato un vettore di valori anch'esso unico, possiamo passare alla presentazione formale del problema: la formulazione riprenderà quella già vista nei problemi primali P1 (2.8) e D1 (2.9) estendendola al caso in cui la funzione di reward nei diversi stati sia di tipo polinomiale, come quella definita dall'equazione (3.1), e che lo spazio delle strategie sia infinito e non numerabile, sfruttando le nozioni presentate nella sezione precedente riguardante i giochi polinomiali. Il nuovo problema, che può essere visto come una generalizzazione di P1 (2.8), è formalizzato come segue:

$$\min_{v(s)} \sum_{s=1}^{S} v(s), \quad \text{subject to}$$

$$v(s) \geq \int_{a_2 \in A_2} r(s, a_1, a_2)\nu(s, a_2)da_2 + \beta \sum_{s'=1}^{S} p(s', s, a_1)v(s'), \quad \forall s \in S, \forall a_1 \in A_1 \tag{3.23}$$

ν(s) è una misura di probabilità supportata su A_2 per ogni $s \in S$

Sapendo che, fissando i valori del vettore ν(s), $P(s', s, a_1)$ diventa sostanzialmente un polinomio univariato nella variabile a_1, se mostriamo che anche la funzione di reward R lo diventa, potremo riscrivere il precedente problema in funzione non più delle misure di probabilità ν, ma direttamente in funzione dei momenti ν_j. I passaggi, molto semplici, di questa trasformazione sono i

seguenti:

$$\begin{aligned}\int_{a_2\in A_2} r(s,a_1,a_2)\nu(s,a_2)da_2 &= \int_{a_2\in A_2}\sum_{i=0}^{n_s}\sum_{j=0}^{m_s} r_{ij}(s){a_1}^i{a_2}^j\nu(s,a_2)da_2 \\ &= \sum_{i=0}^{n_s}\sum_{j=0}^{m_s} r_{ij}(s){a_1}^i\int_{a_2\in A_2}{a_2}^j\nu(s,a_2)da_2 \\ &= \sum_{i=0}^{n_s}\sum_{j=0}^{m_s} r_{ij}(s)\nu_j(s){a_1}^i\end{aligned}$$

Ora possiamo riscrivere il precedente problema(3.23) come un nuovo problema (PP1) in funzione dei momenti:

$$\min_{v(s)} \textstyle\sum_{s=1}^{S} v(s), \quad \text{subject to}$$

$$v(s) - \sum_{i=0}^{n_s}\sum_{j=0}^{m_s} r_{ij}(s)\nu_j(s){a_1}^i - \beta\sum_{s'=1}^{S} P(s',s,a_1)v(s') \in \mathcal{P}(A_1), \quad \forall s\in S \tag{3.24a}$$

$$\overline{\nu}(s) \in \mathcal{M}(A_2), \quad \forall s\in S \tag{3.24b}$$

$$\nu_0(s) = 1, \quad \forall s\in S \tag{3.24c}$$

Leggendo questa nuova formulazione, si nota subito la forte somiglianza con il problema di risoluzione di un gioco polinomiale (3.5) descritto nella sezione precedente; come quest'ultimo infatti, il problema appena presentato è in funzione dei momenti delle strategie, e risulta formulato in modo non direttamente risolvibile. Per renderlo quindi computazionalmente risolvibile, occorrerà dare nuova formulazione agli insiemi $\mathcal{P}(A_1)$ e $\mathcal{M}(A_2)$, in modo molto simile a quanto fatto per i giochi polinomiali.

3.2.2 Risoluzione del problema tramite programmazione SDP

Partiamo dalla riformulazione dell'insieme $\mathcal{P}(A_1)$, presente nel vincolo (3.24a): quest'ultimo rappresenta, per ogni stato s del gioco, una disuguaglianza polinomiale nella variabile a_1. Fissando quindi lo stato s, si può considerare il polinomio univariato:

$$t_s(a_1) \;=\; v(s) - \sum_{i=0}^{n_s}\sum_{j=0}^{m_s} r_{ij}(s)\nu_j(s){a_1}^i - \beta\sum_{s'=1}^{S} P(s',s,a_1)v(s').$$

Di questo polinomio si vorranno esplicitare i coefficienti, in modo tale da poter riformulare il problema in termini di formulazione semidefinita positiva. Si definisca con d_s il grado del polinomio per lo stato s, e con $[a_1]_{d_s} = [1, a_1, a_1^2, \ldots, a_1^{d_s}]^T$ il vettore delle azioni nello stato s, indicizzato dal grado. Si definisca poi con $R(s)$ la matrice dei coefficienti del polinomio $r(s, a_1, a_2)$, con $Q(s)$ la matrice dei coefficienti del polinomio $p(s', s, a_1)$, e con $\overline{\nu}(s) \in \mathbb{R}^{d_s+1}$ il vettore dei primi d_s+1 momenti della misura di probabilità $\nu(s)$. Grazie alle nozioni appena introdotte, riformularemo le varie parti del vincolo (3.24a) in forma vettoriale.

Partendo dal primo termine del polinomio, cioè $v(s)$, si consideri $\overline{v} = [v(1), \ldots, v(S)]^T$, cioè il vettore dei valori del gioco. Per quanto riguarda invece la funzione di reward, possiamo riscriverla in forma vettoriale come segue:

$$\sum_{i=0}^{n_s} \sum_{j=0}^{m_s} r_{ij}(s) \nu_j(s) {a_1}^i \; = \; \overline{\nu}(s)^T R(s)^T [a_1]_{d_s}. \tag{3.25}$$

Infine, per quanto riguarda la probabilità di transizione tra i vari stati del gioco, possiamo riscriverla nella seguente forma:

$$\sum_{s'=1}^{S} P(s', s, a_1) v(s') \; = \; \overline{v}^T Q(s)^T [a_1]_{d_s}. \tag{3.26}$$

Data questa nuova formulazione vettoriale delle parti che compongono il vincolo (3.24a), e riprendendo quanto detto nel Lemma 4, si può esprimere la non negatività del polinomio univariato $t_s(a_1)$ sull'intervallo $[c, d]$ tramite il vincolo

$$\mathcal{H}^*(Z(s)+(c+d)\frac{1}{2}(L_1 W(s) L_2^T + L_2 W(s) L_1^T) - (c*d) L_1 W(s) L_1^T - L_2 W(s) L_2^T) \; =$$

$$= \; E(s)\overline{v} - R(s)\overline{\nu}(s) - \beta Q(s)\overline{v}. \tag{3.27}$$

con le matrici $Z(s) \in \mathcal{S}^{d_s+1}$ e $W(s) \in \mathcal{S}^{d_s}$ semidefinite positive, e con $E(s) \in \mathbb{R}^{d_s \times S}$ che rappresenta una matrice formata da tutti zeri eccetto un 1 in posizione $(1, s)$.

Per quanto riguarda invece la riformulazione dell'insieme $\mathcal{M}(A_2)$ presente nei vincoli (3.24b) e (3.24c) riguardante la validità dei momenti relativi alle misure di probabilità, si consideri il Lemma 6 presentato nella precedente sezione. Si fissi quindi uno stato $s \in S$: grazie al lemma appena indicato, possiamo esprimere i 2 vincoli (3.24b) e (3.24c) nell'intervallo delle azioni $[c, d]$ tramite formulazione in programmazione semidefinita positiva come segue:

$$e_1^T [\nu_0(s), \ldots, \nu_m(s)] \; = \; 1. \tag{3.28a}$$

$$-c * \mathcal{H}([\nu_0(s), \ldots, \nu_m(s)]^T) + \mathcal{H}([\nu_1(s), \ldots, \nu_{m+1}(s)]^T) \succeq 0 \quad (3.28b)$$

$$d * \mathcal{H}([\nu_0(s), \ldots, \nu_m(s)]^T) - \mathcal{H}([\nu_1(s), \ldots, \nu_{m+1}(s)]^T) \succeq 0 \quad (3.28c)$$

Ora, grazie alle 2 formulazioni (3.27)(3.28) appena definite, possiamo riformulare il problema di ottimizzazione polinomiale astratto (3.24) in modo più concreto, computazionalmente risolvibile, tramite la seguente formulazione in programmazione semidefinita positiva (SP1):

$$\min_{v(s)} \sum_{s=1}^{S} v(s),$$

subject to

$$\mathcal{H}^*(Z(s)+(c+d)\frac{1}{2}(L_1 W(s) L_2^T + L_2 W(s) L_1^T) - (c*d) L_1 W(s) L_1^T - L_2 W(s) L_2^T) =$$

$$= E(s)\overline{v} - R(s)\overline{\nu}(s) - \beta Q(s)\overline{v}, \quad \forall s \in S \quad (3.29a)$$

$$e_1^T \overline{\nu}(s) = 1, \quad \forall s \in S \quad (3.29b)$$

$$-c * \mathcal{H}([\nu_0(s), \ldots, \nu_m(s)]^T) + \mathcal{H}([\nu_1(s), \ldots, \nu_{m+1}(s)]^T) \succeq 0, \quad \forall s \in S \quad (3.29c)$$

$$d * \mathcal{H}([\nu_0(s), \ldots, \nu_m(s)]^T) - \mathcal{H}([\nu_1(s), \ldots, \nu_{m+1}(s)]^T) \succeq 0, \quad \forall s \in S \quad (3.29d)$$

$$Z(s), W(s) \succeq 0, \quad \forall s \in S \quad (3.29e)$$

Possiamo quindi definire il seguente Lemma:

Lemma 8. Sia $A = A_1 = A_2 = [c, d]$. Il problema di programmazione semidefinita positiva SP1, definito in (3.29), risolve esattamente il problema di ottimizzazione polinomiale PP1, definito in (3.24).

Dimostrazione. La disuguaglianza polinomiale definita in (3.24a) ha come vettore dei coefficienti $E(s)\overline{v} - R(s)\overline{\nu}(s) - \beta Q(s)\overline{v}$, come mostrato nella presentazione delle equazioni (3.25) e (3.26). La dimostrazione si ha quindi per diretta conseguenza del Lemma 4, riguardante la rappresentazione della non negatività dei polinomi univariati nell'intervallo $[c, d]$ tramite programmazione semidefinita positiva, applicato come mostrato nell'equazione (3.27), e del Lemma 6, riguardante la rappresentazione delle sequenze valide di momenti di una misura non negativa supportata nell'intervallo $[c, d]$ tramite

programmazione semidefinita positiva, applicato come mostrato nel sistema di equazioni (3.28). □

La soluzione di questo problema produrrà il vettore dei valori del gioco ed i momenti delle misure, una per ogni stato del gioco, per il secondo giocatore, che è quello che non controlla le probabilità di transizione tra uno stato e l'altro: per ottenere dai momenti i supporti ed i pesi delle strategie miste ottime del giocatore si utilizzerà poi la procedura mostrata nella Sezione 3.1.4.

Per ricavare invece le strategie ottime del primo giocatore, dovremo ricavare i momenti delle misure del problema duale a quello appena mostrato. Si definisca con $\epsilon = [\epsilon(1), \ldots, \epsilon(\mathrm{S})]$ una strategia mista, e con $\overline{\epsilon}(\mathrm{s}) \in \mathbb{R}^{d_s+1}$ il vettore relativo ai suoi primi d_s+1 momenti. Si definisca poi la variabile $\delta(s, s')$, che vale 1 se $s = s'$, e vale 0 in tutti gli altri i casi. Infine si definisca con $\alpha = [\alpha(1), \ldots, \alpha(S)]$ il vettore dei valori del gioco relativamente al problema duale. Il problema di ottimizzazione polinomiale duale PD1 relativo al primale PP1 mostrato in (3.24) sarà così definito:

$$\max_{\alpha(s)} \textstyle\sum_{s=1}^{S} \alpha(s), \quad \text{subject to}$$

$$-\alpha(s) + \sum_{i=0}^{n_s} \sum_{j=0}^{m_s} r_{ij}(s)\epsilon_i(s){a_2}^j \in \mathcal{P}(A_2), \quad \forall s \in S \tag{3.30a}$$

$$\overline{\epsilon}(s) \in \mathcal{M}(A_1), \quad \forall s \in S \tag{3.30b}$$

$$\sum_{s=1}^{S} \int_{a_1 \in A_1} \Big(\delta(s, s') - \beta p(s', s, a_1)\Big) d\epsilon(s) = 1, \quad \forall s' \in S \tag{3.30c}$$

Applicando le stesse tecniche mostrate in precedenza, mostriamo ora il problema di programmazione semidefinita duale SD1, equivalente a quello di ottimizzazione polinomiale PD1 appena mostrato in (3.30) e duale del problema di programmazione semidefinita SP1 mostrato in (3.29):

$$\max_{\alpha(s)} \sum_{s=1}^{S} \alpha(s),$$

subject to

$$\mathcal{H}^*(J(s)+(c+d)\frac{1}{2}(L_1K(s)L_2^T+L_2K(s)L_1^T)-(c*d)L_1K(s)L_1^T-L_2K(s)L_2^T) =$$

$$= -\alpha(s)e_1 + R(s)^T\overline{\epsilon}(s), \quad \forall s \in S \tag{3.31a}$$

$$\sum_{s\in S}\Big[E(s) - \beta Q(s)\Big]^T \overline{\epsilon}(s) = 1 \tag{3.31b}$$

$$-c * \mathcal{H}([\epsilon_0(s), \ldots, \epsilon_n(s)]^T) + \mathcal{H}([\epsilon_1(s), \ldots, \epsilon_{n+1}(s)]^T) \succeq 0, \quad \forall s \in S \tag{3.31c}$$

$$d * \mathcal{H}([\epsilon_0(s), \ldots, \epsilon_n(s)]^T) - \mathcal{H}([\epsilon_1(s), \ldots, \epsilon_{n+1}(s)]^T) \succeq 0, \quad \forall s \in S \tag{3.31d}$$

$$J(s), K(s) \succeq 0, \quad \forall s \in S \tag{3.31e}$$

Come fatto per il problema SDP primale, esprimiamo ora il seguente lemma riguardante il problema duale:

Lemma 9. Sia $A = A_1 = A_2 = [c, d]$. Il problema di programmazione semidefinita positiva SD1, definito in (3.31), risolve esattamente il problema di ottimizzazione polinomiale PD1, definito in (3.30).

Dimostrazione. La dimostrazione è del tutto analoga a quella mostrata nel Lemma 8, cioè procede tramite applicazione diretta del Lemma 4 e del Lemma 6 sul problema di ottimizzazione polinomiale PD1 (3.30). □

Per formalizzare il rapporto di dualità tra i problemi di ottimizzazione polinomiale PP1 (3.24) e PD1 (3.30), quindi anche tra i problemi di programmazione semidefinita SP1 (3.29) e DP1 (3.31), come conseguenza diretta rispettivamente del Lemma 8 e del Lemma 9, presentiamo ora il seguente lemma:

Lemma 10. I problemi di ottimizzazione polinomiale PP1 (3.24) e PD1 (3.30) sono tra loro in rapporto di dualità forte.

Dimostrazione. La dimostrazione si può trovare in [10]. □

Analizzando meglio il problema duale, è importante notare che la sequenza dei momenti $\overline{\epsilon}$ ricavata non corrisponde sempre ad una misura di probabilità, quindi non è propriamente la sequenza dei momenti delle misure ottime per il primo giocatore, che all'inizio della sezione era stata appunto definita in un altro modo, cioè come $\overline{\mu}$. Stessa cosa succede per il vettore dei valori α, che non equivale al vettore ottimo dei valori del gioco. Questo succede perché entrambi i vettori devono essere in qualche modo normalizzati per renderli i veri vettori ottimi. Poiché il vettore ottimo dei valori del gioco è ottenuto già tramite la soluzione del problema primale, ora mostreremo solo la normalizzazione da realizzare sul vettore dei momenti.

Fissato uno stato $s \in S$, si consideri la sequenza di momenti $(\epsilon_0(s), \ldots, \epsilon_{n+1}(s))$ ricavata da (3.31). La sequenza normalizzata corrispondente, cioè μ(s), si ricava dividendo tutti gli elementi della stessa per il momento di ordine zero, cioè:

$$\mu(s) = \Big(\frac{\epsilon_0(s)}{\epsilon_0(s)}, \frac{\epsilon_1(s)}{\epsilon_0(s)}, \ldots, \frac{\epsilon_{n+1}(s)}{\epsilon_0(s)}\Big) \tag{3.32}$$

Ora abbiamo anche la sequenza valida dei momenti delle misure, una per ogni stato del gioco, per il primo giocatore: per ottenere dai momenti i supporti ed i pesi delle strategie miste ottime si utilizzerà poi la procedura mostrata nella Sezione 3.1.4.

Per concludere la trattazione, ci manca solo dare una dimostrazione formale dell'ottimalità della soluzione trovata. Per fare ciò, riporteremo alcuni dei risultati presentati in [6]. Per prima cosa presentiamo il seguente lemma:

Lemma 11. Siano $\overline{\nu}^*(s)$ e $\overline{\epsilon}^*(s)$ le sequenze di momenti ottime rispettivamente per PP1 (3.24) e PD1 (3.30). Siano $\nu^*(s)$ e $\epsilon^*(s)$ le corrispondenti misure supportate rispettivamente su A_2 e su A_1. Si ha quindi il seguente risultato complementare per l'ottimo di PP1 (3.24) e PD1 (3.30):

$$v^*(s) \int_{A_1} d\epsilon^*(s) = \int_{A_2}\int_{A_1} r(s, a_1, a_2) d\epsilon^*(s) d\nu^*(s) +$$

$$+ \beta \sum_{s' \in S} v^*(s') \int_{A_1} p(s', s, a_1) d\epsilon^*(s), \quad \forall s \in S \tag{3.33}$$

$$\alpha^*(s) \int_{A_2} d\nu^*(s) = \int_{A_2}\int_{A_1} r(s, a_1, a_2) d\epsilon^*(s) d\nu^*(s), \quad \forall s \in S \tag{3.34}$$

Dimostrazione. La dimostrazione si può trovare a pagina 68 di [6] □

Ora possiamo infine enunciare il teorema che dimostra l'ottimalità della soluzione trovata tramite la coppia di problemi PP1 (3.24) e PD1 (3.30):

Teorema 12. *Sia p^* il valore ottimo di PP1 (3.24), e d^* sia il valore ottimo di PD1 (3.30). Siano ν^* e ϵ^* le misure ottime ricostruite rispettivamente in PP1 (3.24) e PD1 (3.30), e siano*

$$\mu^*(s) = \frac{\epsilon^*(s)}{\int_{A_1} d\epsilon^*(s)}$$

in modo che μ^* sia la versione normalizzata di ϵ^* (ovvero μ^* è effettivamente una misura di probabilità). Infine sia v^* il vettore ottenuto dalla soluzione ottima di PP1 (3.24). La soluzione ottima della coppia primale-duale PP1 (3.24) e PD1 (3.30) così definita soddisfa le seguenti affermazioni:

1. $p^* = d^*$.
2. $v^* = v_\beta(\mu^*, \nu^*)$.
3. $v_\beta(\mu^*, \nu^*)$ *soddisfa la saddle-point condition (2.7) come segue:*

$$v_\beta(\mu, \nu^*) \leq v_\beta(\mu^*, \nu^*) \leq v_\beta(\mu^*, \nu)$$

per ogni coppia di strategie miste (μ, ν).

Dimostrazione. La dimostrazione si può trovare a pagina 69 di [6] □

Per concludere, si può osservare che i risultati prodotti in questa sezione non sono altro che una naturale generalizzazione dei risultati ottenibili tramite la classica soluzione in programmazione lineare di giochi finiti a somma zero, infatti ne condividono anche le stesse importanti proprietà di dualità.

3.3 MDP Polinomiali per il calcolo delle Best Response

In questa sezione presenteremo e risolveremo un particolare problema MDP che riguarda la ricerca della strategia di Best Response di un determinato giocatore, supponendo di aver fissato la strategia del giocatore avversario. Questo tipo di problema può essere visto come una rivisitazione del classico problema MDP mostrato in (2.5) nella Sezione 2.5. La formulazione e risoluzione di questo problema saranno poi utilizzati anche nella prossima sezione, in quanto questo problema è fondamentale per la fase iterativa dell'algoritmo di risoluzione di un MDP Switching Controller I risultati che verranno presentati estendono ad un spazio delle azioni continuo nell'intervallo $[c, d]$ quelli trattati in [6].

3.3.1 Problema di Best Response astratto

Iniziamo la trattazione di questo particolare problema descrivendone tutte le componenti fondamentali. Si consideri un MDP come quello definito nella Definizione 21, in cui siano presenti nel gioco $N = 2$ giocatori, come nel caso analizzato in Sezione 2.5. Si mantenga lo spazio degli stati, S, come un insieme finito di elementi s, mentre si consideri lo spazio delle azioni A di gioco come un insieme infinito di elementi in un determinato intervallo finito, cioè $A = [c, d] \in \mathbb{R}$; per semplicità di trattazione si assuma che $A = A_1 = A_2$, e che lo spazio delle azioni sia uguale per ogni stato del gioco. Si consideri la funzione di reward R a coefficienti reali di tipo polinomiale nelle variabili a_1 ed a_2 come quella definita in (3.1) per il giocatore 1 in ogni stato s del gioco, senza porre nessuna ipotesi, come si faceva di solito indicando che il gioco fosse per esempio a somma zero, sulle proprietà della funzione stessa. Infine si consideri la probabilità di transizione $p(s', s, a_1)$ a coefficienti reali di tipo polinomiale nelle variabili a_1 ed a_2, definita come segue:

$$p(s', s, a_1, a_2) = \sum_{i=0}^{n_{ss'}} \sum_{j=0}^{m_{ss'}} p_{ij}(s', s) {a_1}^i {a_2}^j$$

Come fatto per gli MDP Single Controller, essendo il processo potenzialmente infinito in termini di orizzonte decisionale, ci concentreremo soprattutto sulla ricerca delle strategie stazionarie dei diversi giocatori, strategie definite nella Definizione 19 e che prevedono la dipendenza delle stesse solo dallo stato del gioco e non dal tempo. Di conseguenza, risulterà naturale considerare l'insieme delle strategie miste dei giocatori, in modo da potersi

ricondurre alla nozione di equilibrio minmax. A tal proposito, definiamo una strategia mista per il primo giocatore come l'insieme finito delle misure di probabilità $\mu = [\mu(1), \ldots, \mu(\mathrm{S})]$ supportato in A_1 ed indicizzato dallo stato. In modo analogo si definisca una strategia mista per il secondo giocatore come $\nu = [\nu(1), \ldots, \nu(\mathrm{S})]$ supportato in A_2.

Presentati tutti gli elementi caratteristici del problema, passiamo ora alla formulazione dello stesso. Poiché il problema prevede, nel caso classico, la ricerca della best response del primo giocatore ad un fissato profilo di strategie del giocatore avversario, si fissi la strategia mista $\nu = \nu_0$ per ogni stato $s \in S$. Si definisca quindi con $\nu_0(s, a_2)$ la probabilità che nello stato s sia scelta dal secondo giocatore l'azione a_2 seguendo la strategia fissata $\nu_0(s)$. Infine si definisca con $\Psi_{\nu_0} = (\Psi_{\nu_0}(1), \ldots, \Psi_{\nu_0}(S))$ l'insieme dei supporti della strategia mista ν_0, indicizzato dallo stato: essendo un insieme dei supporti delle strategie, sarà formanto unicamente da tutte e sole le azioni giocate con probabilità maggiore di 0 secondo la strategia fissata ν_0.

Ora abbiamo tutte le nozioni necessarie per scrivere la formulazione. Il problema di ottimizzazione polinomiale per la ricerca della Best Response per il primo giocatore, chiamato BR1, rispetto alla strategia fissata ν_0 del giocatore avversario sarà così definito:

$$\min_{v(s)} \sum_{s=1}^{S} v(s), \quad \textit{subject to}$$

$$v(s) - r(s, a_1) - \beta \sum_{s'=1}^{S} p(s', s, a_1) v(s') \in \mathcal{P}(A), \quad \forall s \in S, \tag{3.35}$$

dove la funzione di reward e quella di transizione sono definite rispettivamente come segue:

$$r(s, a_1) = \sum_{z \in \Psi_{\nu_0}(s)} \sum_{i=0}^{n_s} \sum_{j=0}^{m_s} r_{ij}(s) {a_1}^i z^j \nu_0(s, z), \quad \forall s \in S, a_1 \in A_1 \tag{3.36}$$

$$p(s', s, a_1) = \sum_{z \in \Psi_{\nu_0}(s)} \sum_{i=0}^{n_{ss'}} \sum_{j=0}^{m_{ss'}} p_{ij}(s', s) {a_1}^i z^j \nu_0(s, z), \quad \forall s \in S, a_1 \in A_1. \tag{3.37}$$

3.3.2 Best Response del giocatore 1 in programmazione SDP

Poiché la formulazione del problema BR1 (3.35) appena presentata risulta astratta e non direttamente risolvibile, dobbiamo cercare di modificarla in

modo da esplicitare il vincolo basato sull'insieme dei polinomi univariati non negativi $\mathcal{P}(A)$.

Il vincolo presente nel problema di Best Response può essere visto come una disuguaglianza polinomiale nella variabile a_1 per ogni stato s; fissando quindi uno stato $s \in S$, il nostro obiettivo sarà quindi di esplicitare i coefficienti del polinomio $t_s(a_1)$ della disuguaglianza, cioè:

$$t_s(a_1) \;=\; v(s) - \sum_{z \in \Psi_{\nu_0}(s)} \sum_{i=0}^{n_s} \sum_{j=0}^{m_s} r_{ij}(s) {a_1}^i z^j \nu_0(s,z) +$$

$$-\beta \sum_{s'=1}^{S} \sum_{z \in \Psi_{\nu_0}(s)} \sum_{i=0}^{n_{ss'}} \sum_{j=0}^{m_{ss'}} p_{ij}(s',s) {a_1}^i z^j \nu_0(s,z) v(s')$$

Si definisca con d_s il grado del polinomio per lo stato s, e con $[a_1]_{d_s} = [1, a_1, a_1^2, \ldots, a_1^{d_s}]^T$ il vettore delle azioni nello stato s, indicizzato dal grado. Si definisca con $R(s)$ la matrice dei coefficienti del polinomio $r(s,a)$ definito in (3.36), e con $Q(s)$ la matrice dei coefficienti del polinomio $p(s',s,a_1)$ definito in (3.37). Si definisca poi $\overline{v} = [v(1), \ldots, v(S)]^T$, cioè il vettore dei valori del gioco. Per ogni insieme X, sia $\prod(X)$ l'insieme di tutte le distribuzioni di probabilità su X. Si definisca quindi con $\Phi = \prod A_2(1) \times \prod A_2(2) \times \cdots \times \prod A_2(S)$ l'insieme delle strategie miste per il secondo giocatore. Si definisca infine l'operatore lineare $\mathcal{F} : S \times A_S \times \Phi \longmapsto \mathbb{R}^{m_s+1}$ come segue:

$$\mathcal{F}(s, a_2, \nu) \;=\; \nu(s, a_2)[1, a_2, a_2^2, \ldots, a_2^{m_s}]$$

Date le nozioni appena presentate, possiamo riscrivere i termini del polinomio univariato $t_s(a_1)$, rispettivamente la funzione di reward e la funzione di transizione, così:

$$\sum_{z \in \Psi_{\nu_0}(s)} \sum_{i=0}^{n_s} \sum_{j=0}^{m_s} r_{ij}(s) {a_1}^i z^j \nu_0(s,z) \;=\; R(s) \Big(\sum_{z \in \Psi_{\nu_0}(s)} \mathcal{F}(s,z,\nu) \Big)^T [a_1]_{d_s}; \quad (3.38)$$

$$\sum_{z \in \Psi_{\nu_0}(s)} \sum_{i=0}^{n_{ss'}} \sum_{j=0}^{m_{ss'}} p_{ij}(s',s) {a_1}^i z^j \nu_0(s,z) \;=\; Q(s) \overline{v} [a_1]_{d_s}. \quad (3.39)$$

Data questa nuova formulazione vettoriale delle parti che compongono il vincolo (3.35), e riprendendo quanto detto nel Lemma 4, si può esprimere la non negatività del polinomio univariato $t_s(a_1)$ sull'intervallo $[c,d]$ tramite il vincolo

$$\mathcal{H}^*(Z(s) + (c+d)\frac{1}{2}(L_1 W(s) L_2^T + L_2 W(s) L_1^T) - (c*d) L_1 W(s) L_1^T - L_2 W(s) L_2^T) \;=$$

$$= E(s)\overline{v} - R(s)\Big(\sum_{z\in\Psi_{\nu_0}(s)} \mathcal{F}(s,z,\nu)\Big)^T - \beta Q(s)\overline{v} \tag{3.40}$$

con le matrici $Z(s) \in \mathcal{S}^{d_s+1}$ e $W(s) \in \mathcal{S}^{d_s}$ semidefinite positive, e con $E(s) \in \mathbb{R}^{d_s \times S}$ che rappresenta una matrice formata da tutti zeri eccetto un 1 in posizione $(1, s)$.

Ora, grazie alla formulazione in (3.40) appena definita, possiamo riformulare il problema di ottimizzazione polinomiale astratto BR1 (3.35) in modo più concreto, computazionalmente risolvibile, tramite la seguente formulazione in programmazione semidefinita positiva (SBR1):

$$\min_{v(s)} \sum_{s=1}^{S} v(s),$$

subject to

$$\mathcal{H}^*(Z(s)+(c+d)\frac{1}{2}(L_1W(s)L_2^T+L_2W(s)L_1^T)-(c*d)L_1W(s)L_1^T-L_2W(s)L_2^T) =$$

$$= E(s)\overline{v} - R(s)\Big(\sum_{z\in\Psi_{\nu_0}(s)} \mathcal{F}(s,z,\nu)\Big)^T - \beta Q(s)\overline{v}, \quad \forall s \in S \tag{3.41a}$$

$$Z(s),\, W(s) \succeq 0, \quad \forall s \in S \tag{3.41b}$$

Possiamo quindi definire il seguente Lemma:

Lemma 12. Sia $A = A_1 = A_2 = [c, d]$. Il problema di programmazione semidefinita positiva SBR1, definito in (3.41), risolve esattamente il problema di ottimizzazione polinomiale BR1, definito in (3.35).

Dimostrazione. La disuguaglianza polinomiale definita in (3.35) ha come vettore dei coefficienti $E(s)\overline{v} - R(s)\Big(\sum_{z\in\Psi_{\nu_0}(s)} \mathcal{F}(s,z,\nu)\Big)^T - \beta Q(s)\overline{v}$, come mostrato nelle espressioni (3.38) e (3.39). La dimostrazione si ha quindi per diretta conseguenza del Lemma 4, riguardante la rappresentazione della non negatività dei polinomi univariati nell'intervallo $[c, d]$ tramite programmazione semidefinita positiva, applicato come mostrato nell'equazione (3.40). □

La soluzione di questo problema produrrà il vettore dei valori del gioco quando il primo giocatore applica la sua Best Response rispetto ad un profilo di strategie fissato del secondo giocatore.

3.3.3 Best Response del giocatore 2

Per completare questa trattazione, presentiamo brevemente anche il problema di ricerca della best response del secondo giocatore ad un fissato profilo di strategie del giocatore avversario. Si ripetano tutte le considerazioni fatte per caso di ricerca della best response del primo giocatori.

Il problema di ottimizzazione polinomiale astratto per la ricerca della Best Response per il secondo giocatore, chiamato BR2, rispetto alla strategia fissata μ_0 del giocatore avversario sarà così definito:

$$\max_{v(s)} \sum_{s=1}^{S} v(s), \quad subject\, to$$

$$-v(s) + r(s, a_2) + \beta \sum_{s'=1}^{S} p(s', s, a_2) v(s') \in \mathcal{P}(A), \quad \forall s \in S, \tag{3.42}$$

dove la funzione di reward e quella di transizione sono definite rispettivamente come segue:

$$r(s, a_2) = \sum_{z \in \Psi_{\mu_0}(s)} \sum_{i=0}^{n_s} \sum_{j=0}^{m_s} r_{ij}(s) {a_2}^j z^i \mu_0(s, z), \quad \forall s \in S, a_2 \in A_2 \tag{3.43}$$

$$p(s', s, a_2) = \sum_{z \in \Psi_{\mu_0}(s)} \sum_{i=0}^{n_{ss'}} \sum_{j=0}^{m_{ss'}} p_{ij}(s', s) {a_2}^j z^i \mu_0(s, z), \quad \forall s \in S, a_1 \in A_2. \tag{3.44}$$

Si definisca l'operatore lineare $\mathcal{G} : S \times A_S \times \Phi \longmapsto \mathbb{R}^{n_s+1}$ come segue:

$$\mathcal{G}(s, a_1, \mu) = \mu(s, a_1)[1, a_1, a_1^2, \ldots, a_1^{n_s}]$$

Si ripetano quindi tutte le definizioni e le considerazioni fatte in Sezione 3.3.2.

Possiamo infine riformulare il problema di ottimizzazione polinomiale astratto BR2 (3.42) in modo computazionalmente risolvibile tramite la seguente formulazione in programmazione semidefinita positiva (SBR2):

$$\max_{v(s)} \sum_{s=1}^{S} v(s),$$

subject to

$$\mathcal{H}^*(Z(s)+(c+d)\frac{1}{2}(L_1W(s)L_2^T+L_2W(s)L_1^T)-(c*d)L_1W(s)L_1^T-L_2W(s)L_2^T) =$$

$$= -E(s)\overline{v} + R(s)\Big(\sum_{z\in\Psi_{\mu_0}(s)} \mathcal{G}(s,z,\mu)\Big)^T + \beta Q(s)\overline{v}, \quad \forall s \in S \tag{3.45a}$$

$$Z(s),\, W(s) \succeq 0, \quad \forall s \in S \tag{3.45b}$$

Possiamo quindi definire il seguente Lemma:

Lemma 13. Sia $A = A_1 = A_2 = [c,d]$. Il problema di programmazione semidefinita positiva SBR2, definito in (3.45), risolve esattamente il problema di ottimizzazione polinomiale BR2, definito in (3.42).

Dimostrazione. La disuguaglianza polinomiale definita in (3.42) ha come vettore dei coefficienti $-E(s)\overline{v} + R(s)\Big(\sum_{z\in\Psi_{\mu_0}(s)} \mathcal{G}(s,z,\mu)\Big)^T + \beta Q(s)\overline{v}$. La dimostrazione si ha quindi per diretta conseguenza del Lemma 4, riguardante la rappresentazione della non negatività dei polinomi univariati nell'intervallo $[c,d]$ tramite programmazione semidefinita positiva. □

La soluzione di questo problema produrrà il vettore dei valori del gioco quando il secondo giocatore applica la sua Best Response rispetto ad un profilo di strategie fissato del primo giocatore.

3.4 MDP Polinomiali Switching Controller

Nella precedente sezione è stata presentato il problema di risoluzione di un MDP con proprietà di Single Controller, in cui cioè le probabilità di transizione da uno stato all'altro del gioco dipendono solo dalle azioni di uno stesso unico giocatore, quindi ne è stata proposta una soluzione basata su formulazione in programmazione semidefinita positiva. In questa sezione estenderemo il problema suddetto introducendo la sottoclasse di giochi stocastici con proprietà di Switching controller, in cui cioè i 2 giocatori si alternano nel controllo delle probabilità di transizione in base al particolare sottoinsieme degli stati in cui si trovano. Di questo nuovo problema ne descriveremo inizialmente la formulazione risolutiva nel caso semplice in cui lo spazio delle azioni dei giocatori sia finito, quindi ne presenteremo la formulazione estesa (astratta) nel caso in cui si consideri uno spazio delle azioni con numero di elementi infinito di elementi e una funzione di reward di tipo polinomiale, infine produrremo una formulazione in programmazione semidefinita positiva computazionalmente calcolcabile (concreta) per quest'ultimo caso. I risultati presenti quindi in questa sezione riprendono quelli trattati in [6] estendendoli al caso in cui lo spazio delle azioni è un intervallo continuo generico $[c, d]$.

3.4.1 Presentazione di un problema MDP finito con proprietà di Switching Controller

I giochi stocastici con proprietà di Switching Controller sono definiti come una superclasse rispetto a quelli già visti con proprietà di Single Controller: infatti in questi giochi si divide l'insieme degli stati S in due sottoinsiemi, S_1 ed S_2, quindi ogni giocatore ha il controllo delle transizioni solo in uno di questi 2. Questa classe di giochi fù presentata per la prima volta nel 1981 in [12], quindi in [13] si dimostrò che questi giochi posseggono una particolare proprietà, detta proprietà di Orderfield, la quale permette di affermare che si può risolvere il problema descritto da questa classe tramite un algoritmo finito. Proprio per questa sua proprietà, crebbero interesse e ricerche attorno a questo problema, come mostrato in [14], [15], [16] e [17].

Ora diamo una definizione formale di questa classe di giochi:

Definizione 29 (Gioco Stocastico con Switching Controller)**.** Un gioco stocastico a due giocatori G, definito dalla tupla (S, N, A, P, R), ha la proprietà di Switching Controller se l'insieme degli stati S può essere partizionato in due sottoinsiemi S_1 ed S_2 tali che le probabilità di transizione nel gioco sono

definite come

$$p(s', s, a_1, a_2) = p(s', s, a_1), \quad \forall s \in S_1, s' \in S, a_1 \in A_1(s), a_2 \in A_2(s); \tag{3.46a}$$

$$p(s', s, a_1, a_2) = p(s', s, a_2), \quad \forall s \in S_2, s' \in S, a_1 \in A_1(s), a_2 \in A_2(s); \tag{3.46b}$$

Data questa definizione, possiamo quindi passare alla descrizione del problema di ricerca di strategie d'equilibrio nel gioco, in particolare restringendo, come già fatto precedentemente, l'analisi alla sottoclasse dei giochi stocastici chiamata MDP. Presentiamo per prima cosa gli elementi che definiscono il nostro problema MDP. Si consideri un gioco a 2 giocatori, a somma zero, con uno spazio degli stati $S = 1, ..., N$ partizionato in 2 sottoinsiemi $S_1 = 1, ..., k$ ed $S_2 = k + 1, ..., N$, tale che la funzione P relativa alla probabilità di transizione da uno stato all'altro del gioco sia definita come nell'equazione (3.46). Si consideri poi uno spazio delle azioni $A = A_1 = A_2$ che, in questo primo caso base, è finito. Si definisca il fattore di sconto β. Si considerino le definizioni di strategie miste, rispettivamente per il primo e secondo giocatore, f e g date nella Sezione 2.5. Infine si definisca, per un dato stato $s \in S$, la funzione di reward indotta da entrambe le strategie f e g come:

$$r(s, f(s), g(s)) = \sum_{a_1 \in A_1, a_2 \in A_2} r(s, a_1, a_2) f(s, a_1) g(s, a_2).$$

Il reward accumulabile su un orizzonte ipoteticamente infinito di turni partendo dallo stato s, dati gli elementi appena presentati, sarà definito dal seguente sistema di equazioni:

$$v_\beta(s, f(s), g(s)) = r(s, f(s), g(s)) + \beta \sum_{s' \in S_1} \Big(\sum_{a_1 \in A_1} p(s', s, a_1) f(s, a_1) \Big) v_\beta(s', f(s'), g(s')) +$$

$$+ \beta \sum_{s' \in S_2} \Big(\sum_{a_1 \in A_1} p(s', s, a_1) f(s, a_1) \Big) w_\beta(s', f(s'), g(s'))$$

$$w_\beta(s, f(s), g(s)) = r(s, f(s), g(s)) + \beta \sum_{s' \in S_1} \Big(\sum_{a_2 \in A_2} p(s', s, a_2) g(s, a_2) \Big) v_\beta(s', f(s'), g(s')) +$$

$$+ \beta \sum_{s' \in S_2} \Big(\sum_{a_2 \in A_2} p(s', s, a_2) g(s, a_2) \Big) w_\beta(s', f(s'), g(s'))$$

Come si può subito notare dal sistema appena presentato, passando da MDP a Single Controller a quelli Switching Controller, la complessità della soluzione aumenta: infatti in questo caso, per calcolare una soluzione di

equilibrio minmax, non basterà più risolvere una singola coppia di problemi primale-duale di programmazione lineare, ma bisognerà costruire un algoritmo che risolve 2 coppie di problemi in modo iterativo fino al raggiungimento della convergenza della soluzione. In questo algoritmo, i problemi primali restituiranno i momenti delle strategie di un giocatore negli stati in cui le probabilità di transizione sono governate dall'avversario, e il vettore dei valori del gioco visto come somma dei vettori dei valori dei 2 problemi; per quanto riguarda invece i problemi duali, essi restituiranno i momenti delle strategie di un giocatore negli stati in cui è proprio lui che governa le transizioni.

Si supponga di avere delle stime aribitrarie di $\widehat{v}$, cioè $(\widehat{v}(1), \widehat{v}(2), \ldots, \widehat{v}(k))$, e di $\widehat{w}$, cioè $(\widehat{w}(k+1), \widehat{w}(k+2), \ldots, \widehat{w}(N))$. Il primo dei 2 problemi primali, formulato in programmazione lineare e specificato con la sigla LP3, è definito come segue:

$$\max_{w(s)} \sum_{s \in S_2} w(s),$$

subject to

$$+\beta \sum_{s' \in S_2} p(s', s, a_2) w(s') + \sum_{a_1 \in A_1} r(s, a_1, a_2) f(s, a_1) +$$

$$-w(s) \geq -\beta \sum_{s' \in S_1} p(s', s, a_2) \widehat{v}(s'), \quad \forall s \in S_2, \forall a_2 \in A_2 \tag{3.47a}$$

$$\sum_{a_1 \in A_1} f(s, a_1) = 1, \quad \forall s \in S_2 \tag{3.47b}$$

$$f(s, a_1) \geq 0, \quad \forall s \in S_2, \forall a_1 \in A_1 \tag{3.47c}$$

Questo problema, che sappiamo avere una soluzione ottima grazie ai risultati proposti in [18], restituisce il vettore dei valori del gioco ed il vettore delle strategie del primo giocatore per tutti gli stati $s \in S_2$, cioè negli stati in cui le transizioni sono governate dal secondo giocatore, con l'assunzione di aver fissato i valori di $\widehat{v}$.

Il secondo dei 2 problemi primali, formulato sempre in programmazione lineare e specificato con la sigla LP4, è definito invece come segue:

$$\min_{v(s)} \sum_{s \in S_1} v(s),$$

subject to

$$-\beta \sum_{s' \in S_1} p(s', s, a_1) v(s') - \sum_{a_2 \in A_2} r(s, a_1, a_2) g(s, a_2) +$$

$$+ v(s) \geq \beta \sum_{s' \in S_2} p(s', s, a_1) \widehat{w}(s'), \quad \forall s \in S_1, \forall a_1 \in A_1 \tag{3.48a}$$

$$\sum_{a_2 \in A_2} g(s, a_2) = 1, \quad \forall s \in S_1 \tag{3.48b}$$

$$g(s, a_2) \geq 0, \quad \forall s \in S_1, \forall a_2 \in A_2 \tag{3.48c}$$

Questo problema, che sappiamo avere una soluzione ottima sempre grazie ai risultati proposti in [18], restituisce il vettore dei valori del gioco ed il vettore delle strategie del secondo giocatore per tutti gli stati $s \in S_1$, cioè negli stati in cui le transizioni sono governate dal primo giocatore, con l'assunzione di aver fissato i valori di $\widehat{w}$.

Presentiamo ora un teorema che ci conferma che quello che verrà trovato iterando i 2 problemi primali è realmente il vettore dei valori del gioco.

Teorema 13. *Si supponga di avere le seguenti due coppie di vettori di valori del gioco e strategie miste:*

$$v(s), g(s, a_2), \quad \forall s \in S_1, \, a_2 \in A_2$$

$$w(s), f(s, a_1), \quad \forall s \in S_2, \, a_1 \in A_1$$

dove $\Big(v(s), g(s, a_2), \quad \forall s \in S_1, \, a_2 \in A_2\Big)$ *è ottima per il problema LP4 descritto in (3.48) quando* $\widehat{w}(t) = w(t)$*, con* $t \in S_2$*, e dove* $\Big(w(s), f(s, a_1), \quad \forall s \in S_2, \, a_1 \in A_1\Big)$ *è ottima per il problema LP3 descritto in (3.47) quando* $\widehat{v}(s) = v(s)$*, con* $s \in S_1$*. Allora* $\Big(v(1), v(2), \ldots, v(k), w(k+1), w(k+2), \ldots, w(N)\Big)$ *è il vettore dei valori del gioco*

Dimostrazione. La dimostrazione è mostrata in [18]. □

Dati i due problemi primali di cui sopra, mostriamo ora i rispettivi problemi duali. Il primo di questi 2 problemi, cioè il duale di LP3 (3.47) formulato in programmazione lineare e specificato con la sigla LD3, è definito come segue:

$$\max_{\alpha(s)} \sum_{s\in S_2} \Big(\alpha(s) - \beta g(s,a_2) \sum_{a_2\in A_2} \sum_{s\in S_1} P(s's,a_2)\widehat{v}(s')\Big),$$

subject to

$$\alpha(s) \le \sum_{a_2\in A_2} g(s,a_2) r(s,a_1,a_2), \quad \forall s \in S_2, \forall a_1 \in A_1 \tag{3.49a}$$

$$\sum_{s\in S_2} \sum_{a_2\in A_2} [\delta(s,s') - \beta P(s,s',a_1)] g(s',a_1) = 1, \quad \forall s' \in S_2 \tag{3.49b}$$

$$g(s,a_2) \ge 0, \quad \forall s \in S_2, \forall a_2 \in A_2 \tag{3.49c}$$

Questo problema restituisce il vettore delle strategie del secondo giocatore per tutti gli stati $s \in S_2$, cioè negli stati in cui è lui il giocatore che controlla le transizioni, con l'assunzione di aver fissato, come nel problema primale, i valori di $\widehat{v}$.

Il secondo dei 2 problemi duali, cioè il duale di LP4 (3.48) formulato sempre in programmazione lineare e specificato con la sigla LD4, è definito come segue:

$$\max_{\eta(s)} \sum_{s\in S_1} \Big(\eta(s) + \beta f(s,a_1) \sum_{a_1\in A_1} \sum_{s\in S_1} P(s's,a_1)\widehat{w}(s')\Big),$$

subject to

$$\eta(s) \le \sum_{a_1\in A_1} f(s,a_1) r(s,a_1,a_2), \quad \forall s \in S_1, \forall a_2 \in A_2 \tag{3.50a}$$

$$\sum_{s\in S_1} \sum_{a_1\in A_1} [\delta(s,s') - \beta P(s,s',a_1)] f(s',a_1) = 1, \quad \forall s' \in S_1 \tag{3.50b}$$

$$f(s,a_1) \ge 0, \quad \forall s \in S_1, \forall a_1 \in A_1 \tag{3.50c}$$

Questo problema restituisce invece il vettore delle strategie del primo giocatore per tutti gli stati s $\in S_1$, cioè negli stati in cui è lui il giocatore che controlla le transizioni, con l'assunzione di aver fissato, come nel problema primale, i valori di $\widehat{w}$.

Per concludere la trattazione di questo primo problema, con spazio delle azioni finito, manca solo la descrizione dell'iterazione dell'algoritmo, e la relativa analisi di convergenza. Si consideri il problema primale LP4 (3.48), e si assuma $\widehat{w}(t) = 0 = w^0(t)$, con $t \in S_2$; sia quindi $(v^0(s), g^0(s))$, con $s \in S_1$, la soluzione ottima del problema LP4, con G_0 che rappresenta questa base ottima. Si consideri successivamente il problema primale LP3 (3.47), e si assuma $\widehat{v}(s) = v^0(s)$, con $s \in S_2$; sia quindi $(w^1(t), g^0(t))$ con $t \in S_2$, la soluzione

ottima del problema LP4, con F_1 che rappresenta questa base ottima. Si ritorni quindi di nuovo al problema primale LP4, e si assuma $\widehat{w}(t) = 0 = w^1(t)$, con $t \in S_2$; la base prima calcolata, cioé G_0, soddisfa sempre la condizione di ottimalità, tuttavia non è detto che la relativa soluzione $(v^0(s), g^0(s))$ sia ancora ammissibile. Se non lo è, si ricalcola una nuova soluzione per LP4, ottenendo $(v^1(s), g^1(s))$, con la relativa base G_1. Si effettua poi lo stesso ragionamento per il problema primale LP3.

L'algoritmo quindi non fà altro che iterare questi passaggi, generando una sequenza di basi G_r, F_r, associate alle soluzioni (v^r, w^r) relative ai 2 problemi primali. Ad ogni nuova coppia di basi generate, viene controllata la sua ammissibilità rispetto ad entrambe le disuguaglianze lineari espresse nei 2 problemi primali LP3 ed LP4: se la coppia di basi risulta ammissibile, allora l'algoritmo terminerà la sua iterazione. La convergenza dell'algoritmo, che equivale anche alla restituzione del vettore ottimo dei valori del gioco (v^*, w^*) e della coppia di strategie ottime dei giocatori, è garantita, in un numero finito di passi, dal seguente teorema:

Teorema 14.

$$\lim_{r \longrightarrow \infty} (v^r, w^r) \; = \; (v^*, w^*).$$

Dimostrazione. La dimostrazione di questo teorema è mostrata in [18]. □

3.4.2 Analisi del problema polinomiale astratto

Dopo aver descritto, analizzato e risolto il primo caso di MDP Switching Controller con spazio delle azioni finito, possiamo ora presentare il suo problema esteso, cioè il problema relativo alla risoluzione di un MDP Switching Controller con spazio delle azioni avente numero di elementi infinito e non numerabile, e con funzione di reward di tipo polinomiale.

Analizziamo per prima cosa i cambiamenti presenti nella tupla che definisce questo nuovo problema. Si consideri un gioco a 2 giocatori, a somma zero, con uno spazio degli stati $S = 1, ..., N$ partizionato in 2 sottoinsiemi $S_1 = 1, ..., k$ ed $S_2 = k + 1, ..., N$, tale che la funzione P relativa alla probabilità di transizione da uno stato all'altro del gioco sia definita come nell'equazione (3.46). Si consideri poi uno spazio delle azioni $A = A_1 = A_2$ con un numero infinito e non numerabile di elementi nell'intervallo continuo $[c, d]$, spazio degli azioni che, per semplicità, si assume sia uguale per ogni stato $s \in S$. Si assuma che la funzione di reward sia di tipo polinomiale, come quella descritta nell'equazione (3.1), per ogni stato $s \in S$. Si definisca infine il fattore di sconto β.

Introdotto il problema, se ne può iniziare l'analisi: per prima cosa si considerino le strategie miste f e g dei 2 giocatori, definite in questo caso tramite le corrispondenti misure di probabilità $\mu(s)$ e $\nu(s)$ sull'insieme delle strategie miste, cioè l'intervallo $[c, d]$ su cui variano a_1 ed a_2. Quest'ultime poi inducono a riformulare le funzioni di reward e di transizione in loro funzione, cioè si avrà una definizione del payoff atteso del primo giocatore come

$$r(s, f(s), g(s)) = \int_{a_1 \in A_1} \int_{a_2 \in A_2} r(s, a_1, a_2) d\mu(s) d\nu(s), \quad \forall s \in S;$$

mentre, data una strategia μ, si definirà una matrice di probabilità di transizione $P(\mu)$ come segue

$$P_{ss'}(\mu) \; = \; \int_{A_1} p(s', s, a_1) d\mu(s), \quad \forall s \in S_1, s' \in S;$$

infine, data una strategia ν, si definirà una matrice di probabilità di transizione $P(\nu)$ come segue

$$P_{ss'}(\nu) \; = \; \int_{A_2} p(s', s, a_2) d\nu(s), \quad \forall s \in S_2, s' \in S;$$

Il reward accumulabile su un orizzonte ipoteticamente infinito di turni partendo dallo stato s, dati gli elementi appena presentati, sarà definito dal seguente sistema di equazioni:

$$v_\beta(s, \mu(s), \nu(s)) \; = \; r(s, \mu(s), \nu(s)) + \beta \sum_{s' \in S_1} \Big(\int_{a_1 \in A_1} p(s', s, a_1) \mu(s, a_1) \Big) v_\beta(s', \mu(s'), \nu(s')) +$$

$$+ \beta \sum_{s' \in S_2} \Big(\int_{a_1 \in A_1} p(s', s, a_1) \mu(s, a_1) \Big) w_\beta(s', \mu(s'), \nu(s')), \quad \forall s \in S_1$$

$$w_\beta(s, \mu(s), \nu(s)) \; = \; r(s, \mu(s), \nu(s)) + \beta \sum_{s' \in S_1} \Big(\int_{a_2 \in A_2} p(s', s, a_2) \nu(s, a_2) \Big) v_\beta(s', \mu(s'), \nu(s')) +$$

$$+ \beta \sum_{s' \in S_2} \Big(\int_{a_2 \in A_2} p(s', s, a_2) \nu(s, a_2) \Big) w_\beta(s', \mu(s'), \nu(s')), \quad \forall s \in S_2$$

Per quanto riguarda la ricerca della soluzione del gioco, si ricorda che valgono le stesse considerazioni fatte nella Sezione 3.2.1 per gli MDP Single Controller (valendo le stesse condizioni di gioco anche in questo caso), cioè è garantita l'esistenza e l'unicità sia di una strategia stazionaria d'equilibrio sia del vettore dei valori del gioco.

Ora abbiamo tutte le nozioni necessarie per poter presentare i 2 problemi primali che generalizzano ad uno spazio di azioni continuo i problemi LP3 (3.47) e LP4 (3.48).

Si supponga di avere delle stime aribitrarie di $\widehat{v}$, cioè $(\widehat{v}(1), \widehat{v}(2), \ldots, \widehat{v}(k))$, e di $\widehat{w}$, cioè $(\widehat{w}(k+1), \widehat{w}(k+2), \ldots, \widehat{w}(N))$. Il primo problema primale di ottimizzazione polinomiale, chiamato PM3, estende il problema LP3 (3.47) ed è definito come segue:

$$\max_{w(s)} \sum_{s \in S_2} w(s),$$

subject to

$$+ \int_{A_1} r(s, a_1, a_2)\mu(s, a_1)da_1 + \beta \sum_{s' \in S_2} p(s', s, a_2)w(s') +$$

$$-w(s) \geq -\beta \sum_{s' \in S_1} p(s', s, a_2)\widehat{v}(s'), \quad \forall s \in S_2, \forall a_2 \in A_2 \tag{3.51a}$$

$$\int_{A_1} \mu(s, a_1)da_1 = 1, \quad \forall s \in S_2 \tag{3.51b}$$

$$\mu(s, a_1) \geq 0, \quad \forall s \in S_2, \forall a_1 \in A_1 \tag{3.51c}$$

Il secondo problema primale di ottimizzazione polinomiale, chiamato PM4, estende il problema LP4 (3.48) ed è definito come segue:

$$\min_{v(s)} \sum_{s \in S_1} v(s),$$

subject to

$$- \int_{A_2} r(s, a_1, a_2)\nu(s, a_2)a_2 - \beta \sum_{s' \in S_1} p(s', s, a_1)v(s') +$$

$$+ v(s) \geq \beta \sum_{s' \in S_2} p(s', s, a_1)\widehat{w}(s'), \quad \forall s \in S_1, \forall a_1 \in A_1 \tag{3.52a}$$

$$\int_{A_2} \nu(s, a_2)da_2 = 1, \quad \forall s \in S_1 \tag{3.52b}$$

$$\nu(s, a_2) \geq 0, \quad \forall s \in S_1, \forall a_2 \in A_2 \tag{3.52c}$$

Questa formulazione però, come nel caso Single Controller, è astratta e non direttamente risolvibile: per renderla tale dovremmo ridefinirla successivamente tramite formulazione in programmazione SDP. Per fare questo però,

occorre prima trasformare i vincoli (3.51a) e (3.52a) in sistemi di disequazioni polinomiali univariate nelle azioni rispettivamente a_2 ed a_1, assumendo i valori di $\mu(s)$ e $\nu(s)$ fissati. L'unica componente non espressa esplicitamente come polinomio univariato risulta essere la funzione di reward, che andrà quindi riscritta così

$$\int_{A_1} r(s, a_1, a_2)\mu(s, a_1)da_1 \;=\; \sum_{i=0}^{n_s}\sum_{j=0}^{m_s} r_{ij}(s)\mu_i(s){a_2}^j$$

nel caso del vincolo (3.51a) in PM3, e così

$$\int_{A_2} r(s, a_1, a_2)\nu(s, a_2)da_2 \;=\; \sum_{i=0}^{n_s}\sum_{j=0}^{m_s} r_{ij}(s)\nu_j(s){a_1}^i$$

nel caso del vincolo (3.52a) in PM4.

Ora possiamo riscrivere i 2 problemi primali relazionandoli agli insiemi $\mathcal{P}_n$, cioè l'insieme dei polinomi univariati di grado n non negativi nell'intervallo $[c, d]$, e $\mathcal{M}_m$, cioè l'insieme dei primi $m+1$ momenti di una misura di probabilità non negativa con supporto sempre nell'intervallo $[c, d]$. Il primo problema di ottimizzazione polinomiale, chiamato PP3 ed equivalente a PM3 (3.51), è così ridefinito:

$$\max_{w(s)} \sum_{s\in S_2} w(s),$$

subject to

$$+\sum_{i=0}^{n_s}\sum_{j=0}^{m_s} r_{ij}(s)\mu_i(s){a_2}^j + \beta \sum_{s'\in S_2} p(s', s, a_2)w(s') +$$

$$+\beta \sum_{s'\in S_1} p(s', s, a_2)\widehat{v}(s') - w(s) \;\in\; \mathcal{P}(A_1), \quad \forall s \in S_2 \tag{3.53a}$$

$$\overline{\mu}(s) \;\in\; \mathcal{M}(A_1), \quad \forall s \in S_2 \tag{3.53b}$$

$$\mu_0(s) = 1, \quad \forall s \in S_2 \tag{3.53c}$$

Il secondo problema di ottimizzazione polinomiale, chiamato PP4 ed equivalente a PM4 (3.52), è invece così ridefinito:

$$\min_{v(s)} \sum_{s \in S_1} v(s),$$

subject to

$$-\sum_{i=0}^{n_s}\sum_{j=0}^{m_s} r_{ij}(s)\nu_j(s){a_1}^i - \beta \sum_{s' \in S_1} p(s', s, a_1)v(s') +$$

$$+ v(s) - \beta \sum_{s' \in S_2} p(s', s, a_1)\widehat{w}(s') \in \mathcal{P}(A_1), \quad \forall s \in S_1 \tag{3.54a}$$

$$\overline{\nu}(s) \in \mathcal{M}(A_2), \quad \forall s \in S_1 \tag{3.54b}$$

$$\nu_0(s) = 1, \quad \forall s \in S_1 \tag{3.54c}$$

Prima di concludere questa parte, presentiamo anche i rispettivi 2 problemi duali di quelli appena mostrati. Si supponga, come già fatto per i problemi primali, di avere delle stime aribitrarie di $\widehat{v}$, cioè $(\widehat{v}(1), \widehat{v}(2), \ldots, \widehat{v}(k))$, e di $\widehat{w}$, cioè $(\widehat{w}(k+1), \widehat{w}(k+2), \ldots, \widehat{w}(N))$.
Si definisca con $\varepsilon = [\varepsilon(\mathrm{k}{+}1), \ldots, \varepsilon(\mathrm{N})]$ una strategia mista del giocatore 2, e con $\overline{\varepsilon}(\mathrm{s}) \in \mathbb{R}^{d_s+1}$ il vettore relativo ai suoi primi $d_s{+}1$ momenti, con d_s che indica il grado del polinomio univariato. Il primo problema, chiamato PD3 e duale di PP3 (3.53), sarà quindi definito così:

$$\max_{\eta(s), \varepsilon_j(s)} \sum_{s \in S_2} \Big(\eta(s) - \beta \sum_{s' \in S_1} \widehat{v}(s') \sum_{j=0}^{m_s} p_j(s', s)\varepsilon_j(s) \Big),$$

subject to

$$-\eta(s) - \sum_{i=0}^{n_s}\sum_{j=0}^{m_s} r_{ij}(s){a_1}^i\varepsilon_j(s) \in \mathcal{P}(A_1), \quad \forall s \in S_2 \tag{3.55a}$$

$$\overline{\varepsilon}(s) \in \mathcal{M}(A_2), \quad \forall s \in S_2 \tag{3.55b}$$

$$\sum_{s \in S_2} \int_{a_2 \in A_2} \Big(\delta(s, s') - \beta p(s', s, a_2) \Big) \varepsilon(s, a_2) da_2 = 1, \quad \forall s' \in S_2 \tag{3.55c}$$

Si definisca infine con $\epsilon = [\epsilon(1), \ldots, \epsilon(\mathrm{k})]$ una strategia mista del giocatore 1, e con $\overline{\epsilon}(\mathrm{s}) \in \mathbb{R}^{d_s+1}$ il vettore relativo ai suoi primi $d_s{+}1$ momenti, con d_s che indica il grado del polinomio univariato.
Il secondo problema, chiamato PD4 e duale di PP4 (3.54), sarà quindi così definito:

$$\max_{\alpha(s),\epsilon_i(s)} \sum_{s\in S_1}\Big(\alpha(s)+\beta\sum_{s'\in S_2}\widehat{w}(s')\sum_{i=0}^{n_s}p_i(s',s)\epsilon_i(s)\Big),$$

subject to

$$-\alpha(s)+\sum_{i=0}^{n_s}\sum_{j=0}^{m_s}r_{ij}(s){a_2}^j\epsilon_i(s)\ \in\ \mathcal{P}(A_2),\quad \forall s\in S_1 \tag{3.56a}$$

$$\overline{\epsilon}(s)\ \in\ \mathcal{M}(A_1),\quad \forall s\in S_1 \tag{3.56b}$$

$$\sum_{s\in S_1}\int_{a_1\in A_1}\Big(\delta(s,s')-\beta p(s',s,a_1)\Big)\epsilon(s,a_1)da_1\ =\ 1,\quad \forall s'\in S_1 \tag{3.56c}$$

3.4.3 Riformulazione del problema tramite programmazione semidefinita positiva

Poiché i problemi presentati precedentemente erano tutti astratti e computazionalmente non risolvibili, bisognerà riformularli: come già fatto per gli MDP Single Controller, si analizzerano gli insiemi $\mathcal{P}(A_1)$, $\mathcal{P}(A_2)$, $\mathcal{M}(A_1)$ e $\mathcal{M}(A_2)$ per rendere i vincoli nei problemi computazionalmente risolvibili. Iniziamo l'analisi dall'insieme dei polinomi univariati nelle azioni, cioè dai vincoli (3.53a) e (3.54a): l'obiettivo sarà quello di esplicitare i coefficienti dei polinomi in modo da riformulare i relativi vincoli in programmazione semidefinita positiva.

Si definisca con d_s il grado del polinomio per univariato per un dato stato s, quindi si considerino i vettori delle azioni, indicizzati dal grado, definiti come segue:

$$[a_1]_{d_s}=[1,a_1,a_1^2,\ldots,a_1^{d_s}]^T,\quad [a_2]_{d_s}=[1,a_2,a_1^2,\ldots,a_2^{d_s}]^T.$$

I termini dei 2 polinomi relativi alla funzione di reward si potranno quindi riscrivere in forma vettoriale come segue:

$$\sum_{i=0}^{n_s}\sum_{j=0}^{m_s}r_{ij}(s)\mu_i(s){a_2}^j\ =\ \overline{\mu}(s)^TR(s)^T[a_2]_{d_s}. \tag{3.57a}$$

$$\sum_{i=0}^{n_s}\sum_{j=0}^{m_s}r_{ij}(s)\nu_j(s){a_1}^i\ =\ \overline{\nu}(s)^TR(s)^T[a_1]_{d_s}. \tag{3.57b}$$

Si vettorizzino poi i valori della funzione d'utilità per ogni stato del gioco come segue:

$$\overline{v}\ =\ [v(1),\ldots,v(k)]^T,\quad \overline{w}\ =\ [w(k+1),\ldots,w(N)]^T \tag{3.58}$$

Infine si riscrivano i termini dei 2 polinomi relativi alla funzione di probabilità di transizione in forma vettoriale come di seguito:

$$\sum_{s' \in S_2} p(s', s, a_2) w(s') \;=\; \overline{w}^T Q_{22}(s)^T [a_2]_{d_s}, \tag{3.59a}$$

$$\sum_{s' \in S_1} p(s', s, a_2) \widehat{v}(s') \;=\; \widehat{v}^T Q_{21}(s)^T [a_2]_{d_s}, \tag{3.59b}$$

$$\sum_{s' \in S_1} p(s', s, a_1) v(s') \;=\; \overline{v}^T Q_{11}(s)^T [a_1]_{d_s}, \tag{3.59c}$$

$$\sum_{s' \in S_2} p(s', s, a_1) \widehat{w}(s') \;=\; \widehat{w}^T Q_{12}(s)^T [a_1]_{d_s}, \tag{3.59d}$$

con $Q_{ef}(s)$ che rappresenta la matrice, parametrica negli indici e ed f, dei coefficienti della probabilità di transizione $p(s', s, a_e)$ con $s \in S_e$, ed $s' \in S_f$.

Grazie alle definizioni appena mostrate, ed applicando il Lemma 4 presentato nella Sezione 3.1.1, si può affermare che i polinomi univariati nelle azioni definiti rispettivamente in (3.53a) e (3.54a) possono essere riscritti come segue:

$$\mathcal{H}^*(Z_3(s)+(c+d)\frac{1}{2}(L_1 W_3(s) L_2^T + L_2 W_3(s) L_1^T) - (c*d) L_1 W_3(s) L_1^T - L_2 W_3(s) L_2^T) \;=$$

$$= \; -E(s)\overline{w} + R(s)\overline{\mu}^T + \beta Q_{22}(s)\overline{w}^T + \beta Q_{21}(s)\widehat{v}^T, \quad \forall s \in S_2 \tag{3.60}$$

$$\mathcal{H}^*(Z_4(s)+(c+d)\frac{1}{2}(L_1 W_4(s) L_2^T + L_2 W_4(s) L_1^T) - (c*d) L_1 W_4(s) L_1^T - L_2 W_4(s) L_2^T) \;=$$

$$= \; E(s)\overline{v} - R(s)\overline{\nu}(s)^T - \beta Q_{11}(s)\overline{v}^T - \beta Q_{12}(s)\widehat{w}^T, \quad \forall s \in S_1 \tag{3.61}$$

con $Z_3 \in \mathcal{S}^{d_s+1}$, $Z_4 \in \mathcal{S}^{d_s+1}$, $W_3 \in \mathcal{S}^{d_s}$ e $Z_4 \in \mathcal{S}^{d_s}$ matrici tutte semidefinite positive, ed $E(s) \in \mathbb{R}^{d_s+1}$ matrice che contiene tutti 0 a parte un 1 in posizione $(1, s)$.

Ora passiamo all'analisi degli insieme dei momenti delle misure di probabilità, relativi ai vincoli (3.53b) e (3.54b), e dei successivi vincoli sul momento di ordine 0, cioè (3.53c) e (3.54c). Grazie al Lemma 6 presentato nella Sezione 3.1.2, possiamo esprimere i vincoli presenti nel problema primale PP3, cioè (3.53b) e (3.53c), nell'intervallo $[c, d]$ delle azioni come segue:

$$e_1^T [\mu_0(s), \ldots, \mu_n(s)] \;=\; 1, \quad \forall s \in S_2 \tag{3.62a}$$

$$-c * \mathcal{H}([\mu_0(s), \ldots, \mu_n(s)]^T) + \mathcal{H}([\mu_1(s), \ldots, \mu_{n+1}(s)]^T) \;\succeq\; 0, \quad \forall s \in S_2 \tag{3.62b}$$

$$d * \mathcal{H}([\mu_0(s), \ldots, \mu_n(s)]^T) - \mathcal{H}([\mu_1(s), \ldots, \mu_{n+1}(s)]^T) \;\succeq\; 0, \quad \forall s \in S_2 \tag{3.62c}$$

Sempre grazie al Lemma 6, possiamo esprimere i vincoli presenti nel problema primale PP4, cioè (3.54b) e (3.54c), nell'intervallo $[c, d]$ delle azioni come segue:

$$e_1^T[\nu_0(s), \ldots, \nu_m(s)] \;=\; 1, \quad \forall s \in S_1 \tag{3.63a}$$

$$-c * \mathcal{H}([\nu_0(s), \ldots, \nu_m(s)]^T) + \mathcal{H}([\nu_1(s), \ldots, \nu_{m+1}(s)]^T) \;\succeq\; 0, \quad \forall s \in S_1 \tag{3.63b}$$

$$d * \mathcal{H}([\nu_0(s), \ldots, \nu_m(s)]^T) - \mathcal{H}([\nu_1(s), \ldots, \nu_{m+1}(s)]^T) \;\succeq\; 0, \quad \forall s \in S_1 \tag{3.63c}$$

Riformulazione dei problemi primali Ora possiamo costruire il primo problema primale formulato in programmazione SDP, chiamato SP3 ed equivalente a PP3 (3.53), tramite le espressioni (3.60) (3.62) come segue:

$$\max_{w(s)} \sum_{s \in S_2} w(s),$$

subject to

$$\mathcal{H}^*(Z_3(s) + (c+d)\frac{1}{2}(L_1 W_3(s) L_2^T + L_2 W_3(s) L_1^T) - (c*d) L_1 W_3(s) L_1^T - L_2 W_3(s) L_2^T) \;=$$

$$= \; -E(s)\overline{w} + R(s)\overline{\mu}^T + \beta Q_{22}(s)\overline{w} + \beta Q_{21}(s)\widehat{v}, \quad \forall s \in S_2 \tag{3.64a}$$

$$e_1^T[\mu_0(s), \ldots, \mu_n(s)] \;=\; 1, \quad \forall s \in S_2 \tag{3.64b}$$

$$-c * \mathcal{H}([\mu_0(s), \ldots, \mu_n(s)]^T) + \mathcal{H}([\mu_1(s), \ldots, \mu_{n+1}(s)]^T) \;\succeq\; 0, \quad \forall s \in S_2 \tag{3.64c}$$

$$d * \mathcal{H}([\mu_0(s), \ldots, \mu_n(s)]^T) - \mathcal{H}([\mu_1(s), \ldots, \mu_{n+1}(s)]^T) \;\succeq\; 0, \quad \forall s \in S_2 \tag{3.64d}$$

$$Z_3(s),\, W_3(s) \;\succeq\; 0, \quad \forall s \in S_2 \tag{3.64e}$$

La soluzione di questo problema SDP restituisce:

- il vettore dei valori del gioco $\overline{w}$, relativo agli stati $s \in S_2$ in cui il secondo giocatore decide le probabilità di transizione;

- il vettore dei momenti delle misure $[\mu_0(s), \ldots, \mu_{n+1}(s)]$ relativo al giocatore 1 per ogni stato $s \in S_2$ in cui è il secondo giocatore a controllare le probabilità di transizione.

Lemma 14. Sia $A = A_1 = A_2 = [c, d]$. Il problema di programmazione semidefinita positiva SP3 (3.64) risolve esattamente il problema di ottimizzazione polinomiale PP3 (3.53).

Dimostrazione. La disuguaglianza polinomiale (3.53a) nel problema PP3 ha come vettore dei coefficienti $-E(s)\overline{w} + R(s)\overline{\mu} + \beta Q_{22}(s)\overline{w} + \beta Q_{21}(s)\widehat{v}^T$, come mostrato dalle espressioni (3.57a) e (3.59). La dimostrazione si ha quindi per diretta conseguenza del Lemma 4, riguardante la rappresentazione di polinomi univariati non negativi nell'intervallo $[c, d]$ tramite formulazione in programmazione semidefinita positiva, applicato come mostrato nell'espressione (3.60), e del Lemma 6, riguardante la rappresentazione delle sequenze valide di momenti di misure di probabilità non negative supportate nell'intervallo $[c, d]$, applicato come mostrato nell'espressione (3.62). □

Il secondo problema primale formulato in programmazione SDP, chiamato SP4, equivalente a PP4 (3.54) e costruito tramite le espressioni (3.61) (3.63), sarà invece definito come segue:

$$\min_{v(s)} \sum_{s \in S_1} v(s),$$

subject to

$$\mathcal{H}^*(Z_4(s)+(c+d)\frac{1}{2}(L_1 W_4(s) L_2^T + L_2 W_4(s) L_1^T) - (c*d) L_1 W_4(s) L_1^T - L_2 W_4(s) L_2^T) =$$
$$= E(s)\overline{v} - R(s)\overline{\nu}(s)^T - \beta Q_{11}(s)\overline{v} - \beta Q_{12}(s)\widehat{w}, \quad \forall s \in S_1 \tag{3.65a}$$

$$e_1^T [\nu_0(s), \ldots, \nu_m(s)] = 1, \quad \forall s \in S_1 \tag{3.65b}$$

$$-c * \mathcal{H}([\nu_0(s), \ldots, \nu_m(s)]^T) + \mathcal{H}([\nu_1(s), \ldots, \nu_{m+1}(s)]^T) \succeq 0, \quad \forall s \in S_1 \tag{3.65c}$$

$$d * \mathcal{H}([\nu_0(s), \ldots, \nu_m(s)]^T) - \mathcal{H}([\nu_1(s), \ldots, \nu_{m+1}(s)]^T) \succeq 0, \quad \forall s \in S_1 \tag{3.65d}$$

$$Z_4(s),\, W_4(s) \succeq 0, \quad \forall s \in S_1 \tag{3.65e}$$

La soluzione di questo problema SDP restituisce:

- il vettore dei valori del gioco $\overline{v}$, relativo agli stati $s \in S_1$ in cui il primo giocatore decide le probabilità di transizione;
- il vettore dei momenti delle misure $[\nu_0(s), \ldots, \nu_{m+1}(s)]$ relativo al giocatore 2 per ogni stato $s \in S_1$ in cui è il primo giocatore a controllare le probabilità di transizione.

Lemma 15. Sia $A = A_1 = A_2 = [c, d]$. Il problema di programmazione semidefinita positiva SP4 (3.65) risolve esattamente il problema di ottimizzazione polinomiale PP4 (3.54).

Dimostrazione. La disuguaglianza polinomiale (3.54a) nel problema PP3 ha come vettore dei coefficienti $E(s)\overline{v} - R(s)\overline{\nu}(s)^T - \beta Q_{11}(s)\overline{v} - \beta Q_{12}(s)\widehat{w}$, come mostrato dalle espressioni (3.57b) e (3.59). La dimostrazione si ha quindi per

diretta conseguenza del Lemma 4, riguardante la rappresentazione di polinomi univariati non negativi nell'intervallo $[c, d]$ tramite formulazione in programmazione semidefinita positiva, applicato come mostrato nell'espressione (3.61), e del Lemma 6, riguardante la rappresentazione delle sequenze valide di momenti di misure di probabilità non negative supportate nell'intervallo $[c, d]$, applicato come mostrato nell'espressione (3.63). □

Dato che le soluzioni di questi 2 problemi primali restituiscono i vettori dei momenti delle misure, e grazie all'applicazione della procedura di ricostruzione delle strategie miste dai momenti indicata in Sezione 3.1.4, è infine possibile ricostruire le coppie supporto-pesi relative alle strategie miste di ogni giocatore in tutti gli stati in cui le transazioni sono governate dal giocatore avversario.

Riformulazione dei problemi duali Per completare la ricostruzione del problema, presentiamo ora la riformulazione relativa ai problemi duali PD3 (3.55) e PD4 (3.56). Per non avere una trattazione troppo ridondante con quanto appena fatto per i problemi primali, si eviterà di presentare i vari passaggi intermedi per ottenere le riformulazioni dei problemi duali.

Il primo problema duale formulato in programmazione SDP, chiamato SD3 ed equivalente a PD3 (3.55), sarà così riformulato:

$$\max_{\eta(s),\overline{\varepsilon}(s)} \sum_{s\in S_2} \Big[\eta(s) - \beta\overline{\varepsilon}(s)^T \Big(Q_{21}(s)\widehat{v}\Big)\Big]$$

subject to

$$\mathcal{H}^*(J_3(s)+(c+d)\frac{1}{2}(L_1K_3(s)L_2^T+L_2K_3(s)L_1^T)-(c*d)L_1K_3(s)L_1^T-L_2K_3(s)L_2^T) = \\ = -R(s)\overline{\varepsilon}(s) - E(s)\overline{\eta}(s), \quad \forall s \in S_2 \tag{3.66a}$$

$$-c * \mathcal{H}([\varepsilon_0(s), ..., \varepsilon_m(s)]^T) + \mathcal{H}([\varepsilon_1(s), ..., \varepsilon_{m+1(s)}]^T) \succeq 0, \quad \forall s \in S_2 \tag{3.66b}$$

$$d * \mathcal{H}([\varepsilon_0(s), ..., \varepsilon_m(s)]^T) - \mathcal{H}([\varepsilon_1(s), ..., \varepsilon_{m+1}(s)]^T) \succeq 0, \quad \forall s \in S_2 \tag{3.66c}$$

$$\sum_{s\in S_2} \Big(E(s) - \beta Q_{22}(s)\Big)^T \overline{\varepsilon}(s) = 1. \tag{3.66d}$$

$$J_3(s),\, K_3(s) \succeq 0, \quad \forall s \in S_2 \tag{3.66e}$$

La soluzione di questo problema duale SDP restituisce il vettore dei momenti non normalizzato $[\varepsilon_0(s), \ldots, \varepsilon_{m+1}(s)]$ relativo al giocatore 2 per ogni stato $s \in S_2$ in cui è lui stesso a controllare le probabilità di transizione.

Lemma 16. Sia $A = A_1 = A_2 = [c, d]$. Il problema di programmazione semidefinita positiva SD3 (3.66) risolve esattamente il problema di ottimizzazione polinomiale PD3 (3.55).

Dimostrazione. La disuguaglianza polinomiale (3.55a) nel problema PD3 ha come vettore dei coefficienti $-R(s)\overline{\varepsilon}(s) - E(s)\overline{\eta}(s)$. La dimostrazione si ha quindi per diretta conseguenza del Lemma 4, riguardante la rappresentazione di polinomi univariati non negativi nell'intervallo $[c, d]$ tramite formulazione in programmazione semidefinita positiva, e del Lemma 6, riguardante la rappresentazione delle sequenze valide di momenti di misure di probabilità non negative supportate nell'intervallo $[c, d]$. □

Il secondo problema duale formulato in programmazione SDP, chiamato SD4 ed equivalente a PD4 (3.56), sarà invece così riformulato:

$$\max_{\alpha(s),\overline{\epsilon}(s)} \sum_{s\in S_1} \Big[\alpha(s) + \beta\overline{\epsilon}(s)^T \Big(Q_{12}(s)\widehat{w}\Big)\Big]$$

subject to

$$\mathcal{H}^*(J_4(s)+(c+d)\frac{1}{2}(L_1K_4(s)L_2^T+L_2K_4(s)L_1^T)-(c*d)L_1K_4(s)L_1^T-L_2K_4(s)L_2^T) =$$
$$= R(s)^T\overline{\epsilon}(s) - E(s)\overline{\alpha}(s), \quad \forall s \in S_1 \tag{3.67a}$$

$$-c * \mathcal{H}([\epsilon_0(s), ..., \epsilon_n(s)]^T) + \mathcal{H}([\epsilon_1(s), ..., \epsilon_{n+1(s)}]^T) \succeq 0, \quad \forall s \in S_1 \tag{3.67b}$$

$$d * \mathcal{H}([\epsilon_0(s), ..., \epsilon_n(s)]^T) - \mathcal{H}([\epsilon_1(s), ..., \epsilon_{n+1}(s)]^T) \succeq 0, \quad \forall s \in S_1 \tag{3.67c}$$

$$\sum_{s\in S_1} \Big(E(s) - \beta Q_{11}(s)\Big)^T \overline{\epsilon}(s) = 1 \tag{3.67d}$$

$$J_4(s),\, K_4(s) \succeq 0, \quad \forall s \in S_1 \tag{3.67e}$$

La soluzione di questo secondo problema duale SDP restituisce il vettore dei momenti non normalizzato $[\epsilon_0(s), \ldots, \epsilon_{n+1}(s)]$ relativo al giocatore 1 per ogni stato $s \in S_1$ in cui è lui stesso a controllare le probabilità di transizione.

Lemma 17. Sia $A = A_1 = A_2 = [c, d]$. Il problema di programmazione semidefinita positiva SD4 (3.67) risolve esattamente il problema di ottimizzazione polinomiale PD4 (3.56).

Dimostrazione. La disuguaglianza polinomiale (3.56a) nel problema PD4 ha come vettore dei coefficienti $R(s)^T\overline{\epsilon}(s) - E(s)\overline{\alpha}(s)$. La dimostrazione si ha quindi per diretta conseguenza del Lemma 4, riguardante la rappresentazione di polinomi univariati non negativi nell'intervallo $[c, d]$ tramite formulazione in programmazione semidefinita positiva, e del Lemma 6, riguardante la rappresentazione delle sequenze valide di momenti di misure di probabilità non negative supportate nell'intervallo $[c, d]$. □

Dato che le soluzioni di questi 2 problemi duali restituiscono i vettori dei momenti non normalizzati, si applichi per prima cosa la procedura di normalizzazione illustrata dall'espressione 3.32 all'interno della Sezione 3.2.2; quindi, grazie all'applicazione della successiva procedura di ricostruzione delle strategie miste dai momenti indicata in Sezione 3.1.4, si ricostruiranno le coppie supporto-pesi relative alle strategie miste di ogni giocatore in tutti gli stati in cui le transazioni sono governate da loro stessi.

Ora che abbiamo tutti gli elementi che compongono il problema di risoluzione di un MDP con spazio delle azioni continuo e con proprietà di Swtiching Controller, possiamo passare alla fase di descrizione dell'algoritmo che permette di calcolarne l'equilibrio, sottolineandone anche le proprietà di convergenza ed ottimalità della soluzione.

3.4.4 Descrizione dell'algoritmo iterativo di risoluzione del problema Switching Controller

L'algoritmo di risoluzione di questo tipo di problema, che si ispira a quello mostrato alla fine della Sezione 3.1.4, permette di ottenere il vettore dei valori e le strategie miste dei giocatori nel gioco in un punto di equilibrio che non rappresenta propriamente un equilibrio minmax, ma è un ϵ-Equilibrio di Nash; questa scelta è dovuta al fatto che, poiché non esiste l'equivalente della soluzione di base ammissibile nella programmazione SDP, questo tipo di equilibrio permette di avere una condizione di termine dell'algoritmo che possiede anche delle garanzie sulla soluzione trovata, garanzie date dalla definizione dell'equilibrio stesso. Questo algoritmo si può dividere principalmente in 4 fasi: la prima è una fase di inizializzazione preiterativa, mentre le successive 3, consecutive tra loro, vengono iterate dall'algoritmo finché la condizione di termine posta sull'ϵ-Nash viene raggiunta.

Inizializzazione: calcolo dei fattori di normalizzazione In questa fase iniziale vengono calcolati i fattori di normalizzazione $\Delta(s)$ relativi ai valori di ogni stato del gioco: questi sono resi necessari dal fatto che, dalla Definizione 13 di ϵ-Nash fatta nella Sezione 2.3.3, sappiamo che la funzione di reward deve essere normalizzata, quindi di conseguenza anche la funzione d'utilità dovrà esserlo. Per farlo, bisognerebbe calcolare un doppio problema di ottimizzazione polinomiale bivariato: nel primo entrambi i giocatori dovrebbero agire per avvantaggiare al massimo l'utilità del giocatore 1, mentre nel secondo entrambi i giocatori dovrebbero agire per avvantaggiare al massimo l'utilità del giocatore 2, il tutto facendo variare le strategie di entrambi contemporaneamente. Data la complessità dell'operazione, viene

calcolato un fattore di normalizzazione equivalente, ma molto più semplice a livello computazionale. Questo fattore Δ, fissato uno stato $s \in S$, sarà così definito:

$$\Delta(s) = \frac{1-\beta}{\max\limits_{a_1,a_2} r(s,a_1,a_2)} \tag{3.68}$$

con $\max\limits_{a_1,a_2} r(s,a_1,a_2)$ che rappresenta il massimo valore che può assumere la funzione di reward variando contemporaneamente le strategie dei 2 giocatori. Lavorando sulla funzione di reward e non direttamente su quella d'utilità, infatti possiamo calcolare questi termini senza risolvere problemi di ottimizzazione particolarmente complessi.

Dopo la fase di calcolo dei fattori di normalizzazione per ogni stato del gioco, si fissa il valore $\epsilon > 0$ relativo all'ϵ-Nash da valutare, infine si inizializza il vettore $\overline{w}^0(\mathrm{s}) = 0$, $\forall s \in S_2$ per la prima iterazione dell'algoritmo.

Calcolo delle soluzioni delle coppie di SDP primali e duali Si risolva il problema primale SP4 (3.65) ed il corrispondente problema duale SD4 (3.67), assumendo $\widehat{w}(\mathrm{s}) = \overline{w}^t(s)$, $\forall sinS_2$, con t che rappresenta il numero di iterazioni già effettuate dall'algoritmo. Sia $(\overline{v}^t(s), \overline{\mu}^t(s), \overline{\nu}^t(s))$, per ogni $s \in S_1$, la soluzione ottima a questa coppia di problemi. Si risolva successivamente il problema primale SP3 (3.64) ed il corrispondente problema duale SD3 (3.66), assumendo $\widehat{v}(\mathrm{s}) = \overline{v}^t(\mathrm{s})$. Sia $(\overline{w}^{t+1}(s), \overline{\mu}^{t+1}(s), \overline{\nu}^{t+1}(s))$, per ogni $s \in S_2$, la soluzione ottima a questa seconda coppia di problemi.

Calcolo del valore reale degli stati Le soluzioni ottenute dalle 2 coppie di problemi SDP generano un vettore dei valori del gioco ed un profilo di strategie miste, uno per ogni stato del gioco, per i 2 giocatori, che definiremo come $(\overline{\mu}_t, \overline{\nu}_t)$. Questo vettore però non rappresenta i veri valori del gioco nel caso si giochi il profilo di strategie calcolato, in quanto si basa sull'assunzione di conoscere il valore di alcuni stati. In questa fase dell'algoritmo quindi, per avere il vero vettore dei valori del gioco $\overline{V}_t$ relativo al profilo di strategie calcolato, dovremo risolvere un semplice sistema lineare ad N equazioni ed N incognite, che saranno i valori della funzione d'utilità nei vari stati, definito come segue:

$$V_t(s) = R(s) + \beta \sum_{s' \in S} p(s', s) V_t(s'), \quad \forall s \in S.$$

Verifica della condizione di termine dell'algoritmo In quest'ultima fase calcoleremo la condizione di termine dell'algoritmo seguendo questi passaggi:

1. si calcoli il vettore V_{BR1} dei valori degli stati derivanti dalla risoluzione dell'SDP di Best Response SBR1 mostrato nell'espressione (3.41) presente nella Sezione 3.3.2 fissando la strategia $\overline{\mu}_t$ del secondo giocatore;

2. si calcoli il vettore V_{BR2} dei valori degli stati derivanti dalla risoluzione dell'SDP di Best Response SBR2 mostrato nell'espressione (3.45) presente nella Sezione 3.3.3 fissando la strategia $\overline{\nu}_t$ del primo giocatore;

3. si calcolino i vettori differenza $D_1 = V_{BR1} - V_t$ e $D_2 = V_t - V_{BR2}$, quindi si normalizzano questi vettori moltiplicando ogni elemento per corrispondente fattore di normalizzazione $\Delta(s)$, ottenendo i vettori normalizzati D_1^{norm} e D_2^{norm}.

4. infine si consideri il valore massimo tra tutti gli elementi dei vettori normalizzati D_1^{norm} e D_2^{norm}: tale valore rappresenta il massimo incremento di utilità che può ottenere uno dei 2 giocatori quando questo cambia la propria strategia, mentre l'avversario mantiene la sua. Se questo valore è inferiore a quello del nostro ϵ, allora il profilo di strategie appena trovato dall'algoritmo $(\overline{\mu}_t, \overline{\nu}_t)$ è un ϵ-Equilibrio di Nash, quindi l'algoritmo termina; in caso contrario, si incrementa t di un'unità e si riparte con una nuova iterazione dell'algoritmo.

Descritto nei particolari il procedimento iterativo dell'algoritmo, possiamo ora farne una breve analisi di convergenza ed ottimalità della soluzione trovata, basata sugli stessi risultati presenti in [6].

Per prima cosa dimostriamo che il vettore dei valori restituito dall'algoritmo è effettivamente il vero vettore dei valori del gioco.

Teorema 15. *si supponga di avere le seguenti 2 coppie di vettori di valori del gioco e strategie:*

$$v(s), \nu(s, a_2), \quad \forall s \in S_1,\, a_2 \in A_2$$

$$w(s), \mu(s, a_1), \quad \forall s \in S_2,\, a_1 \in A_1$$

dove $\Big(v(s), \nu(s, a_2), \quad \forall s \in S_1,\, a_2 \in A_2\Big)$ *è ottima per il problema SP4 descritto in (3.65) quando* $\widehat{w}(s) = w(s)$*, con* $s \in S_2$*, e dove* $\Big(w(s), \mu(s, a_1), \quad \forall s \in S_2,\, a_1 \in A_1\Big)$ *è ottima per il problema SP3 descritto in (3.64) quando* $\widehat{v}(s) = v(s)$*, con* $s \in S_1$*. Allora* $\Big(v(1), v(2), \ldots, v(k), w(k+1), w(k+2), \ldots, w(N)\Big)$ *è il vettore dei valori del gioco.*

Dimostrazione. La dimostrazione di questo teorema è mostrata in [6]. □

Per quanto riguarda quindi la convergenza dell'algoritmo, che equivale anche alla restituzione del vettore ottimo dei valori del gioco (v^*, w^*) e della coppia di strategie ottime dei giocatori, questa è garantita, in un numero finito di passi, dal seguente teorema:

Teorema 16.

$$\lim_{t \longmapsto \infty} (v^t, w^t) = (v^*, w^*).$$

Dimostrazione. La dimostrazione di questo teorema è mostrata in [6]. □

Per quanto riguarda invece l'ottimalità della soluzione restituita dall'algoritmo, questa è garantita dal seguente teorema:

Teorema 17. *Siano μ^* e ν^* le strategie miste dei giocatori calcolate dall'algoritmo, e sia $\overline{v} = \overline{v}(\mu^*, \nu^*)$ il vettore dei valori degli stati con tali strategie. Allora $\overline{v}$ soddisfa la seguente la saddle-point condition (2.7), cioè vale la seguente disuguaglianza:*

$$\overline{v}(\mu, \nu^*) - \epsilon e_S \leq \overline{v}(\mu^*, \nu^*) \leq \overline{v}(\mu^*, \nu) + \epsilon e_S$$

per ogni coppia di strategie miste (μ, ν), dove $e_S \in \mathbb{R}^S$ è un vettore composto unicamente da 1.

Dimostrazione. La dimostrazione di questo teorema è mostrata in [6]. □

3.5 Analisi dell'errore nella risoluzione di giochi non-polinomiali

Nelle precedenti sezioni sono state presentate forme di risoluzione per giochi stocastici, in particolare per gli MDP, con spazio discreto degli stati e spazio continuo delle azioni nel caso in cui abbiano sia proprietà di Single Controller sia proprietà di Switching Controller. Un'altra particolarità dei giochi da noi analizzati e risolti è che sia la funzione di reward che la funzione di probabilità di transizione tra uno stato e l'altro del gioco sono di tipo polinomiale nelle variabili delle azioni dei giocatori, tuttavia non molti giochi sono effettivamente formulati tramite funzioni polinomiali, quindi è interessante sapere qual'è l'errore teorico che si ha trovando una soluzione basata su funzioni polinomiali, che sono però approssimazioni di funzioni non polinomiali. Inizieremo quindi questa sezione presentando una tecnica molto semplice ed efficiente di approssimazione di funzioni continue in forma non polinomiale in funzioni in forma polinomiale chiamata Chebyshev Approximation. L'analisi che svolgeremo sarà poi basata sulla valutazione dell'errore di approssimazione che si ha quando si considerano giochi basati su funzioni continue in forma non polinomiale che vengono risolti approssimando quest'ultime con funzioni polinomiali. I risultati di questa sezione derivano principalmente dalle analisi teoriche presenti in [19] e [20].

3.5.1 Chebyshev Approximation

Nella classe dei giochi infiniti, cioè quei giochi caratterizzati da almeno una componente con un numero infinito e non numerabile di elementi, i giochi cosiddetti polinomiali ne rappresentano solo una ristretta sottoclasse. Questo è dovuto al fatto che la maggior parte dei giochi di questa classe ha funzioni di reward e/o funzioni di transizione che sono di tipo continuo ma non polinomiale. Dalla teoria dell'approssimazione, però, sappiamo che è possibile approssimare una funzione continua definita su un sottoinsieme compatto di uno spazio euclideo m-dimensionale tramite un'equivalente funzione polinomiale approssimata con accuratezza arbitraria di grado n, introducendo un certo errore di approssimazione. Una delle tecniche più usate per questo tipo di approssimazioni è la cosiddetta Chebyshev Approximation, presentata per la prima volta in [20], che andremo ora a descrivere.

Questa tecnica definisce per prima cosa una particolare forma di polinomio, chiamato Polinomio di Chebyshev appunto, denotato dal simbolo $T_n(x)$, con x ed n che rappresentano rispettivamente la variabile ed il grado

del polinomio, e definito esplicitamente dalla seguente espressione:

$$T_n(x) \;=\; cos(n * arccos(x)).$$

In un primo momento questa può sembrare l'espressione di una funzione trigonometrica, tuttavia è possibile esprimerla come un'espressione polinomiale esplicita. Infatti, grazie all'uso di particolari identità trigonometriche, può essere riscritta, in base al grado del polinomio, come segue:

$$T_0(x) \;=\; 1;$$

$$T_1(x) \;=\; x;$$

$$T_2(x) \;=\; 2x^2 - 1;$$

$$T_{n+1}(x) \;=\; 2xT_n(x) - T_{n-1}(x), \quad \forall n \geq 1.$$

I polinomi di Chebyshev posseggono alcune importanti proprietà nell'intervallo $[-1, 1]$:

1. proprietà di ortogonalità continua rispetto al fattore $(1 - x^2)^{-\frac{1}{2}}$, cioè

$$\int_{-1}^{1} \frac{T_i(x)T_j(x)}{\sqrt{1-x^2}}dx \;=\; \begin{cases} 0 & i \neq j \\ \frac{\pi}{2} & i = j \neq 0 \\ \pi & i = j = 0 \end{cases};$$

2. il polinomio di Chebyshev di grado n, cioè $T_n(x)$, ha n zeri nell'intervallo specificato, zeri che sono localizzati nei punti

$$x \;=\; cos\Big(\frac{\pi(k - \frac{1}{2})}{n}\Big), \quad con\, k = 1, 2, \ldots, n; \tag{3.70}$$

3. il polinomio di Chebyshev di grado n, cioè $T_n(x)$, ha $n+1$ punti estremi nell'intervallo specificato, punti che assumono i valori

$$x \;=\; cos\Big(\frac{\pi k}{n}\Big), \quad con\, k = 0, 1, \ldots, n;$$

In tutti i punti di massimo si avrà $T_n(x) = 1$, mentre i tutti quelli di minimo si avrà $T_n(x) =$ -1. Questa è probabilmente la proprietà più utile per approssimare funzioni continue in funzioni polinomiali;

4. se x_k (con $k = 1, ..., m$) sono gli m zeri di $T_m(x)$, definiti come nell'espressione 3.70, e $i, j < m$, allora il polinomio di Chebyshev soddisfa la seguente relazione di ortogonalità discreta:

$$\sum_{k=1}^{m} T_i(x_k)T_j(x_k) \;=\; \begin{cases} 0 & i \neq j \\ \frac{m}{2} & i = j \neq 0\,; \\ m & i = j = 0 \end{cases}$$

Grazie alle proprietà appena mostrate, è possibile presentare il seguente teorema, che mostra la formula di approssimazione di una funzione arbitraria in una funzione polinomiale.

Teorema 18. *Sia $f(x)$ una funzione arbitraria definita nell'intervallo $[-1, 1]$. Si definiscano i coefficienti c_j, con $j = \{0, ..., N-1\}$, come segue:*

$$\begin{aligned} c_j &= \frac{2}{N}\sum_{k=1}^{N} f(x_k)T_j(x_k) \\ &= \frac{2}{N}\sum_{k=1}^{N} f\Big[cos\Big(\frac{\pi(k-\frac{1}{2})}{N}\Big)\Big]cos\Big(\frac{\pi j(k-\frac{1}{2})}{N}\Big). \end{aligned} \tag{3.71}$$

Allora la seguente formula di approssimazione:

$$f(x) \;\approx\; \Big[\sum_{k=0}^{N-1} c_k T_k(x)\Big] - \frac{1}{2}c_0 \tag{3.72}$$

è esatta per valori di x uguali agli N zeri di $T_N(x)$.

L'equazione (3.72) rappresenta quindi il polinomio, di un fissato grado N, nella variabile x che approssima una funzione arbitraria $f(x)$ nell'intervallo $[-1, 1]$. Se si considera un valore molto elevato di N, si può dire che (3.72) è virtualmente un'approssimazione perfetta di $f(x)$.

La scelta di questo metodo di approssimazione è dovuta al fatto che permette di troncare il polinomio di approssimazione ad un grado $m \ll N$ a piacere, garantendo che sia la migliore approssimazione possibile per il grado m specificato. In particolare, si consideri la seguente funzione di approssimazione troncata:

$$f(x) \;\approx\; \Big[\sum_{k=0}^{m-1} c_k T_k(x)\Big] - \frac{1}{2}c_0 \tag{3.73}$$

con i coefficienti c_j definiti come nell'espressione (3.71). Poiché i valori dei polinomi $T_k(x)$ sono tutti limitati tra $[-1, 1]$, la differenza tra le due espressioni (3.72) e (3.73) sarà limitata alla somma dei coefficienti mancanti c_k, con $k = \{m, ..., N-1\}$. Di conseguenza, poiché i valori dei coefficienti sono progressivamente decrescenti, l'errore d'approssimazione sarà dominato dal termine $c_m T_m(x)$, che non è altro che una funzione oscillante com $m+1$ punti estremi distribuiti in modo regolare sull'intervallo $[-1, 1]$. Questa proprietà di regolarizzazione dell'espansione dell'errore fà del Chebyshev Approximation una scelta molto efficiente e facile da calcolare.

Prima di concludere la trattazione di questo metodo di approssimazione, mostriamo come generalizzare la formula di approssimazione in (3.72) nel caso si lavori con funzioni in un intervallo generico $[c, d]$. Si definiscano 2 valori c, d tali che $c < d$. Affinché il range di approssimazione passi dall'intervallo $[-1, 1]$ ad un intervallo arbitrario $[c, d]$, si applicherà il seguente cambio di variabile

$$y \equiv \frac{x - \frac{1}{2}(d+c)}{\frac{1}{2}(d-c)}$$

quindi si approssimerà la funzione $f(x)$ tramite polinomi di Chebyshev nella nuova variabile y.

3.5.2 Analisi dell'errore teorico nell'approssimazione di giochi non polinomiali

Descritto il metodo di approssimazione polinomiale utilizzato, possiamo iniziare ad analizzare l'errore di approssimazione relativo alle soluzioni d'equilibrio restituite dai metodi presentati nelle precedenti sezioni, con particolare riferimento alle tecniche relative ai giochi stocastici con spazio delle azioni continuo presentate in Sezione 3.4.

Si definisca con $\mathcal{G}$ un gioco stocastico a due giocatori a somma zero, con spazio degli stati finito S, con funzione di reward continua arbitraria $r(s, a_1, a_2)$, e con funzione di probabilità di transizione da uno stato all'altro continua arbitraria $p(s', s, a)$. Siano $\tilde{r}(s, a_1, a_2)$ e $\tilde{p}(s', s, a)$ le funzioni approssimate polinomiali rispettivamente della funzione di reward e di quella di transizione. Date queste nozioni, possiamo definire i seguenti limiti massimi di errore di approssimazione:

$$\|r(s, a_1, a_2) - \tilde{r}(s, a_1, a_2)\|_\infty \leq \delta, \quad \forall s \in S \tag{3.74}$$

$$\|p(s', s, a) - \tilde{p}(s', s, a)\|_\infty \leq \rho, \quad \forall s, s' \in S \tag{3.75}$$

Si definisca poi con $v(s)$ il valore della funzione d'utilità reale nello stato s, e con $\tilde{v}(s)$ la sua versione approssimata. Si definisca infine con $\epsilon_s - NE$ un profilo di strategie, ricavato usando l'algoritmo descritto in Sezione 3.4.4, tale che nessuno dei due agenti possa guadagnare più di ϵ nello stato s deviando dalla strategia scelta. Possiamo ora presentare il primo teorema sull'errore di approssimazione.

Teorema 19. *Dato un gioco stocastico $\mathcal{G}$ ed uno stato pozzo $\underline{s}$, il profilo di strategie $(\tilde{\sigma}_1, \tilde{\sigma}_2)$, corrispondente al NE del gioco quando la funzione d'utilità è $\tilde{v}(\underline{s})$, è un $\epsilon_s - NE$, con $\epsilon_s \leq 2\delta$, quando la funzione d'utilità è $v(\underline{s})$*

Dimostrazione. Sia σ_1^* la strategia di Best Response del primo giocatore alla strategia σ_2 del secondo giocatore. Possiamo calcolare il limite superiore rispetto alla perdita attesa dell'utilità del primo giocatore come segue:

$$\begin{aligned}
\epsilon_{\underline{s}} &= v(\underline{s}, \sigma_1^*, \tilde{\sigma_2}) - v(\underline{s}, \tilde{\sigma_1}, \tilde{\sigma_2}) \\
&= r(\underline{s}, \sigma_1^*, \tilde{\sigma_2}) - r(\underline{s}, \tilde{\sigma_1}, \tilde{\sigma_2}) \\
&\leq r(\underline{s}, \sigma_1^*, \tilde{\sigma_2}) - \tilde{r}(\underline{s}, \sigma_1^*, \tilde{\sigma_2}) + \tilde{r}(\underline{s}, \tilde{\sigma_1}, \tilde{\sigma_2}) - r(\underline{s}, \tilde{\sigma_1}, \tilde{\sigma_2}) \\
&\leq |r(\underline{s}, \sigma_1^*, \tilde{\sigma_2}) - \tilde{r}(\underline{s}, \sigma_1^*, \tilde{\sigma_2})| + |\tilde{r}(\underline{s}, \tilde{\sigma_1}, \tilde{\sigma_2}) - r(\underline{s}, \tilde{\sigma_1}, \tilde{\sigma_2})| \\
&\leq \|r(\underline{s}, \sigma_1^*, \tilde{\sigma_2}) - \tilde{r}(\underline{s}, \sigma_1^*, \tilde{\sigma_2})\|_\infty + \|\tilde{r}(\underline{s}, \tilde{\sigma_1}, \tilde{\sigma_2}) - r(\underline{s}, \tilde{\sigma_1}, \tilde{\sigma_2})\|_\infty \\
&\leq 2\delta
\end{aligned}$$

Si applichi lo stesso ragionamento per il secondo giocatore. Il teorema è così provato. □

Ora dobbiamo generalizzare il precedente teorema nel caso si consideri uno stato generico s. Per fare questo, presenteremo alcune nozioni intermedie basate sull'assunzione che ci sia, di volta in volta, solo una funzione approssimata in gioco: quando assumeremo $\delta = 0$, allora avremo solo la funzione di transizione approssimata, mentre quando assumeremo $\rho = 0$, allora avremo solo la funzione di reward approssimata. Iniziamo considerando il caso in cui $\rho = 0$, e $\delta > 0$. Dato uno stato $s \in S$, definiamo con $l(s)$ il massimo numero di azioni da effettuare per raggiungere uno stato pozzo dallo stato s.

Lemma 18. Si assuma $\rho = 0$. Dato un gioco stocastico tree-based $\mathcal{G}$, per ogni stato s e per ogni profilo di strategie (σ_1, σ_2), la differenza massima in

valore assoluto tra l'utilità del primo giocatore calcolata rispetto a v e l'utilità dello stesso calcolata rispetto a $\tilde{v}$ è inferiore o uguale a $\delta\frac{1-\gamma^{l(s)+1}}{1-\gamma}$, cioè

$$\|v(s,\sigma_1,\sigma_2) - \tilde{v}(s,\sigma_1,\sigma_2)\|_\infty \leq \delta\frac{1-\gamma^{l(s)+1}}{1-\gamma}$$

Dimostrazione. Grazie al limite d'errore sull'approssimazione della funzione di reward definito in (3.74) possiamo fare i seguenti passaggi:

$$\begin{aligned}\|v(s,\sigma_1,\sigma_2) - \tilde{v}(s,\sigma_1,\sigma_2)\|_\infty &\leq \|r(s,\sigma_1,\sigma_2) - \tilde{r}(s,\sigma_1,\sigma_2)\|_\infty \\ &\quad + \gamma\|\sum_{s'} p(s',s,\sigma_c)\Big(v(s',\sigma_1,\sigma_2) - \tilde{v}(s',\sigma_1,\sigma_2)\Big)\|_\infty \\ &\leq \delta + \gamma \max_{s'}\|v(s',\sigma_1,\sigma_2) - \tilde{v}(s',\sigma_1,\sigma_2)\|_\infty\end{aligned}$$

Applicando ricorsivamente la formula appena presentata partendo da uno stato pozzo $\underline{s}$ ad uno stato s si otterrà
$\|v(s,\sigma_1,\sigma_2) - \tilde{v}(s,\sigma_1,\sigma_2)\|_\infty \leq \delta\sum_{i=0}^{l(s)}(\gamma)^i = \delta\frac{1-\gamma^{l(s)+1}}{1-\gamma}$ □

Ora possiamo estendere il Teorema 19 alla classe dei giochi tree-based come segue.

Teorema 20. *Si assuma $\rho = 0$. Dato un gioco stocastico tree-based $\mathcal{G}$ ed uno stato sv arbitrario, il profilo di strategie $(\tilde{\sigma_1},\tilde{\sigma_2})$, relativo ad un NE di un sottogioco nel quale il nodo di partenza è s e dove la funzione d'utilità è $\tilde{v}$, è un $\epsilon_s - NE$, con $\epsilon_s \leq 2\delta\frac{1-\gamma^{l(s)+1}}{1-\gamma}$, di un sottogioco dove la funzione d'utilità è v.*

Dimostrazione. Per ogni stato s' nel sottogioco dove s è il nodo di partenza, sia σ_1^* la miglior strategia, calcolata tramite backward induction, del primo giocatore rispetto alla strategia fissata σ_2 del secondo giocatore. Ripetendo quanto fatto nel Teorema 19, ed applicando il Lemma 18, possiamo affermare quanto segue:

$$\begin{aligned}\epsilon_s &= v(s,\sigma_1^*,\tilde{\sigma_2}) - v(s,\tilde{\sigma_1},\tilde{\sigma_2}) \\ &\leq v(s,\sigma_1^*,\tilde{\sigma_2}) - \tilde{v}(s,\sigma_1^*,\tilde{\sigma_2}) + \tilde{v}(s,\tilde{\sigma_1},\tilde{\sigma_2}) - v(s,\tilde{\sigma_1},\tilde{\sigma_2}) \\ &\leq \|v(s,\sigma_1^*,\tilde{\sigma_2}) - \tilde{v}(s,\sigma_1^*,\tilde{\sigma_2})\|_\infty + \|\tilde{v}(s,\tilde{\sigma_1},\tilde{\sigma_2}) - v(s,\tilde{\sigma_1},\tilde{\sigma_2})\|_\infty \\ &\leq 2\delta\frac{1-\gamma^{l(s)+1}}{1-\gamma}\end{aligned}$$

□

Estendiamo quindi il precedente teorema alla classe dei giochi graph-based tramite il seguente corollario.

Corollario 1. Si assuma $\rho = 0$. Il limite all'errore d'approssimazione definito nel teorema 20 può essere generalizzato ai giochi stocastici $\mathcal{G}$ graph-based come segue:

$$\epsilon \leq \lim_{l(s) \longmapsto +\infty} 2\delta \frac{1-\gamma^{l(s)+1}}{1-\gamma} = \frac{2\delta}{1-\gamma}$$

Si osservi che il limite appena mostrato è indipendente dallo stato.

Passiamo ora a considerare il caso in cui $\delta = 0$, e $\rho > 0$. Si definisca il termine q_s, che indica il numero di stati s' raggiungibili dallo stato s con una singola azione diretta. Si definiscano poi gli insiemi $FH(s)$ e $LH(s)$, con $FH(s)$ che rappresenta l'insieme dei $[\frac{q_s}{2}]$ stati con il maggiore valore di utilità $\tilde{v}(s', \tilde{\sigma_1}, \tilde{\sigma_2})$, e con LH(s) che rappresenta l'insieme dei $[\frac{q_s}{2}]$ stati con il minor valore di utilità $\tilde{v}(s', \tilde{\sigma_1}, \tilde{\sigma_2})$ Si definisca infine l'espressione $\Phi(s) = \sum_{s' \in FH(s)} \tilde{v}(s', \tilde{\sigma_1}, \tilde{\sigma_2}) - \sum_{s' \in LH(s)} \tilde{v}(s', \tilde{\sigma_1}, \tilde{\sigma_2})$, e $\overline{\Phi} = max_s\{\Phi(s)\}$.

Lemma 19. Si assuma $\delta = 0$. Dato un gioco stocastico tree-based $\mathcal{G}$, per ogni stato s e per ogni profilo di strategie (σ_1, σ_2), la differenza massima in valore assoluto tra l'utilità del primo giocatore calcolata rispetto a v e l'utilità dello stesso calcolata rispetto a $\tilde{v}$ è inferiore o uguale a $\gamma\rho\overline{\Phi}\frac{1-\gamma^{l(s)+1}}{1-\gamma}$, cioè

$$\left\| v(s, \sigma_1, \sigma_2) - \tilde{v}(s, \sigma_1, \sigma_2) \right\|_\infty \leq \gamma\rho\overline{\Phi}\frac{1-\gamma^{l(s)+1}}{1-\gamma}$$

Dimostrazione. Grazie al limite d'errore sull'approssimazione della funzione

di transizione definito in (3.75) possiamo fare i seguenti passaggi:

$$\begin{aligned}
\|v(s,\sigma_1,\sigma_2) - \tilde{v}(s,\sigma_1,\sigma_2)\|_\infty \leq\ & \|r(s,\sigma_1,\sigma_2) - \tilde{r}(s,\sigma_1,\sigma_2)\|_\infty \\
& + \gamma\|\sum_{s'}\Big(p(s',s,\sigma_c)v(s',\sigma_1,\sigma_2) - \tilde{p}(s',s,\sigma_c)\tilde{v}(s',\sigma_1,\sigma_2)\Big)\|_\infty \\
=\ & \gamma\|\sum_{s'}\Big(p(s',s,\sigma_c)v(s',\sigma_1,\sigma_2) - \tilde{p}(s',s,\sigma_c)\tilde{v}(s',\sigma_1,\sigma_2)\Big)\|_\infty \\
& + \gamma\|\sum_{s'}\Big(p(s',s,\sigma_c)v(s',\sigma_1,\sigma_2) - p(s',s,\sigma_c)\tilde{v}(s',\sigma_1,\sigma_2)\Big)\|_\infty \\
\leq\ & \gamma\|\sum_{s'}p(s',s,\sigma_c)\Big(v(s',\sigma_1,\sigma_2) - \tilde{v}(s',\sigma_1,\sigma_2)\Big)\|_\infty \\
& + \gamma\|\sum_{s'}\Big(p(s',s,\sigma_c) - \tilde{p}(s',s,\sigma_c)\Big)\tilde{v}(s',\sigma_1,\sigma_2)\|_\infty \\
\leq\ & \gamma\, max_{s'}\|v(s,\sigma_1,\sigma_2) - \tilde{v}(s,\sigma_1,\sigma_2)\|_\infty + \gamma\rho\Phi(s) \\
\leq\ & \gamma\, max_{s'}\|v(s,\sigma_1,\sigma_2) - \tilde{v}(s,\sigma_1,\sigma_2)\|_\infty + \gamma\rho\overline{\Phi}(s)
\end{aligned}$$

Applicando ricorsivamente la formula appena presentata partendo da uno stato pozzo $\underline{s}$ ad uno stato s si otterrà
$\|v(s,\sigma_1,\sigma_2) - \tilde{v}(s,\sigma_1,\sigma_2)\|_\infty \leq \gamma\rho\overline{\Phi}\frac{1-\gamma^{l(s)+1}}{1-\gamma}$ □

Teorema 21. *Si assuma $\delta = 0$. Dato un gioco stocastico tree-based $\mathcal{G}$ ed uno stato s arbitrario, il profilo di strategie $(\tilde{\sigma_1}, \tilde{\sigma_2})$, relativo ad un NE di un sottogioco nel quale il nodo di partenza è s e dove la funzione d'utilità è $\tilde{v}$, è un $\epsilon_s - NE$, con $\epsilon_s \leq 2\gamma\rho\overline{\Phi}\frac{1-\gamma^{l(s)+1}}{1-\gamma}$, di un sottogioco dove la funzione d'utilità è v.*

Dimostrazione. Ripetendo quanto fatto nel Teorema 19, ed applicando il Lemma 19, possiamo affermare quanto segue:

$$\begin{aligned}
\epsilon_s =\ & v(s,\sigma_1^*,\tilde{\sigma_2}) - v(s,\tilde{\sigma_1},\tilde{\sigma_2}) \\
\leq\ & v(s,\sigma_1^*,\tilde{\sigma_2}) - \tilde{v}(s,\sigma_1^*,\tilde{\sigma_2}) + \tilde{v}(s,\tilde{\sigma_1},\tilde{\sigma_2}) - v(s,\tilde{\sigma_1},\tilde{\sigma_2}) \\
\leq\ & \|v(s,\sigma_1^*,\tilde{\sigma_2}) - \tilde{v}(s,\sigma_1^*,\tilde{\sigma_2})\|_\infty + \|\tilde{v}(s,\tilde{\sigma_1},\tilde{\sigma_2}) - v(s,\tilde{\sigma_1},\tilde{\sigma_2})\|_\infty \\
\leq\ & 2\gamma\rho\overline{\Phi}\frac{1-\gamma^{l(s)+1}}{1-\gamma}
\end{aligned}$$

□

Come già fatto precedentemente, estendiamo il precedente teorema alla classe dei giochi graph-based tramite il seguente corollario.

Corollario 2. Si assuma $\delta = 0$. Il limite all'errore d'approssimazione definito nel Teorema 21 può essere generalizzato ai giochi stocastici $\mathcal{G}$ graph-based come segue:

$$\epsilon \leq \lim_{l(s) \longmapsto +\infty} 2\gamma\rho\overline{\Phi}\frac{1-\gamma^{l(s)+1}}{1-\gamma} = \frac{2\gamma\rho\overline{\Phi}}{1-\gamma}$$

Si osservi che anche il limite appena mostrato è indipendente dallo stato.

Ora che abbiamo presentato tutte queste nozioni, possiamo unirle per presentare il caso più generale in cui si approssimino entrambe le funzioni del gioco, cioè il caso in cui si ha $\delta, \rho \geq 0$.

Teorema 22. *Dato un gioco stocastico tree-based $\mathcal{G}$ ed uno stato s arbitrario, il profilo di strategie $(\tilde{\sigma_1}, \tilde{\sigma_2})$, relativo ad un NE di un sottogioco nel quale il nodo di partenza è s e dove la funzione d'utilità è $\tilde{v}$, è un $\epsilon_s - NE$, con $\epsilon_s \leq 2\Big(\delta + \gamma\rho\overline{\Phi}\Big)\frac{1-\gamma^{l(s)+1}}{1-\gamma}$, di un sottogioco dove la funzione d'utilità è v.*

Dimostrazione. La dimostrazione di questo teorema deriva dalla proprietà di additività dei 2 limiti definiti nei Teoremi 20 e 21. □

Estendiamo nuovamente il precedente teorema alla classe dei giochi graph-based tramite il seguente corollario.

Corollario 3. Il limite all'errore d'approssimazione definito nel teorema 22 può essere generalizzato ai giochi stocastici $\mathcal{G}$ graph-based come segue:

$$\epsilon \leq \lim_{l(s) \longmapsto +\infty} 2\Big(\delta + \gamma\rho\overline{\Phi}\Big)\frac{1-\gamma^{l(s)+1}}{1-\gamma} = \frac{2\Big(\delta+\gamma\rho\overline{\Phi}\Big)}{1-\gamma}$$

Si osservi che anche il limite appena mostrato è indipendente dallo stato.

Concludiamo infine questa trattazione presentando la valutazione di un $\tilde{\epsilon}$-Equilibrio di Nash calcolato su un gioco approssimato $\tilde{\mathcal{G}}$ relativamente al suo uso diretto nel gioco originale $\mathcal{G}$. Questo caso quindi ci riconduce finalmente alla valutazione della soluzione di equilibrio approssimato che viene restituiva tramite l'algoritmo presentato in Sezione 3.4.4.

Teorema 23. *Dato un gioco stocastico $\mathcal{G}$ ed uno stato s arbitrario, il profilo di strategie $(\tilde{\sigma_1}, \tilde{\sigma_2})$, relativo ad un epsilon-NE del gioco approssimato $\tilde{\mathcal{G}}$, è un ϵ^*-NE del gioco originale $\mathcal{G}$, tale che*

$$\epsilon^* \leq max_s\{\epsilon_s\} + \tilde{\epsilon} \tag{3.76}$$

Dimostrazione. La dimostrazione di questo teorema deriva dalla proprietà di additività dei 2 limiti definiti nei Teoremi 20 e 21 e dalla definizione di ϵ-Equilibrio di Nash presentata in Sezione 2.3.3. □

Capitolo 4

Giochi Polinomiali con Spazio degli Stati Continuo

Nel precedente capitolo è stata presentata una particolare sottoclasse di giochi infiniti chiamata Giochi Polinomiali, e relativamente a questa sono stati mostrati alcuni approcci di risoluzione per problemi MDP caratterizzati da funzioni in forma polinomiale, quindi con spazio continuo delle azioni. Altra caratteristica fondamentale di questi problemi era l'essere definiti in uno spazio degli stati discreto, cioè con un numero di elementi finito e numerabile. Negli ultimi anni però, è aumentato l'interesse scientifico e di ricerca su problemi MDP caratterizzati da uno spazio degli stati continuo in quanto questi modelli di gioco rappresentano meglio situazioni del mondo reale. In questo capitolo quindi verrà estesa la trattazione degli MDP con funzioni in forma polinomiale e spazio degli stati continuo ad uno spazio degli stati continuo, quindi con un numero infinito e non numerabile di elementi.

In particolare, nella Sezione 4.1 verranno ripresi i problemi MDP polinomiali con proprietà di Single Controller estendendoli ad uno spazio degli stati continuo, quindi verrà mostrato il relativo metodo di risoluzione.

Nella Sezione 4.2 invece, verranno ripresi i problemi MDP polinomiali con proprietà di Switching Controller, estendendo anche questi ad uno spazio degli stati continuo, quindi verrà mostrato un algoritmo iterativo che ne restituisce una soluzione approssimata.

Infine, in Sezione 4.3 verrà presentata la valutazione teorica dell'errore di approssimazione della funzione d'utilità che si ha quando si risolve un gioco con spazio continuo degli stati approssimandolo con uno spazio degli stati campionato ed assumendo che le funzioni di reward, probabilità di transizione e d'utilità del gioco siano Lipschitz-continue.

4.1 MDP Polinomiali SC con spazio degli stati continuo

Nella Sezione 3.2 è stata presentata una particolare classe di Markov Decision Processes (MDP) a 2 giocatori, a somma zero ed aventi proprietà di Single Controller, e ne è stata proposta prima una soluzione tramite formulazione in programmazione polinomiale ottimizzata non direttamente risolvibile, quindi una soluzione tramite formulazione in programmazione SDP equivalente alla precedente ma direttamente computabile. In questa sezione riprenderemo quindi la trattazione di questi giochi per analizzare il caso in cui lo spazio delle stati del gioco passi da un numero finito ad un numero infinito e non numerabile di elementi, in un intervallo continuo $[c, d]$. Per fare ciò, presenteremo prima la nozione di funzione Lipschitz-continua, quindi useremo questo nuovo concetto all'interno di un problema MDP polinomiale Single Controller per permettere la trattazione di spazi continui degli stati di gioco, sfruttando una tipologia di approccio non parametrico molto simile a quello presentato già in Sezione 2.6.4. I risultati di questa sezione rappresentano uno dei contributi più originali di questa tesi.

4.1.1 Introduzione alle funzioni Lipschitz-continue

Prima di iniziare la trattazione riguardante problemi MDP polinomiali Single Controller con spazio degli stati continuo, occorre introdurre una nuova nozione che risulterà assolutamente fondamentale per la sua formulazione. Questa nozione riguarda le cosiddette funzioni Lipschitz-continue: queste sono definite tramite una particolare assunzione di regolarità sul modello della funzione stessa. Nel nostro caso, utilizzeremo queste assunzioni per ridefinire le varie funzioni caratterizzanti i problemi MDP, cioè la funzione di reward, la funzione di probabilità di transizione da uno stato all'altro e la funzione d'utilità del gioco.

Si consideri un MDP con spazio degli stati S continuo nell'intervallo $[c, d]$. Si definisca quindi con $d(s, s')$ la funzione che restituisce la distanza tra lo stato s allo stato s', cioè:

$$d(s, s') \;=\; \|k(s) - k(s')\|$$

dove $k(s)$ rappresenta una mappatura dallo spazio degli stati ad uno spazio di vettori normalizzato.

Definizione 30. [Funzione di Reward Lipschitz-continua] Si definisca con $R : S \times A \longmapsto \mathbb{R}$ il modello della funzione di Reward, tale che $r(s, a)$ rappresenta il

Reward ottenuto nello stato s facendo l'azione a. La funzione di Reward così definita è Lipschitz-continua di costante L_R se soddisfa il seguente vincolo per ogni coppia di stati s_1 e $s_2 \in S$:

$$|r(s_1, a) - r(s_2, a)| \leq L_R d(s_1, s_2)$$

Definizione 31. [Modello di Transizione Lipschitz-continuo] Si definisca con $P : S \times A \times S \longmapsto [0; 1]$ la funzione di probabilità di transizione, tale che $p(s', s, a)$ rappresenta la probabilità condizionata di spostarsi dallo stato s allo stato s' facendo l'azione a. Il modello di transizione basato su una funzione di transizione così definita è Lipschitz-continuo di costante L_P se, per ogni coppia di stati s_1 e $s_2 \in S$ e per ogni valore normalizzato V della funzione d'utilità del gioco, soddisfa il seguente vincolo:

$$|\int_{s' \in S} \Big(p(s', s_1, a) - p(s', s_2, a)\Big) V(s') ds'| \leq L_P d(s_1, s_2)$$

Si noti che, per quest'ultima definizione, si presuppone che i valori della funzione d'utilità siano normalizzati, tuttavia nel caso questi non lo siano, è sempre possibile scalarli appropriatamente per riportarsi alle condizioni richieste nella definizione appena data.

Definizione 32. [Funzione d'utilità Lipschitz-continuo] Si definisca con $V : S \longmapsto \mathbb{R}$ la funzione d'utilità del gioco, tale che $V(s)$ rappresenta il valore della policy ottima nello stato s rispettando il vincolo dato dall'Equazione di Bellman definito nell'espressione (2.4). La funzione d'utilità così definita è Lipschitz-continua di costante L_V se soddisfa il seguente vincolo per ogni coppia di stati s_1 e $s_2 \in S$:

$$|V(s) - V(s')| \leq L_V d(s, s')$$

Nel corso della nostra trattazione si userà la notazione $\mathcal{F}_L$ per rappresentare l'insieme delle funzioni Lipschitz-continue di costante L.

Si noti che, se per le funzioni di reward e di probabilità di transizione è possibile calcolare la vera costante di Lipschitz, cioè rispettivamente L_R e L_P, questo non è vero per la funzione d'utilità, se non per particolari casi ben conosciuti. Si vuole quindi determinare un modo per limitare in un intervallo specifico la scelta della costante di Lipschitz L_V che andremo a definire per

risolvere i nostri problemi. Si definisca con L_{V^*} la vera costante di Lipschitz relativa alla funzione d'utilità; il seguente teorema ci permette di limitare superiormente il valore della nostra L_V

Teorema 24. *Se $\beta L_P \leq 1$, allora si può definire la seguente disuguaglianza:*

$$L_{V^*} \leq \frac{L_R}{1 - \beta L_P}.$$

Dimostrazione. La dimostrazione è mostrata in [33]. □

La scelta di usare questa particolare tipologia di funzioni deriva dal fatto che permette di considerare una campionatura di stati invece dell'intero insieme (infinito) di stati per la risoluzione degli MDP con spazio degli stati continuo. La campionatura fatta sotto l'assunzione di funzioni Lipschitz-continue ha infatti dei grandi vantaggi intrinsechi:

- permette di passare da uno spazio continuo degli stati ad un suo particolare sottoinsieme con un numero di elementi finito, quindi il gioco approssimato risultante diventa risolvibile mediante le modalità già viste nello scorso capitolo, risoluzione che garantisce anche un determinato bound sull'errore di approssimazione dovuto alla campionatura stessa;

- permette di scegliere un numero finito a piacere di stati campionati per la risoluzione del gioco approssimato, in quanto il valore di un qualsiasi stato t non scelto nella campionatura è ricavabile anche a posteriori della risoluzione identificando lo stato campionato s che massimizza questo semplice vincolo:

$$V(t) \geq V(s) - L_V d(s, t).$$

4.1.2 Presentazione del problema CMDP polinomiale Single Controller

Per iniziare questa trattazione, partiamo dalla descrizione del problema, recuperando anche alcune nozioni già citate nelle sezioni precedenti.
Si consideri un MDP come quello definito nella Definizione 21 in cui valga la condizione di Single Controller, definita in Definizione 20, e in cui siano presenti nel gioco $N = 2$ giocatori, e si consideri il fatto che il gioco a somma zero.
Si modifichi la definizione relativa allo spazio degli stati: in questo caso consideriamo uno spazio degli stati S continuo, quindi con un numero infinito e

non numerabile di elementi, e definito nell'intervallo chiuso $[c, d]$. All'interno di S si definiscano 2 suoi sottoinsiemi, S^* ed $\tilde{S}$, i quali sono ricavati attraverso 2 campionature diverse degli stati nello spazio continuo S; il primo, cioè S^*, è il sottoinsieme degli stati di cui si calcoleranno le strategie ottime dei giocatori e che rappresenta l'insieme campionato degli stati di partenza del modello di transizione, mentre il secondo, cioè $\tilde{S}$, è il sottoinsieme degli stati che rappresenta solo l'insieme campionato degli stati di arrivo del modello di transizione, quindi di questo non verranno calcolate le strategie ottime dei giocatori.
Si consideri lo spazio delle azioni A di gioco come un insieme infinito di elementi in un determinato intervallo finito, cioè $A = [c, d] \in \mathbb{R}$; per semplicità di trattazione si assuma che $A = A_1 = A_2$, e che lo spazio delle azioni sia uguale per ogni stato $s \in S^*$ del gioco. Si consideri poi la funzione di payoff R a coefficienti reali di tipo polinomiale nelle variabili a_1 ed a_2, come quella definita nell'espressione (3.1), e si assuma che questa sia Lipschitz-continua di costante L_R, come quella definita nella Definizione 30. Infine si consideri la probabilità di transizione $p(s', s, a_1)$ a coefficienti reali di tipo polinomiale nella sola variabile a_1, quindi indipendente dall'azione a_2 del secondo giocatore in quanto vale la proprietà già citata di Single Controller, definita come segue:

$$p(s', s, a_1) = \sum_{i=0}^{d_{ss'}} p_i(s, s') a_1^i$$

e si assuma che questa rappresenti un modello di transizione Lipschitz-continuo di costante L_P, come quella definito nella Definizione 31.

Anche per questa particolare tipologia di giochi si può osservare che, essendo il processo potenzialmente infinito in termini di orizzonte decisionale, risulta più che naturale concentrarsi sulla ricerca delle strategie stazionarie dei giocatori, strategie definite nella Definizione 19 e che prevedono la dipendenza delle stesse solo dallo stato del gioco e non dal tempo.

A tal proposito, considereremo in particolare l'insieme delle strategie miste dei giocatori, in modo da poterci ricondurre alla nozione di equilibrio minmax. Si definisca quindi una strategia mista per il primo giocatore come l'insieme finito delle misure di probabilità $\mu = [\mu(1), \ldots, \mu(S^*)]$ il cui supporto è in A_1. In modo analogo si definisca una strategia mista per il secondo giocatore come $\nu = [\nu(1), \ldots, \nu(S^*)]$ il cui supporto è in A_2. Si noti come le strategie siano definite non su tutto lo spazio degli stati ma solo nel sottoinsieme campionato S^*.

Definite quindi le strategie miste dei 2 giocatori, μ e ν, si possono riformulare le nozioni di funzione di Reward R e di probabilità di transizione P. La prima, nel caso si consideri il primo giocatore nello stato di gioco $s \in S^*$,

sarà una funzione del tipo:

$$r(s, \mu(s), \nu(s)) \; = \; \int_{A_1} \int_{A_2} r(s, a_1, a_2) d\mu(s) d\nu(s).$$

$$r(s, \mu(s), \nu(s)) \; \in \; \mathcal{F}_{L_R}$$

La seconda invece, essendo dipendente solo dall'azione del primo giocatore, sarà una matrice di probabilità di transizione $P(\mu)$, indicizzata dagli stati $s \in S^*$ e $s' \in \tilde{S}$, definita come:

$$P_{ss'}(\mu) \; = \; \int_{A_S} p(s', s, a_1) d\mu(s).$$

$$P_{ss'}(\mu) \; \in \; \mathcal{F}_{L_P}$$

Date queste nozioni, si può scrivere il reward $v_\beta(s, \mu(s), \nu(s))$ approssimato accumulabile, convenzionalmente dal primo giocatore, partendo da un dato stato $s \in S^*$ come segue:

$$v_\beta(s, \mu(s), \nu(s)) \; = \; r(s, \mu(s), \nu(s)) + \beta \sum_{s' \in \tilde{S}} \Big(\int_{A_1} p(s', s, a_1) d\mu(s) \Big) v_\beta(s')$$

$$v_\beta(s, \mu(s), \nu(s)), v_\beta(s') \; \in \; \mathcal{F}_{L_V}$$

Grazie a questa formulazione, possiamo ora definire il concetto di strategia d'equilibrio (stazionario) approssimato per la sottoclasse di giochi che stiamo trattando, a cui si collega anche la definizione di vettore dei valori approssimati del gioco:

Definizione 33 (Strategie di Equilibrio Approssimato e Vettore dei Valori Approssimati)**.** Due vettori di strategie miste (indicizzati dallo stato s $\in S^*$) μ^0 e ν^0 che soddisfano la saddle-point condition definita dall'equazione (2.7) per ogni vettore di strategie miste μ e ν sono chiamati Strategie di Equilibrio Approssimato. Il corrispondente vettore $v_\beta(\mu^0, \nu^0)$ è chiamato Vettore dei Valori Approssimato del gioco.

E' stato introdotto il concetto di approssimazione per queste due nozioni principalmente per due motivi:

- il problema che andremo a risolvere si basa su una campionatura dello spazio continuo degli stati, quindi, diversamente dal caso degli MDP polinomiali SC, non restituisce dei valori esatti, ma solo dei valori approssimati rispetto alla campionatura fatta;

- diversamente da quanto affermato per gli MDP polinomiali con spazio degli stati finito, per questa classe di problemi non è stato ancora dimostrato il fatto che il vettore dei valori reale associato alla strategia stazionaria d'equilibrio reale esistea sempre e sia unico.

Presentate tutte le nozioni necessarie per descrivere il nostro problema, possiamo ora passare alla presentazione formale dello stesso: la formulazione riprenderà quella già vista nella coppia di problemi primale PP1 (3.24) e duale PD1(3.30), quindi aggiungierà delle assunzioni più restrittive sia sulla funzione di reward nei diversi stati $s \in S^*$, che, oltre ad essere di tipo polinomiale come quella definita dall'equazione (3.1), dovrà essere anche Lipschitza-continua, cioè definita come in Definizione 30, sia sulla funzione di probabilità di transizione da stati $s \in S^*$ a stati $s' \in \tilde{S}$, che, oltre ad essere di tipo polinomiale, dovrà essere anche Lipschitza-continua, cioè definita come in Definizione 31, infine imporra un nuovo vincolo sulla funzione d'utilità, di cui si andranno a calcolare i valori approssimati, che dovrà essere anche lei Lipschitz-continua, cioè dorà rispettare la Definizione 32.
Il nuovo problema primale, chiamato CPP1, che può essere appunto visto come una versione vincolata di PP1(3.24), è formalizzato in funzione dei momenti come segue:

$$\min_{v(s),v(\tilde{s})} \sum_{s \in S^*} v(s) + \sum_{\tilde{s} \in \tilde{S}} v(\tilde{s}),$$

subject to

$$v(s) - \sum_{i=0}^{n_s} \sum_{j=0}^{m_s} r_{ij}(s) \nu_j(s) {a_1}^i - \beta \sum_{\tilde{s} \in \tilde{S}} P(\tilde{s}, s, a_1) v(\tilde{s}) \in \mathcal{P}(A_1), \quad \forall s \in S^* \tag{4.1a}$$

$$\overline{\nu}(s) \in \mathcal{M}(A_2), \quad \forall s \in S^* \tag{4.1b}$$

$$\nu_0(s) = 1, \quad \forall s \in S^* \tag{4.1c}$$

$$v(s), v(\tilde{s}) \in \mathcal{F}_{L_V}, \quad \forall s \in S^*, \forall \tilde{s} \in \tilde{S} \tag{4.1d}$$

Leggendo questa formulazione, si nota subito che il problema risulta formulato in modo non direttamente risolvibile. Per renderlo quindi computazionalmente risolvibile, occorrerà dare nuova formulazione agli insiemi $\mathcal{P}(A_1)$ e $\mathcal{M}(A_2)$, in modo molto simile a quanto fatto per gli MDP polinomiali Single Controller, ed all'insieme $\mathcal{F}_{L_V}$. Prima però di passare alla riformulazione del problema, presentiamo anche il rispettivo duale.

Si definiscano le variabili ausiliarie $\lambda^+(s, \tilde{s})$ e $\lambda^-(s, \tilde{s})$, con $s \in S^*$ e con $\tilde{s} \in \tilde{S}$, che rappresentano l'equivalente duale del vincolo primale (4.1d). Si

definisca poi la variabile $\delta(s, s')$, con $s, s' \in S^*$, che vale 1 se $s = s'$, e vale 0 in tutti gli altri i casi. Infine si definisca con $\alpha = [\alpha(1), \ldots, \alpha(S^*)]$ il vettore dei valori del gioco relativamente al problema duale. Il nuovo problema duale, chiamato CPD1, che può essere visto come una versione vincolata di PD1(3.30), è formalizzato quindi come segue:

$$\max_{\alpha(s),\lambda^+(s,\tilde{s}),\lambda^-(s,\tilde{s})} \sum_{s\in S^*} \alpha(s) - \sum_{s\in S^*}\sum_{\tilde{s}\in\tilde{S}} L_V d(s,\tilde{s})(\lambda^+(s,\tilde{s}) + \lambda^-(s,\tilde{s})),$$

subject to

$$-\alpha(s) + \sum_{i=0}^{n_s}\sum_{j=0}^{m_s} r_{ij}(s)\epsilon_i(s){a_2}^j \in \mathcal{P}(A_2), \quad \forall s \in S^* \tag{4.2a}$$

$$\sum_{s\in S^*}\int_{a_1\in A_1}\Big(\delta(s,s')\Big)d\epsilon(s') + \sum_{\tilde{s}\in\tilde{S}}\Big(\lambda^+(s',\tilde{s}) - \lambda^-(s',\tilde{s})\Big) = 1, \quad \forall s' \in S^* \tag{4.2b}$$

$$\sum_{s\in S^*}\int_{a_1\in A_1}\Big(-\beta p(s',s,a_1)\Big)d\epsilon(s) - \sum_{s^*\in S^*}\Big(\lambda^+(s^*,s') - \lambda^-(s^*,s'))\Big) = 1, \quad \forall s' \in \tilde{S} \tag{4.2c}$$

$$\overline{\epsilon}(s) \in \mathcal{M}(A_1), \quad \forall s \in S^* \tag{4.2d}$$

$$\lambda^+(s,\tilde{s}),\ \lambda^-(s,\tilde{s}) \geq 0, \quad \forall\Big(s \in S^*, \tilde{s} \in \tilde{S}\Big) \tag{4.2e}$$

4.1.3 Risoluzione approssimata del problema tramite programmazione semidefinita positiva

Iniziamo ora a riformulare il problema primale per renderlo computazionalmente risolvibile. Partiamo dalla riformulazione dell'insieme $\mathcal{P}(A_1)$, presente nel vincolo (4.1a): quest'ultimo rappresenta, per ogni stato $s \in S^*$, una disuguaglianza polinomiale nella variabile a_1. Fissando quindi lo stato $s \in S^*$, si può considerare il polinomio univariato:

$$t_s(a_1) = v(s) - \sum_{i=0}^{n_s}\sum_{j=0}^{m_s} r_{ij}(s)\nu_j(s){a_1}^i - \beta\sum_{s'\in\tilde{S}} P(s',s,a_1)v(s').$$

Di questo polinomio si vorranno esplicitare i coefficienti, in modo tale da poter riformulare il problema in termini di formulazione semidefinita positiva. Si definisca con d_s il grado del polinomio per lo stato $s \in S^*$, e con $[a_1]_{d_s} = [1, a_1, a_1^2, \ldots, a_1^{d_s}]^T$ il vettore delle azioni nello stato s, indicizzato

dal grado. Si definisca poi con $R(s)$ la matrice dei coefficienti del polinomio $r(s, a_1, a_2)$, e con $\overline{\nu}(s) \in \mathbb{R}^{d_s+1}$ il vettore dei primi d_s+1 momenti della misura di probabilità $\nu(s)$. Si definisca infine con $Q(s)$ la matrice dei coefficienti del polinomio $p(s', s, a_1)$ per ogni s' $\in \tilde{S}$. Grazie alle nozioni appena introdotte, riformularemo le varie parti del vincolo(4.1a) in forma vettoriale.

Partendo dal primo termine del polinomio, cioè v(s), si consideri $\overline{v} = [v(1), \ldots, v(S^*)]^T$, cioè il vettore dei valori del gioco ricavato dal problema primale. Si consideri anche $\tilde{v} = [v(1), \ldots, v(\tilde{S})]^T$, cioè il vettore dei valori degli stati su cui non si calcoleranno le strategie miste dei giocatori, ricavato sempre dal problema primale. Per quanto riguarda invece la funzione di reward, possiamo riscriverla in forma vettoriale come segue:

$$\sum_{i=0}^{n_s} \sum_{j=0}^{m_s} r_{ij}(s) \nu_j(s) {a_1}^i \; = \; \overline{\nu}(s)^T R(s)^T [a_1]_{d_s}. \tag{4.3}$$

Infine, per quanto riguarda la probabilità di transizione dagli stati s $\in S^*$ agli stati s' $\in \tilde{S}$, possiamo riscriverla nella seguente forma:

$$\sum_{s' \in \tilde{S}} P(s', s, a_1) v(s') \; = \; \tilde{v}^T Q(s)^T [a_1]_{d_s}. \tag{4.4}$$

Data questa nuova formulazione vettoriale delle parti che compongono il vincolo (4.1a), e riprendendo quanto detto nel Lemma 4 presentato in Sezione 3.1.1, si può esprimere la non negatività del polinomio univariato $t_s(a_1)$ sull'intervallo $[c, d]$ tramite il vincolo

$$\mathcal{H}^*(Z(s)+(c+d)\frac{1}{2}(L_1 W(s) L_2^T + L_2 W(s) L_1^T) - (c*d) L_1 W(s) L_1^T - L_2 W(s) L_2^T) \; =$$

$$= \; E(s)\overline{v} - R(s)\overline{\nu}(s) - \beta Q(s)\tilde{v}. \tag{4.5}$$

con le matrici $Z(s) \in (\mathcal{S}^*)^{d_s+1}$ e $W(s) \in (\mathcal{S}^*)^{d_s}$ semidefinite positive, e con $E(s) \in \mathbb{R}^{d_s \times S^*}$ che rappresenta una matrice formata da tutti zeri eccetto un 1 in posizione $(1, s)$.

Passiamo ora alla riformulazione dell'insieme $\mathcal{M}(A_2)$ presente nei vincoli (4.1b) e (4.1c) e riguardante la validità dei momenti relativi alle misure di probabilità. Si definisca con $e_1 \in \mathbb{R}^{m+1}$ un vettore di lunghezza $m+1$ composto dal primo elemento che vale 1, e da tutti gli altri che valgono 0. Si consideri il Lemma 6 presentato nella Sezione 3.1.2, quindi si fissi uno stato $s \in S^*$: grazie al lemma appena indicato, possiamo esprimere i 2 vincoli (4.1b) e (4.1c) nell'intervallo delle azioni $[c, d]$ tramite formulazione in programmazione semidefinita positiva come segue:

$$e_1^T [\nu_0(s), \ldots, \nu_m(s)] \; = \; 1. \tag{4.6a}$$

$$-c * \mathcal{H}([\nu_0(s), \ldots, \nu_m(s)]^T) + \mathcal{H}([\nu_1(s), \ldots, \nu_{m+1}(s)]^T) \succeq 0 \tag{4.6b}$$

$$d * \mathcal{H}([\nu_0(s), \ldots, \nu_m(s)]^T) - \mathcal{H}([\nu_1(s), \ldots, \nu_{m+1}(s)]^T) \succeq 0 \tag{4.6c}$$

Per quanto riguarda infine la riformulazione dell'insieme $\mathcal{F}(L_V)$ presente nel vincolo (4.1d) riguardante l'assunzione di Lipschitz-continuità sulla funzione d'utilità, si consideri la Definizione 32 presentata nella precedente sezione. Grazie all'espressione presente nella definizione appena indicata, possiamo esprimere il vincolo (4.1d) tramite formulazione in programmazione lineare, e valida in questo caso anche per la programmazione SDP, come segue:

$$v(s) - v(\tilde{s}) \geq -L_V * d(s, \tilde{s}), \quad \forall\Big(s \in S^*, \tilde{s} \in \tilde{S}\Big) \tag{4.7a}$$

$$-v(s) + v(\tilde{s}) \geq -L_V * d(s, \tilde{s}), \quad \forall\Big(s \in S^*, \tilde{s} \in \tilde{S}\Big) \tag{4.7b}$$

Ora, grazie alle 3 formulazioni (4.5),(4.6) e (4.7) appena definite, possiamo riformulare il problema di ottimizzazione polinomiale astratto CPP1 (4.1) in modo più concreto, computazionalmente risolvibile, tramite la seguente formulazione in programmazione semidefinita positiva (CSP1):

$$\min_{v(s),v(\tilde{s})} \sum_{s\in S^*} v(s) + \sum_{\tilde{s}\in\tilde{S}} v(\tilde{s}),$$

subject to

$$\mathcal{H}^*(Z(s)+(c+d)\frac{1}{2}(L_1W(s)L_2^T+L_2W(s)L_1^T)-(c*d)L_1W(s)L_1^T-L_2W(s)L_2^T) = \\ = E(s)\overline{v} - R(s)\overline{\nu}(s) - \beta Q(s)\tilde{v}, \quad \forall s \in S^* \tag{4.8a}$$

$$e_1^T[\nu_0(s),\ldots,\nu_m(s)] = 1, \quad \forall s \in S^* \tag{4.8b}$$

$$-c*\mathcal{H}([\nu_0(s),\ldots,\nu_m(s)]^T)+\mathcal{H}([\nu_1(s),\ldots,\nu_{m+1}(s)]^T) \succeq 0, \quad \forall s \in S^* \tag{4.8c}$$

$$d*\mathcal{H}([\nu_0(s),\ldots,\nu_m(s)]^T)-\mathcal{H}([\nu_1(s),\ldots,\nu_{m+1}(s)]^T) \succeq 0, \quad \forall s \in S^* \tag{4.8d}$$

$$v(s) - v(\tilde{s}) \geq -L_V * d(s,\tilde{s}), \quad \forall\Big(s \in S^*, \tilde{s} \in \tilde{S}\Big) \tag{4.8e}$$

$$-v(s) + v(\tilde{s}) \geq -L_V * d(s,\tilde{s}), \quad \forall\Big(s \in S^*, \tilde{s} \in \tilde{S}\Big) \tag{4.8f}$$

$$Z(s), W(s) \succeq 0, \quad \forall s \in S^* \tag{4.8g}$$

Possiamo quindi definire il seguente Lemma:

Lemma 20. Il problema di programmazione semidefinita positiva CSP1, definito in (4.8), risolve esattamente il problema di ottimizzazione polinomiale CPP1, definito in (4.1).

Dimostrazione. La disuguaglianza polinomiale definita in (4.1) ha come vettore dei coefficienti $E(s)\overline{v} - R(s)\overline{\nu}(s) - \beta Q(s)\tilde{v}$, come mostrato nella presentazione delle equazioni (4.3) e (4.4). La dimostrazione si ha quindi per diretta

conseguenza del Lemma 4, riguardante la rappresentazione della non negatività dei polinomi univariati nell'intervallo $[c, d]$ tramite programmazione semidefinita positiva, applicato come mostrato nell'equazione (4.5), del Lemma 6, riguardante la rappresentazione delle sequenze valide di momenti di una misura non negativa supportata nell'intervallo $[c, d]$ tramite programmazione semidefinita positiva, applicato come mostrato nel sistema di equazioni (4.6), e della Definizione 32, riguardante la rappresentazione di funzioni d'utilità Lipschitz-continue di costante L_V. □

La soluzione di questo problema produrrà il vettore dei valori approssimati del gioco ed i momenti delle misure, una per ogni stato $s \in S^*$, per il secondo giocatore, che è quello che non controlla le probabilità di transizione tra uno stato e l'altro: per ottenere dai momenti i supporti ed i pesi delle strategie miste ottime approssimate del giocatore si utilizzerà poi la procedura mostrata nella Sezione 3.1.4.

Per ricavare invece le strategie ottime del primo giocatore, dovremo ricavare i momenti delle misure del problema duale CPD1 definito in (4.2). Si definisca con $\overline{q}(s, \tilde{s})$ il vettore dei coefficienti del polinomio $p(s', s, a_1)$. Si definisca poi con $\overline{\delta}(s, s')$ un vettore formato da tutti 0, a parte il primo elemento che assume il valore 1 se $s = s'$, con $s, s' \in S^*$, 0 altrimenti.

Applicando le stesse tecniche mostrate in precedenza, mostriamo ora il problema di programmazione semidefinita duale SD1, equivalente a quello di ottimizzazione polinomiale PD1 mostrato in (4.2) e duale del problema di programmazione semidefinita CSP1 mostrato in (4.8):

$$\max_{\alpha(s),\lambda^+(s,\tilde{s}),\lambda^-(s,\tilde{s})} \sum_{s\in S^*} \alpha(s) - \sum_{s\in S^*}\sum_{\tilde{s}\in\tilde{S}} L_V d(s,\tilde{s})(\lambda^+(s,\tilde{s})+\lambda^-(s,\tilde{s})),$$

subject to

$$\mathcal{H}^*(J(s)+(c+d)\frac{1}{2}(L_1K(s)L_2^T+L_2K(s)L_1^T)-(c*d)L_1K(s)L_1^T-L_2K(s)L_2^T) = \\ = -\alpha(s)e_1 + R(s)^T\overline{\epsilon}(s), \quad \forall s\in S^* \tag{4.9a}$$

$$-c*\mathcal{H}([\epsilon_0(s),\ldots,\epsilon_n(s)]^T)+\mathcal{H}([\epsilon_1(s),\ldots,\epsilon_{n+1}(s)]^T) \succeq 0, \quad \forall s\in S \tag{4.9b}$$

$$d*\mathcal{H}([\epsilon_0(s),\ldots,\epsilon_n(s)]^T)-\mathcal{H}([\epsilon_1(s),\ldots,\epsilon_{n+1}(s)]^T) \succeq 0, \quad \forall s\in S \tag{4.9c}$$

$$\sum_{s\in S^*}\Big(\overline{\delta}(s,s')\Big)\overline{\epsilon}(s)^T + \sum_{\tilde{s}\in\tilde{S}}\Big(\lambda^+(s',\tilde{s})-\lambda^-(s',\tilde{s})\Big) = 1, \quad \forall s'\in S^* \tag{4.9d}$$

$$\sum_{s\in S^*}\Big(-\beta\overline{q}(s,s')\Big)\overline{\epsilon}(s)^T - \sum_{s^*\in S^*}\Big(\lambda^+(s^*,s')-\lambda^-(s^*,s'))\Big) = 1, \quad \forall s'\in\tilde{S} \tag{4.9e}$$

$$\lambda^+(s,\tilde{s}),\ \lambda^-(s,\tilde{s})\geq 0, \quad \forall\Big(s\in S^*,\tilde{s}\in\tilde{S}\Big) \tag{4.9f}$$

$$J(s),K(s) \succeq 0, \quad \forall s\in S^* \tag{4.9g}$$

Come fatto per il problema SDP primale, esprimiamo ora il seguente Lemma riguardante il problema duale:

Lemma 21. Il problema di programmazione semidefinita positiva CSD1, definito in (4.9), risolve esattamente il problema di ottimizzazione polinomiale CPD1, definito in (4.2).

Dimostrazione. La dimostrazione è del tutto analoga a quella mostrata nel Lemma 20, cioè procede tramite applicazione diretta del Lemma 4 e del Lemma 6 sul problema di ottimizzazione polinomiale CPD1 (4.2). □

E' importante ricordare che la sequenza dei momenti $\overline{\epsilon}$ ricavata non corrisponde sempre ad una misura di probabilità, quindi non è propriamente la sequenza dei momenti delle misure per il primo giocatore, quindi andrà normalizzata come mostrato nell'espressione (3.32) presentata nella Sezione 3.2.2. Una volta normalizzata, si otterrà la sequenza dei momenti $\overline{\mu}$ delle misure, una per ogni stato $s \in S^*$, per il primo giocatore, che è quello che controlla le probabilità di transizione tra uno stato e l'altro: per ottenere dai momenti i supporti ed i pesi delle strategie miste ottime approssimate del giocatore si utilizzerà poi la procedura mostrata nella Sezione 3.1.4.

Come già detto, purtroppo non è possibile garantire che la soluzione trovata tramite la risoluzione della coppia di problemi primale-duale, cioè la strategia di equilibrio approssimato ed il relativo vettore dei valori approssimati, sia realmente quella ottimale per il problema CMDP reale, quindi sarà fondamentale mostrare qual'è il bound teorico sull'errore massimo di approssimazione tra la soluzione reale e quella approssimata, cosa che mostreremo successivamente dopo aver trattato anche il caso di risoluzione di un problema CMDP polinomiale Switching Controller.

4.2 MDP Polinomiali Switching Controller con spazio degli stati continuo

Nella precedente sezione è stata presentato il problema di risoluzione di un CMDP polinomiale con proprietà di Single Controller, in cui cioè le probabilità di transizione da uno stato all'altro del gioco dipendono solo dalle azioni di uno stesso unico giocatore, quindi ne è stata proposta una soluzione approssimata basata su formulazione in programmazione semidefinita positiva. In questa sezione estenderemo il problema suddetto riprendendo la trattazione della sottoclasse di giochi stocastici con proprietà di Switching controller, in cui cioè i 2 giocatori si alternano nel controllo delle probabilità di transizione in base al particolare sottoinsieme degli stati in cui si trovano, ed ampliandola ad uno spazio continuo degli stati. Di questo problema ne presenteremo la formulazione astratta in programmazione polinomiale ottimizzata, quindi produrremo una formulazione in programmazione semidefinita positiva computazionalmente calcolcabile (concreta) per quest'ultimo caso. I risultati di questa sezione rappresentano uno dei contributi più originali di questa tesi.

4.2.1 Presentazione del problema CMDP polinomiale Switching Controller

Iniziamo la presentazione di questo particolare problema, cioè il problema relativo alla risoluzione di un MDP Scon proprietà di Switching Controller, con spazio degli stati e spazio delle azioni aventi entrambi un numero di elementi infinito e non numerabile, e con funzione di reward e funzione di probabilità di transizione di tipo polinomiale.

Analizziamo per prima cosa i cambiamenti presenti nella tupla che definisce questo nuovo problema. Si consideri uno spazio degli stati S continuo, quindi con un numero infinito e non numerabile di elementi, e definito nell'intervallo chiuso $[c, d]$. All'interno di S poi si definiscano 2 suoi sottoinsiemi, S^* ed $\tilde{S}$, i quali sono ricavati attraverso 2 campionature diverse degli stati nello spazio continuo S; il primo, cioè S^*, è il sottoinsieme degli stati di cui si calcoleranno le strategie ottime dei giocatori e che rappresenta l'insieme campionato degli stati di partenza del modello di transizione, mentre il secondo, cioè $\tilde{S}$, è il sottoinsieme degli stati che rappresenta solo l'insieme campionato degli stati di arrivo del modello di transizione, quindi di questo non verranno calcolate le strategie ottime dei giocatori. Quindi, una volta definiti questi 2 sottoinsiemi di S, si partizionino entrambi in 2 sottoinsiemi $S_1 = \{1, ..., k\}$ ed $S_2 = \{k+1, ..., N\}$, tale che la funzione P relativa alla probabilità di transizione da

uno stato all'altro del gioco sia definita come nell'equazione (3.46); si avranno quindi 4 sottoinsieme campionati di S, definiti rispettivamente come S_1^*, S_2^*, $\tilde{S}_1$ ed $\tilde{S}_2$. Si noti che, per quanto riguarda la funzione di probabilità di transizione, si assumerà che questa rappresenti un modello di transizione Lipschitz-continuo di costante L_P, come quella definito nella Definizione 31. Si consideri poi uno spazio delle azioni $A = A_1 = A_2$ con un numero infinito e non numerabile di elementi nell'intervallo continuo $[c, d]$, spazio delle azioni che, per semplicità, si assume sia uguale per ogni stato $s \in S^*$. Si assuma che la funzione di reward sia di tipo polinomiale, come quella descritta nell'equazione (3.1), per ogni stato $s \in S^*$, , e che sia Lipschitz-continua di costante L_R, come quella definita nella Definizione 30. Si definisca infine il fattore di sconto β.

Definito il problema in analisi, si possono introdurre alcune importanti notazioni: per prima cosa si considerino le strategie miste f e g dei 2 giocatori, definite in questo caso tramite le corrispondenti misure di probabilità $\mu(s)$ e $\nu(s)$, per ogni $s \in S^*$, sull'insieme delle strategie miste, cioè sull'intervallo $[c, d]$ su cui variano a_1 ed a_2. Quest'ultime poi inducono a riformulare le funzioni di reward e di transizione in loro funzione, cioè si avrà una definizione del payoff atteso del primo giocatore come

$$r(s, f(s), g(s)) = \int_{a_1 \in A_1} \int_{a_2 \in A_2} r(s, a_1, a_2) d\mu(s) d\nu(s), \quad \forall s \in S^*$$
$$r(s, f(s), g(s)) \in \mathcal{F}_{L_R};$$

mentre, data una strategia μ, si definirà una matrice di probabilità di transizione $P(\mu)$ come segue

$$P_{ss'}(\mu) = \int_{A_1} p(s', s, a_1) d\mu(s), \quad \forall s \in S_1^*, s' \in \tilde{S}$$
$$P_{ss'}(\mu) \in \mathcal{F}_{L_P};$$

infine, data una strategia ν, si definirà una matrice di probabilità di transizione $P(\nu)$ come segue

$$P_{ss'}(\nu) = \int_{A_2} p(s', s, a_2) d\nu(s), \quad \forall s \in S_2^*, s' \in \tilde{S}$$
$$P_{ss'}(\nu) \in \mathcal{F}_{L_P}.$$

Il reward approssimato accumulabile, convenzionalmente dal primo giocatore, partendo da un dato stato s, dati gli elementi appena presentati, sarà definito dal seguente sistema di equazioni:

$$v_\beta(s, \mu(s), \nu(s)) = r(s, \mu(s), \nu(s)) + \beta \sum_{s' \in \tilde{S}_1} \Big(\int_{a_1 \in A_1} p(s', s, a_1) \mu(s, a_1) \Big) v_\beta(s') +$$

$$+\beta \sum_{s' \in \tilde{S}_2} \Big(\int_{a_1 \in A_1} p(s', s, a_1) \mu(s, a_1) \Big) w_\beta(s'), \quad \forall s \in S_1^*$$

$$w_\beta(s, \mu(s), \nu(s)) \; = \; r(s, \mu(s), \nu(s)) + \beta \sum_{s' \in \tilde{S}_1} \Big(\int_{a_2 \in A_2} p(s', s, a_2) \nu(s, a_2) \Big) v_\beta(s') +$$

$$+\beta \sum_{s' \in \tilde{S}_2} \Big(\int_{a_2 \in A_2} p(s', s, a_2) \nu(s, a_2) \Big) w_\beta(s'), \quad \forall s \in S_2^*$$

$$con \quad v_\beta(s, \mu(s), \nu(s)), w_\beta(s, \mu(s), \nu(s)), v_\beta(s'), w_\beta(s') \; \in \; \mathcal{F}_{L_V}$$

Per quanto riguarda la ricerca della soluzione del gioco, si ricorda che valgono le stesse considerazioni fatte nella Sezione 4.2.1 per i CMDP polinomiali Single Controller (valendo le stesse condizioni di gioco anche in questo caso), cioè è possibile definire il concetto di strategia d'equilibrio (stazionario) approssimato per il problema che stiamo trattando, a cui si collega anche la definizione di vettore dei valori approssimati del gioco, tuttavia non è garantita l'esistenza e l'unicità sia di una strategia stazionaria d'equilibrio esatta sia del vettore dei valori esatti del gioco.

Mostrate tutte le nozioni necessarie alla descrizione completa del problema, possiamo passare alla presentazione dei 2 problemi primali che generalizzano ad uno spazio continuo degli stati ed ad uno spazio continuo delle azioni i problemi lineari LP3 (3.47) e LP4 (3.48), mostrati precedentemente in Sezione 3.4.1.

Si supponga di avere delle stime aribitrarie di $\widehat{v}(s')$, con s' $\in \tilde{S}_1$, cioè $(\widehat{v}(1), \widehat{v}(2), \ldots, \widehat{v}(k))$, e di $\widehat{w}(s')$, con s' $\in \tilde{S}_2$, cioè $(\widehat{w}(k{+}1), \widehat{w}(k{+}2), \ldots, \widehat{w}(N))$.

Il primo problema primale di ottimizzazione polinomiale, chiamato CPM3, estende il problema LP3 (3.47) ed è definito come segue:

$$\max_{w(s),w(\tilde{s})} \sum_{s\in S_2^*} w(s) + \sum_{\tilde{s}\in\tilde{S}_2} w(\tilde{s}),$$

subject to

$$+\int_{A_1} r(s,a_1,a_2)\mu(s,a_1)da_1 + \beta \sum_{s'\in\tilde{S}_2} p(s',s,a_2)w(s') +$$

$$-w(s) \;\geq\; -\beta \sum_{s'\in\tilde{S}_1} p(s',s,a_2)\widehat{v}(s'), \quad \forall s\in S_2^*,\, \forall a_2\in A_2 \tag{4.10a}$$

$$\int_{A_1} \mu(s,a_1)da_1 \;=\; 1, \quad \forall s\in S_2^* \tag{4.10b}$$

$$\mu(s,a_1) \;\geq\; 0, \quad \forall s\in S_2^*,\, \forall a_1\in A_1 \tag{4.10c}$$

$$w(s) - w(s') \;\geq\; -L_V d(s,s'), \quad \forall\Big(s\in S_2^*, s'\in\tilde{S}_2\Big) \tag{4.10d}$$

$$-w(s') \;\geq\; -L_V d(s,s') - \widehat{v}(s), \quad \forall\Big(s\in S_1^*, s'\in\tilde{S}_2\Big) \tag{4.10e}$$

$$-w(s) + w(s') \;\geq\; -L_V d(s,s'), \quad \forall\Big(s\in S_2^*, s'\in\tilde{S}_2\Big) \tag{4.10f}$$

$$w(s') \;\geq\; -L_V d(s,s') + \widehat{v}(s), \quad \forall\Big(s\in S_1^*, s'\in\tilde{S}_2\Big) \tag{4.10g}$$

Si noti che gli ultimi 4 vincoli equivalgono all'assunzione di Lipschitz-continuità fatta sulla funzione d'utilità, cioè:

$$w(s), w(\tilde{s}) \;\in\; \mathcal{F}_{L_V}, \quad \forall s\in S_2^*,\, \forall\tilde{s}\in\tilde{S}_2$$

Il secondo problema primale di ottimizzazione polinomiale, chiamato CPM4, estende il problema LP4 (3.48) ed è definito come segue:

$$\min_{v(s),v(\tilde{s})} \sum_{s\in S_1^*} v(s) + \sum_{\tilde{s}\in\tilde{S}_1} v(\tilde{s}),$$

subject to

$$-\int_{A_2} r(s,a_1,a_2)\nu(s,a_2)a_2 - \beta \sum_{s'\in\tilde{S}_1} p(s',s,a_1)v(s') +$$

$$+ v(s) \;\geq\; \beta \sum_{s'\in\tilde{S}_2} p(s',s,a_1)\widehat{w}(s'), \quad \forall s \in S_1^*,\, \forall a_1 \in A_1 \tag{4.11a}$$

$$\int_{A_2} \nu(s,a_2)da_2 \;=\; 1, \quad \forall s \in S_1^* \tag{4.11b}$$

$$\nu(s,a_2) \;\geq\; 0, \quad \forall s \in S_1^*,\, \forall a_2 \in A_2 \tag{4.11c}$$

$$v(s) - v(s') \;\geq\; -L_V d(s,s'), \quad \forall\Big(s \in S_1^*, s' \in \tilde{S}_1\Big) \tag{4.11d}$$

$$-v(s') \;\geq\; -L_V d(s,s') - \widehat{w}(s), \quad \forall\Big(s \in S_2^*, s' \in \tilde{S}_1\Big) \tag{4.11e}$$

$$-v(s) + v(s') \;\geq\; -L_V d(s,s'), \quad \forall\Big(s \in S_1^*, s' \in \tilde{S}_1\Big) \tag{4.11f}$$

$$v(s') \;\geq\; -L_V d(s,s') + \widehat{w}(s), \quad \forall\Big(s \in S_2^*, s' \in \tilde{S}_1\Big) \tag{4.11g}$$

Si noti che, anche in questo caso, gli ultimi 4 vincoli equivalgono all'assunzione di Lipschitz-continuità fatta sulla funzione d'utilità, cioè:

$$v(s), v(\tilde{s}) \;\in\; \mathcal{F}_{L_V}, \quad \forall s \in S_1^*,\, \forall \tilde{s} \in \tilde{S}_1$$

Questa formulazione però, come nel caso Single Controller, è astratta e non direttamente risolvibile: per renderla tale dovremmo ridefinirla successivamente tramite formulazione in programmazione SDP. Per fare questo però,

occorre prima trasformare i vincoli (4.10a) e (4.11a) in sistemi di disequazioni polinomiali univariate nelle azioni rispettivamente a_2 ed a_1, assumendo i valori di $\mu(s)$ e $\nu(s)$ fissati. L'unica componente non espressa esplicitamente come polinomio univariato risulta essere la funzione di reward, che andrà quindi riscritta così

$$\int_{A_1} r(s, a_1, a_2)\mu(s, a_1)da_1 \; = \; \sum_{i=0}^{n_s}\sum_{j=0}^{m_s} r_{ij}(s)\mu_i(s){a_2}^j$$

nel caso del vincolo (4.10a) in CPM3, e così

$$\int_{A_2} r(s, a_1, a_2)\nu(s, a_2)da_2 \; = \; \sum_{i=0}^{n_s}\sum_{j=0}^{m_s} r_{ij}(s)\nu_j(s){a_1}^i$$

nel caso del vincolo (4.11a) in CPM4.

Ora possiamo riscrivere i 2 problemi primali relazionandoli agli insiemi $\mathcal{P}_n$, cioè l'insieme dei polinomi univariati di grado n non negativi nell'intervallo $[c, d]$, $\mathcal{M}_m$, cioè l'insieme dei primi $m+1$ momenti di una misura di probabilità non negativa con supporto sempre nell'intervallo $[c, d]$. Per semplicità di notazione, i problemi primali saranno relazionati anche all'insieme $\mathcal{F}_{L_V}$, cioè l'insieme delle funzioni d'utilità Lipschitz-continue di costante L_V definite nella Definizione 32 presentata nella Sezione 4.1.1. Il primo problema di ottimizzazione polinomiale, chiamato CPP3 ed equivalente a CPM3 (4.10), è così ridefinito:

$$\max_{w(s),w(\tilde{s})} \sum_{s\in S_2^*} w(s) + \sum_{\tilde{s}\in\tilde{S}_2} w(\tilde{s}),$$

subject to

$$+\sum_{i=0}^{n_s}\sum_{j=0}^{m_s} r_{ij}(s)\mu_i(s){a_2}^j + \beta \sum_{s'\in\tilde{S}_2} p(s', s, a_2)w(s') +$$

$$+\beta \sum_{s'\in\tilde{S}_1} p(s', s, a_2)\widehat{v}(s') - w(s) \; \in \; \mathcal{P}(A_2), \quad \forall s \in S_2^* \tag{4.12a}$$

$$\overline{\mu}(s) \; \in \; \mathcal{M}(A_1), \quad \forall s \in S_2^* \tag{4.12b}$$

$$\mu_0(s) = 1, \quad \forall s \in S_2^* \tag{4.12c}$$

$$w(s), w(\tilde{s}) \; \in \; \mathcal{F}_{L_V}, \quad \forall s \in S_2^*, \forall \tilde{s} \in \tilde{S}_2 \tag{4.12d}$$

Il secondo problema di ottimizzazione polinomiale, chiamato CPP4 ed equivalente a CPM4 (4.11), è invece così ridefinito:

$$\min_{v(s),v(\tilde{s})} \sum_{s\in S_1^*} v(s) + \sum_{\tilde{s}\in\tilde{S}_1} v(\tilde{s}),$$

subject to

$$-\sum_{i=0}^{n_s}\sum_{j=0}^{m_s} r_{ij}(s)\nu_j(s){a_1}^i - \beta\sum_{s'\in S_1} p(s',s,a_1)v(s') +$$

$$+ v(s) - \beta\sum_{s'\in S_2} p(s',s,a_1)\widehat{w}(s') \in \mathcal{P}(A_1), \quad \forall s\in S_1 \tag{4.13a}$$

$$\overline{\nu}(s) \in \mathcal{M}(A_2), \quad \forall s\in S_1 \tag{4.13b}$$

$$\nu_0(s) = 1, \quad \forall s\in S_1 \tag{4.13c}$$

$$v(s), v(\tilde{s}) \in \mathcal{F}_{L_V}, \quad \forall s\in S_1^*, \forall\tilde{s}\in\tilde{S}_1 \tag{4.13d}$$

Prima di concludere questa parte, presentiamo anche i rispettivi 2 problemi duali di quelli appena mostrati. Si supponga, come già fatto per i problemi primali, di avere delle stime aribitrarie di $\widehat{v}(s')$, con $s'\in\tilde{S}_1$, cioè $(\widehat{v}(1),\widehat{v}(2),\ldots,\widehat{v}(k))$, e di $\widehat{w}(s')$, con $s'\in\tilde{S}_2$, cioè $(\widehat{w}(k{+}1),\widehat{w}(k{+}2),\ldots,\widehat{w}(N))$.
Si definisca con $\varepsilon(s)$, per ogni stato $s\in S_2^*$ una strategia mista del giocatore 2, e con $\overline{\varepsilon}(s)\in\mathbb{R}^{d_s+1}$ il vettore relativo ai suoi primi $d_s{+}1$ momenti, con d_s che indica il grado del polinomio univariato; si definisca anche con $\epsilon(s)$, per ogni stato $s\in S_1^*$, una strategia mista del giocatore 1, e con $\overline{\epsilon}(s)\in\mathbb{R}^{d_s+1}$ il vettore relativo ai suoi primi $d_s{+}1$ momenti, con d_s che indica il grado del polinomio univariato.
Si definiscano le variabili ausiliarie $\lambda_3^+(s,\tilde{s})$ e $\lambda_3^-(s,\tilde{s})$, con $s\in S_2^*$ e con $\tilde{s}\in\tilde{S}_2$, e le variabili $\lambda_4^+(s,\tilde{s})$ e $\lambda_4^-(s,\tilde{s})$, con $s\in S_2^*$ e con $\tilde{s}\in\tilde{S}_1$,che rappresentano l'equivalente duale del vincolo primale (4.12d). In modo del tutto equivalente, definiscano anche le variabili ausiliarie $\lambda_1^+(s,\tilde{s})$ e $\lambda_1^-(s,\tilde{s})$, con $s\in S_1^*$ e con $\tilde{s}\in\tilde{S}_1$, e le variabili $\lambda_2^+(s,\tilde{s})$ e $\lambda_2^-(s,\tilde{s})$, con $s\in S_1^*$ e con $\tilde{s}\in\tilde{S}_2$, che rappresentano l'equivalente duale del vincolo primale (4.13d).
Si definisca poi la variabile $\delta(s,s')$, con $s,s'\in S^*$, che vale 1 se $s=s'$, e vale 0 in tutti gli altri i casi.
Infine si definisca con $\eta(s)$, per ogni stato $s\in S_2^*$ il vettore dei valori del gioco relativamente al primo problema duale, e con $\alpha(s)$, per ogni stato $s\in S_2^*$ il vettore dei valori del gioco relativamente al secondo problema duale.

Il primo problema duale, chiamato CPD3 e duale di CPP3 (4.12), sarà quindi definito così:

$$\max_{\eta(s),\lambda_3^+(s,\tilde{s}),\lambda_3^-(s,\tilde{s}),\lambda_4^+(s,\tilde{s}),\lambda_4^-(s,\tilde{s}),\varepsilon_j(s)} \sum_{s\in S_2^*}\Big[\eta(s) - \beta \sum_{s'\in\tilde{S}_1} \widehat{v}(s') \sum_{j=0}^{m_s} p_j(s',s)\varepsilon_j(s)\Big]$$

$$- \sum_{s\in S_2^*}\sum_{\tilde{s}\in\tilde{S}_2} L_V * d(s,\tilde{s})\Big(\lambda_3^+(s,\tilde{s}) + \lambda_3^-(s,\tilde{s})\Big)$$

$$- \sum_{s\in S_1^*}\sum_{\tilde{s}\in\tilde{S}_2}\Big[L_V * d(s,\tilde{s}) + \widehat{v}(\tilde{s})\Big](\lambda_4^+(s,\tilde{s})) - \sum_{s\in S_1^*}\sum_{\tilde{s}\in\tilde{S}_2}\Big[L_V * d(s,\tilde{s}) - \widehat{v}(\tilde{s})\Big](\lambda_4^-(s,\tilde{s}))$$

subject to

$$-\eta(s) - \sum_{i=0}^{n_s}\sum_{j=0}^{m_s} r_{ij}(s) {a_1}^i \varepsilon_j(s) \;\in\; \mathcal{P}(A_1), \quad \forall s \in S_2^* \tag{4.14a}$$

$$\overline{\varepsilon}(s) \;\in\; \mathcal{M}(A_2), \quad \forall s \in S_2^* \tag{4.14b}$$

$$\sum_{s\in S_2^*}\int_{a_2\in A_2}\Big(\delta(s,s')\Big)\varepsilon(s,a_2)da_2 - \sum_{\tilde{s}\in\tilde{S}_2}\Big[\lambda_3^+(s',\tilde{s}) - \lambda_3^-(s',\tilde{s})\Big] \;=\; 1, \quad \forall s' \in S_2^* \tag{4.14c}$$

$$\sum_{s\in S_2^*}\int_{a_2\in A_2}\Big(-\beta p(s',s,a_2)\Big)\varepsilon(s,a_2)da_2 + \sum_{s'\in S_2^*}\Big(\lambda_3^+(s',\tilde{s}) - \lambda_3^-(s',\tilde{s})\Big) + \\ + \sum_{s'\in S_1^*}(\lambda_4^+(s',\tilde{s}) - \lambda_4^-(s',\tilde{s})) \;=\; 1, \quad \forall \tilde{s} \in \tilde{S}_2 \tag{4.14d}$$

$$\lambda_3^+(s,\tilde{s}),\; \lambda_3^-(s,\tilde{s}) \;\geq\; 0, \quad \forall\Big(s \in S_2^*, \tilde{s} \in \tilde{S}_2\Big) \tag{4.14e}$$

$$\lambda_4^+(s,\tilde{s}),\; \lambda_4^-(s,\tilde{s}) \;\geq\; 0, \quad \forall\Big(s \in S_1^*, \tilde{s} \in \tilde{S}_2\Big) \tag{4.14f}$$

Il secondo problema, chiamato CPD4 e duale di CPP4 (4.13), sarà quindi così definito:

$$\max_{\alpha(s),\lambda_1^+(s,\tilde{s}),\lambda_1^-(s,\tilde{s}),\lambda_2^+(s,\tilde{s}),\lambda_2^-(s,\tilde{s}),\epsilon_i(s)} \sum_{s\in S_1^*}\Big[\alpha(s)+\beta\sum_{s'\in\tilde{S}_2}\widehat{w}(s')\sum_{i=0}^{n_s}p_i(s',s)\epsilon_i(s)\Big]$$

$$-\sum_{s\in S_1^*}\sum_{\tilde{s}\in\tilde{S}_1}L_V*d(s,\tilde{s})\Big(\lambda_1^+(s,\tilde{s})+\lambda_1^-(s,\tilde{s})\Big)$$

$$-\sum_{s\in S_2^*}\sum_{\tilde{s}\in\tilde{S}_1}\Big[L_V*d(s,\tilde{s})+\widehat{w}(\tilde{s})\Big](\lambda_2^+(s,\tilde{s}))-\sum_{s\in S_2^*}\sum_{\tilde{s}\in\tilde{S}_1}\Big[L_V*d(s,\tilde{s})-\widehat{w}(\tilde{s})\Big](\lambda_2^-(s,\tilde{s}))$$

subject to

$$-\alpha(s)+\sum_{i=0}^{n_s}\sum_{j=0}^{m_s}r_{ij}(s){a_2}^j\epsilon_i(s)\ \in\ \mathcal{P}(A_2),\quad \forall s\in S_1^* \tag{4.15a}$$

$$\overline{\epsilon}(s)\ \in\ \mathcal{M}(A_1),\quad \forall s\in S_1^* \tag{4.15b}$$

$$\sum_{s\in S_1^*}\int_{a_1\in A_1}\Big[\delta(s,s')\Big]\epsilon(s,a_1)da_1+\sum_{\tilde{s}\in\tilde{S}_1}\Big[\lambda_1^+(s',\tilde{s})-\lambda_1^-(s',\tilde{s})\Big]\ =\ 1,\quad \forall s'\in S_1^* \tag{4.15c}$$

$$\sum_{s\in S_1^*}\int_{a_1\in A_1}\Big[-\beta p(s',s,a_1)\Big]\epsilon(s,a_1)da_1-\sum_{s'\in S_1^*}\Big(\lambda_1^+(s',\tilde{s})-\lambda_1^-(s',\tilde{s})\Big)+$$

$$-\sum_{s'\in S_2^*}(\lambda_2^+(s',\tilde{s})-\lambda_2^-(s',\tilde{s}))\ =\ 1,\quad \forall\tilde{s}\in\tilde{S}_1 \tag{4.15d}$$

$$\lambda_1^+(s,\tilde{s}),\ \lambda_1^-(s,\tilde{s})\ \geq\ 0,\quad \forall\Big(s\in S_1^*,\tilde{s}\in\tilde{S}_1\Big) \tag{4.15e}$$

$$\lambda_2^+(s,\tilde{s}),\ \lambda_2^-(s,\tilde{s})\ \geq\ 0,\quad \forall\Big(s\in S_2^*,\tilde{s}\in\tilde{S}_1\Big) \tag{4.15f}$$

4.2.2 Risoluzione del problema tramite programmazione SDP

Poiché i problemi presentati precedentemente erano tutti astratti, bisognerà ora riformularli per renderli computazionalmente risolvibili: come già fatto per i CMDP Single Controller, si analizzerano gli insiemi $\mathcal{P}(A_1)$, $\mathcal{P}(A_2)$, $\mathcal{M}(A_1)$ e $\mathcal{M}(A_2)$ per rendere i vincoli nei problemi computabili. Iniziamo l'analisi dall'insieme dei polinomi univariati nelle azioni, cioè dai vincoli (4.12a) e (4.13a): l'obiettivo sarà quello di esplicitare i coefficienti dei polinomi in modo da riformulare i relativi vincoli in programmazione semidefinita positiva.

Si definisca con d_s il grado del polinomio per univariato per un dato stato s $\in S^*$, quindi si considerino i vettori delle azioni, indicizzati dal grado, definiti come segue:

$$[a_1]_{d_s} = [1, a_1, a_1^2, \ldots, a_1^{d_s}]^T, \quad [a_2]_{d_s} = [1, a_2, a_1^2, \ldots, a_2^{d_s}]^T.$$

Si definisca poi con $R(s)$ la matrice dei coefficienti del polinomio $r(s, a_1, a_2)$, e con $\overline{\nu}(s) \in \mathbb{R}^{d_s+1}$ il vettore dei primi d_s+1 momenti della misura di probabilità $\nu(s)$. Grazie alle nozioni appena introdotte, riformularemo le varie parti del vincolo (4.12a) in forma vettoriale. I termini dei 2 polinomi relativi alla funzione di reward si potranno quindi riscrivere in forma vettoriale come segue:

$$\sum_{i=0}^{n_s}\sum_{j=0}^{m_s} r_{ij}(s)\mu_i(s){a_2}^j \;=\; \overline{\mu}(s)^T R(s)^T [a_2]_{d_s}. \tag{4.16a}$$

$$\sum_{i=0}^{n_s}\sum_{j=0}^{m_s} r_{ij}(s)\nu_j(s){a_1}^i \;=\; \overline{\nu}(s)^T R(s)^T [a_1]_{d_s}. \tag{4.16b}$$

Si vettorizzino poi i valori della funzione d'utilità per ogni stato del gioco come segue:

$$\overline{v} \;=\; [v(\tilde{s_1}), \ldots, v(\tilde{S_1})]^T, \quad \overline{w} \;=\; [w(\tilde{s_2}), \ldots, w(\tilde{S_2})]^T \tag{4.17}$$

Infine si riscrivano i termini dei 2 polinomi relativi alla funzione di probabilità di transizione in forma vettoriale come di seguito:

$$\sum_{s' \in \tilde{S_2}} p(s', s, a_2) w(s') \;=\; \overline{w}^T Q_{22}(s)^T [a_2]_{d_s}, \tag{4.18a}$$

$$\sum_{s' \in \tilde{S_1}} p(s', s, a_2) \widehat{v}(s') \;=\; \widehat{v}^T Q_{21}(s)^T [a_2]_{d_s}, \tag{4.18b}$$

$$\sum_{s' \in \tilde{S}_1} p(s', s, a_1) v(s') \; = \; \overline{v}^T Q_{11}(s)^T [a_1]_{d_s}, \tag{4.18c}$$

$$\sum_{s' \in \tilde{S}_2} p(s', s, a_1) \widehat{w}(s') \; = \; \widehat{w}^T Q_{12}(s)^T [a_1]_{d_s}, \tag{4.18d}$$

con $Q_{ef}(s)$ che rappresenta la matrice, parametrica negli indici e ed f, dei coefficienti della probabilità di transizione $p(s', s, a_e)$ con $s \in S_e^*$, ed $s' \in \tilde{S}_f$.

Grazie alle definizioni appena mostrate, ed applicando il Lemma 4 presentato nella Sezione 3.1.1, si può affermare che i polinomi univariati nelle azioni definiti rispettivamente in (4.12a) e (4.13a) possono essere riscritti come segue:

$$\mathcal{H}^*(Z_7(s)+(c+d)\frac{1}{2}(L_1 W_7(s) L_2^T + L_2 W_7(s) L_1^T) - (c*d) L_1 W_7(s) L_1^T - L_2 W_7(s) L_2^T) \; =$$

$$= \; -E(s)\overline{w} + R(s)\overline{\mu}^T + \beta Q_{22}(s)\overline{w} + \beta Q_{21}(s)\widehat{v}, \quad \forall s \in S_2^* \tag{4.19}$$

$$\mathcal{H}^*(Z_8(s)+(c+d)\frac{1}{2}(L_1 W_8(s) L_2^T + L_2 W_8(s) L_1^T) - (c*d) L_1 W_8(s) L_1^T - L_2 W_8(s) L_2^T) \; =$$

$$= \; E(s)\overline{v} - R(s)\overline{\nu}(s)^T - \beta Q_{11}(s)\overline{v} - \beta Q_{12}(s)\widehat{w}, \quad \forall s \in S_1^* \tag{4.20}$$

con $Z_7 \in \mathcal{S}^{d_s+1}$, $Z_8 \in \mathcal{S}^{d_s+1}$, $W_7 \in \mathcal{S}^{d_s}$ e $Z_8 \in \mathcal{S}^{d_s}$ matrici tutte semidefinite positive, ed $E(s) \in \mathbb{R}^{d_s+1}$ matrice che contiene tutti 0 a parte un 1 in posizione $(1, s)$.

Ora passiamo all'analisi degli insieme dei momenti delle misure di probabilità, relativi ai vincoli (4.12b) e (4.13b), e dei successivi vincoli sul momento di ordine 0, cioè (4.12c) e (4.13c). Grazie al Lemma 6 presentato nella Sezione 3.1.2, possiamo esprimere i vincoli presenti nel problema primale CPP3, cioè (4.12b) e (4.12c), nell'intervallo $[c, d]$ delle azioni come segue:

$$e_1^T [\mu_0(s), \ldots, \mu_n(s)] \; = \; 1, \quad \forall s \in S_2^* \tag{4.21a}$$

$$-c * \mathcal{H}([\mu_0(s), \ldots, \mu_n(s)]^T) + \mathcal{H}([\mu_1(s), \ldots, \mu_{n+1}(s)]^T) \; \succeq \; 0, \quad \forall s \in S_2^* \tag{4.21b}$$

$$d * \mathcal{H}([\mu_0(s), \ldots, \mu_n(s)]^T) - \mathcal{H}([\mu_1(s), \ldots, \mu_{n+1}(s)]^T) \; \succeq \; 0, \quad \forall s \in S_2^* \tag{4.21c}$$

Sempre grazie al Lemma 6, possiamo esprimere i vincoli presenti nel problema primale CPP4, cioè (4.13b) e (4.13c), nell'intervallo $[c, d]$ delle azioni come segue:

$$e_1^T[\nu_0(s), \ldots, \nu_m(s)] \;=\; 1, \quad \forall s \in S_1^* \tag{4.22a}$$

$$-c * \mathcal{H}([\nu_0(s), \ldots, \nu_m(s)]^T) + \mathcal{H}([\nu_1(s), \ldots, \nu_{m+1}(s)]^T) \;\succeq\; 0, \quad \forall s \in S_1^* \tag{4.22b}$$

$$d * \mathcal{H}([\nu_0(s), \ldots, \nu_m(s)]^T) - \mathcal{H}([\nu_1(s), \ldots, \nu_{m+1}(s)]^T) \;\succeq\; 0, \quad \forall s \in S_1^* \tag{4.22c}$$

Infine, per quanto riguarda l'insieme $\mathcal{F}_{L_V}$, basterà tornare ad esprimere i relativi vincoli in forma estesa come fatto per i problemi CPM3 (4.10) e CPM4 (4.11).

Riscrittura dei problemi primali Ora possiamo costruire il primo problema primale formulato in programmazione SDP, chiamato CSP3 ed equivalente a CPP3 (4.12), tramite le espressioni (4.19) (4.21) come segue:

$$\min_{w(s),w(\tilde{s})} \sum_{s\in S_2^*} w(s) + \sum_{\tilde{s}\in\tilde{S}_2} w(\tilde{s}),$$

subject to

$$\mathcal{H}^*(Z_7(s)+(c+d)\frac{1}{2}(L_1W_7(s)L_2^T+L_2W_7(s)L_1^T)-(c*d)L_1W_7(s)L_1^T-L_2W_7(s)L_2^T) =$$
$$= -E(s)\overline{w} + R^T(s)\overline{\mu}(s) + \beta Q_{22}(s)\overline{w} + \beta Q_{21}(s)\widehat{v}, \quad \forall s \in S_2^* \tag{4.23a}$$

$$e_1^T\overline{\mu}(s) = 1, \quad \forall s \in S_2^* \tag{4.23b}$$

$$-c * \mathcal{H}([\mu_0(s),\ldots,\mu_n(s)]^T) + \mathcal{H}([\mu_1(s),\ldots,\mu_{n+1}(s)]^T) \succeq 0, \quad \forall s \in S_2^* \tag{4.23c}$$

$$d*\mathcal{H}([\mu_0(s),\ldots,\mu_n(s)]^T)-\mathcal{H}([\mu_1(s),\ldots,\mu_{n+1}(s)]^T) \succeq 0, \quad \forall s \in S_2^* \tag{4.23d}$$

$$w(s) - w(s') \geq -L_V d(s,s'), \quad \forall\Big(s \in S_2^*, s' \in \tilde{S}_2\Big) \tag{4.23e}$$

$$-w(s') \geq -L_V d(s,s') - \widehat{v}(s), \quad \forall\Big(s \in S_1^*, s' \in \tilde{S}_2\Big) \tag{4.23f}$$

$$-w(s) + w(s') \geq -L_V d(s,s'), \quad \forall\Big(s \in S_2^*, s' \in \tilde{S}_2\Big) \tag{4.23g}$$

$$w(s') \geq -L_V d(s,s') + \widehat{v}(s), \quad \forall\Big(s \in S_1^*, s' \in \tilde{S}_2\Big) \tag{4.23h}$$

$$W_7(s),\, Z_7(s) \succeq 0, \quad \forall s \in S_2^* \tag{4.23i}$$

Il secondo problema primale formulato in programmazione SDP, chiamato CSP4, equivalente a CPP4 (4.13) e costruito tramite le espressioni (4.20) (4.22), sarà invece definito come segue:

$$\min_{v(s),v(\tilde{s})} \sum_{s\in S_1^*} v(s) + \sum_{\tilde{s}\in\tilde{S}_1} v(\tilde{s}),$$

subject to

$$\mathcal{H}^*(Z_8(s)+(c+d)\frac{1}{2}(L_1W_8(s)L_2^T+L_2W_8(s)L_1^T)-(c*d)L_1W_8(s)L_1^T-L_2W_8(s)L_2^T) =$$
$$= E(s)\overline{v} - R(s)\overline{\nu}(s) - \beta Q_{11}(s)\overline{v} - \beta Q_{12}(s)\widehat{w}, \quad \forall s \in S_1^* \tag{4.24a}$$

$$e_1^T\overline{\nu}(s) = 1, \quad \forall s \in S_1^* \tag{4.24b}$$

$$-c*\mathcal{H}([\nu_0(s),...,\nu_m(s)]^T)+\mathcal{H}([\nu_1(s),...,\nu_{m+1}(s)]^T) \succeq 0, \quad \forall s \in S_1^* \tag{4.24c}$$

$$d*\mathcal{H}([\nu_0(s),...,\nu_m(s)]^T)-\mathcal{H}([\nu_1(s),...,\nu_{m+1}(s)]^T) \succeq 0, \quad \forall s \in S_1^* \tag{4.24d}$$

$$v(s) - v(s') \geq -L_V d(s,s'), \quad \forall\Big(s \in S_1^*, s' \in \tilde{S}_1\Big) \tag{4.24e}$$

$$-v(s') \geq -L_V d(s,s') - \widehat{w}(s), \quad \forall\Big(s \in S_2^*, s' \in \tilde{S}_1\Big) \tag{4.24f}$$

$$-v(s) + v(s') \geq -L_V d(s,s'), \quad \forall\Big(s \in S_1^*, s' \in \tilde{S}_1\Big) \tag{4.24g}$$

$$v(s') \geq -L_V d(s,s') + \widehat{w}(s), \quad \forall\Big(s \in S_2^*, s' \in \tilde{S}_1\Big) \tag{4.24h}$$

$$W_8(s),\, Z_8(s) \succeq 0, \quad \forall s \in S_1^* \tag{4.24i}$$

La soluzione del primo problema primale CSP3 restituisce:

- il vettore dei valori del gioco $\overline{w}$, relativo agli stati $s \in S_2^*$ in cui il secondo giocatore decide le probabilità di transizione;
- il vettore dei momenti delle misure $[\mu_0(s), \ldots, \mu_{n+1}(s)]$ relativo al giocatore 1 per ogni stato $s \in S_2^*$ in cui è il secondo giocatore a controllare le probabilità di transizione.

Lemma 22. Il problema di programmazione semidefinita positiva CSP3 (4.23) risolve esattamente il problema di ottimizzazione polinomiale CPP3 (4.12).

Dimostrazione. La disuguaglianza polinomiale (4.12a) nel problema CPP3 ha come vettore dei coefficienti $-E_s\overline{v} - R(s)\overline{\nu}(s) - \beta Q_{11}(s)\overline{v} - \beta Q_{12}(s)\widehat{w}$, come mostrato dalle espressioni (4.16a) e (4.18). La dimostrazione si ha quindi per diretta conseguenza del Lemma 4, riguardante la rappresentazione di polinomi univariati non negativi nell'intervallo $[c, d]$ tramite formulazione in programmazione semidefinita positiva, applicato come mostrato nell'espressione (4.19), e del Lemma 6, riguardante la rappresentazione delle sequenze valide di momenti di misure di probabilità non negative supportate nell'intervallo $[c, d]$, applicato come mostrato nell'espressione (4.21). □

La soluzione del secondo problema primale CSP4 invece restituisce:

- il vettore dei valori del gioco $\overline{v}$, relativo agli stati $s \in S_1^*$ in cui il primo giocatore decide le probabilità di transizione;
- il vettore dei momenti delle misure $[\nu_0(s), \ldots, \nu_{m+1}(s)]$ relativo al giocatore 2 per ogni stato $s \in S_1^*$ in cui è il primo giocatore a controllare le probabilità di transizione.

Lemma 23. Il problema di programmazione semidefinita positiva CSP4 (4.24) risolve esattamente il problema di ottimizzazione polinomiale CPP4 (4.13).

Dimostrazione. La disuguaglianza polinomiale (4.13a) nel problema CPP3 ha come vettore dei coefficienti $E(s)\overline{v} - R(s)\overline{\nu}(s) - \beta Q_{11}(s)\overline{v} - \beta Q_{12}(s)\widehat{w}$, come mostrato dalle espressioni (4.16b) e (4.18). La dimostrazione si ha quindi per

diretta conseguenza del Lemma 4, riguardante la rappresentazione di polinomi univariati non negativi nell'intervallo $[c, d]$ tramite formulazione in programmazione semidefinita positiva, applicato come mostrato nell'espressione (4.20), e del Lemma 6, riguardante la rappresentazione delle sequenze valide di momenti di misure di probabilità non negative supportate nell'intervallo $[c, d]$, applicato come mostrato nell'espressione (4.22). □

Riscrittura dei problemi duali Per completare la ricostruzione del problema, presentiamo ora la riformulazione relativa ai problemi duali CPD3 (4.14) e CPD4 (4.15). Si eviterà di mostrare tutti i vari passaggi intermedi per ottenere le riformulazioni dei due problemi duali, ma verrà presentata solo la notazione necessaria.

Si definisca con $\overline{q_{ef}}(s, \tilde{s})$ il vettore dei coefficienti del polinomio $p(s', s, a_e)$, parametrico nell'indice (e, f), con $s \in S_e^*$, ed s' $\in \tilde{S}_f$. Si definisca poi con $\overline{\delta}(s, s')$ un vettore formato da tutti 0, a parte il primo elemento che assume il valore 1 se $s = s'$, con $s, s' \in S^*$, 0 altrimenti.

Il primo problema duale formulato in programmazione SDP, chiamato CSD3 ed equivalente a CPD3 (4.14), sarà così riformulato:

$$\max_{\eta(s),\lambda_3^+(s,\tilde{s}),\lambda_3^-(s,\tilde{s}),\lambda_4^+(s,\tilde{s}),\lambda_4^-(s,\tilde{s}),\overline{\varepsilon}(s)} \sum_{s\in S_2^*}\Big[\eta(s)-\beta\overline{\varepsilon}(s)^T\Big(Q_{21}(s)\widehat{v}\Big)\Big]$$

$$-\sum_{s\in S_2^*}\sum_{\tilde{s}\in\tilde{S}_2} L_V * d(s,\tilde{s})\Big(\lambda_3^+(s,\tilde{s})+\lambda_3^-(s,\tilde{s})\Big)$$

$$-\sum_{s\in S_1^*}\sum_{\tilde{s}\in\tilde{S}_2}\Big[L_V*d(s,\tilde{s})+\widehat{v}(s)\Big](\lambda_4^+(s,\tilde{s}))-\sum_{s\in S_1^*}\sum_{\tilde{s}\in\tilde{S}_2}\Big[L_V*d(s,\tilde{s})-\widehat{v}(s)\Big](\lambda_4^-(s,\tilde{s}))$$

subject to

$$\mathcal{H}^*(J_7(s)+(c+d)\frac{1}{2}(L_1K_7(s)L_2^T+L_2K_7(s)L_1^T)-(c*d)L_1K_7(s)L_1^T-L_2K_7(s)L_2^T) = \\ = -R(s)\overline{\varepsilon}(s)-\overline{\eta}(s)e_2, \quad \forall s\in S_2^* \tag{4.25a}$$

$$-c*\mathcal{H}([\varepsilon_0(s),\ldots,\varepsilon_m(s)]^T)+\mathcal{H}([\varepsilon_1(s),\ldots,\varepsilon_{m+1(s)}]^T)\succeq 0, \quad \forall s\in S_2^* \tag{4.25b}$$

$$d*\mathcal{H}([\varepsilon_0(s),\ldots,\varepsilon_m(s)]^T)-\mathcal{H}([\varepsilon_1(s),\ldots,\varepsilon_{m+1}(s)]^T)\succeq 0, \quad \forall s\in S_2^* \tag{4.25c}$$

$$\sum_{s\in S_2^*}\Big(\overline{\delta}(s,s')\Big)\overline{\varepsilon}(s)^T-\sum_{\tilde{s}\in\tilde{S}_2}(\lambda_3^+(s,\tilde{s})-\lambda_3^-(s,\tilde{s})) = 1, \quad \forall s\in S_2^* \tag{4.25d}$$

$$\sum_{s\in S_2^*}\Big(-\beta\overline{q_{22}}(s,\tilde{s})\Big)\overline{\varepsilon}(s)^T+\sum_{s'\in S_2^*}\Big(\lambda_3^+(s',\tilde{s})-\lambda_3^-(s',\tilde{s})\Big)+ \\ +\sum_{s'\in S_1^*}(\lambda_4^+(s',\tilde{s})-\lambda_4^-(s',\tilde{s})) = 1, \quad \forall\tilde{s}\in\tilde{S}_2 \tag{4.25e}$$

$$\lambda_3^+(s,\tilde{s}),\ \lambda_3^-(s,\tilde{s})\ \geq\ 0, \quad \forall\Big(s\in S_2^*,\tilde{s}\in\tilde{S}_2\Big) \tag{4.25f}$$

$$\lambda_4^+(s,\tilde{s}),\ \lambda_4^-(s,\tilde{s})\ \geq\ 0, \quad \forall\Big(s\in S_1^*,\tilde{s}\in\tilde{S}_2\Big) \tag{4.25g}$$

$$J_7(s),\ K_7(s)\succeq 0, \quad \forall s\in S_2^* \tag{4.25h}$$

Il secondo problema duale formulato in programmazione SDP, chiamato CSD4 ed equivalente a CPD4 (4.15), sarà invece così riformulato:

$$\max_{\alpha(s),\lambda_1^+(s,\tilde{s}),\lambda_1^-(s,\tilde{s}),\lambda_2^+(s,\tilde{s}),\lambda_2^-(s,\tilde{s}),\overline{\epsilon}(s)} \sum_{s\in S_1^*}\Big[\alpha(s)+\beta\overline{\epsilon}(s)^T\Big(Q_{12}(s)\widehat{w}\Big)\Big]$$

$$-\sum_{s\in S_1^*}\sum_{\tilde{s}\in\tilde{S}_1} L_V * d(s,\tilde{s})\Big(\lambda_1^+(s,\tilde{s})+\lambda_1^-(s,\tilde{s})\Big)$$

$$-\sum_{s\in S_2^*}\sum_{\tilde{s}\in\tilde{S}_1}\Big[L_V*d(s,\tilde{s})+\widehat{w}(\widehat{s})\Big](\lambda_2^+(s,\tilde{s}))-\sum_{s\in S_2^*}\sum_{\tilde{s}\in\tilde{S}_1}\Big[L_V*d(s,\tilde{s})-\widehat{w}(\widehat{s})\Big](\lambda_2^-(s,\tilde{s}))$$

subject to

$$\mathcal{H}^*(J_8(s)+(c+d)\frac{1}{2}(L_1K_8(s)L_2^T+L_2K_8(s)L_1^T)-(c*d)L_1K_8(s)L_1^T-L_2K_8(s)L_2^T) =$$
$$= R(s)^T\overline{\epsilon}(s)-\overline{\alpha}(s)e_1, \quad \forall s\in S_1^* \tag{4.26a}$$

$$-c*\mathcal{H}([\epsilon_0(s),\ldots,\epsilon_n(s)]^T)+\mathcal{H}([\epsilon_1(s),\ldots,\epsilon_{n+1(s)}]^T)\succeq 0, \quad \forall s\in S_1^* \tag{4.26b}$$
$$d*\mathcal{H}([\epsilon_0(s),\ldots,\epsilon_n(s)]^T)-\mathcal{H}([\epsilon_1(s),\ldots,\epsilon_{n+1}(s)]^T)\succeq 0, \quad \forall s\in S_1^* \tag{4.26c}$$

$$\sum_{s\in S_1^*}(E_s)^T\overline{\epsilon}(s)+\sum_{\tilde{s}\in\tilde{S}_1}(\lambda_1^+(s,\tilde{s})-\lambda_1^-(s,\tilde{s})) = 1, \quad \forall s\in S_1^* \tag{4.26d}$$

$$\sum_{s\in S_1^*}\Big(-\beta\overline{q_{11}}(s,\tilde{s})\Big)\overline{\epsilon}(s)^T-\sum_{s'\in S_1^*}\Big(\lambda_1^+(s',\tilde{s})-\lambda_1^-(s',\tilde{s})\Big)+$$
$$-\sum_{s'\in S_2^*}(\lambda_2^+(s',\tilde{s})-\lambda_2^-(s',\tilde{s})) = 1, \quad \forall\tilde{s}\in\tilde{S}_1 \tag{4.26e}$$

$$\lambda_1^+(s,\tilde{s}),\ \lambda_1^-(s,\tilde{s})\ \geq\ 0, \quad \forall\Big(s\in S_1^*,\tilde{s}\in\tilde{S}_1\Big) \tag{4.26f}$$

$$\lambda_2^+(s,\tilde{s}),\ \lambda_2^-(s,\tilde{s})\ \geq\ 0, \quad \forall\Big(s\in S_2^*,\tilde{s}\in\tilde{S}_1\Big) \tag{4.26g}$$

$$J_8(s),\ K_8(s)\ \succeq\ 0, \quad \forall s\in S_1^* \tag{4.26h}$$

La soluzione del primo problema duale SDP, cioè CSD3 (4.25) restituisce il vettore dei momenti non normalizzato $[\varepsilon_0(s), \ldots, \varepsilon_{m+1}(s)]$ relativo al giocatore 2 per ogni stato $s \in S_2^*$ in cui è lui stesso a controllare le probabilità di transizione.

Lemma 24. Il problema di programmazione semidefinita positiva CSD3 (4.25) risolve esattamente il problema di ottimizzazione polinomiale CPD3 (4.14).

Dimostrazione. La disuguaglianza polinomiale (4.14a) nel problema PD3 ha come vettore dei coefficienti $-R(s)\overline{\varepsilon}(s) - \overline{\eta}(s)e_2$. La dimostrazione si ha quindi per diretta conseguenza del Lemma 4, riguardante la rappresentazione di polinomi univariati non negativi nell'intervallo $[c, d]$ tramite formulazione in programmazione semidefinita positiva, e del Lemma 6, riguardante la rappresentazione delle sequenze valide di momenti di misure di probabilità non negative supportate nell'intervallo $[c, d]$. □

Per quanto riguarda invece il secondo problema duale SDP, cioè CSD4 (4.26), la sua soluzione restituisce il vettore dei momenti non normalizzato $[\epsilon_0(s), \ldots, \epsilon_{n+1}(s)]$ relativo al giocatore 1 per ogni stato s $\in S_1^*$ in cui è lui stesso a controllare le probabilità di transizione.

Lemma 25. Il problema di programmazione semidefinita positiva CSD4 (4.26) risolve esattamente il problema di ottimizzazione polinomiale CPD4 (4.15).

Dimostrazione. La disuguaglianza polinomiale (4.15a) nel problema CPD4 ha come vettore dei coefficienti $R(s)^T\overline{\epsilon}(s) - \overline{\alpha}(s)e_1$. La dimostrazione si ha quindi per diretta conseguenza del Lemma 4, riguardante la rappresentazione di polinomi univariati non negativi nell'intervallo $[c, d]$ tramite formulazione in programmazione semidefinita positiva, e del Lemma 6, riguardante la rappresentazione delle sequenze valide di momenti di misure di probabilità non negative supportate nell'intervallo $[c, d]$. □

Dato che le soluzioni di questi 2 problemi duali, come si sarà già notato, restituiscono i vettori dei momenti non normalizzati, si applichi per prima

cosa la procedura di normalizzazione illustrata dall'espressione (3.32) all'interno della Sezione 3.2.2; quindi, applicando successivamente la procedura di ricostruzione delle strategie miste dai momenti indicata in Sezione 3.1.4, si ricostruiranno le coppie supporto-pesi relative alle strategie miste di ogni giocatore in tutti gli stati in cui le transazioni sono governate da loro stessi.

Ora abbiamo una visione completa del problema di risoluzione di un MDP con spazio degli stati continuo, spazio delle azioni continuo e con proprietà di Swtiching Controller: possiamo quindi presentare l'algoritmo che permette di calcolarne l'equilibrio stazionario approssimato, con il relativo vettore dei valori approssimati del gioco.

4.2.3 Algoritmo di risoluzione di un CMDP polinomiale Switching Controller

L'algoritmo che andremo a presentare si ispira a quello presentato in Sezione 3.4.4 per risolvere problemi riguardanti MDP polinomiali Swtiching Controller con spazio degli stati finito. Questo permetterà di ottenere il vettore dei valori e le strategie miste dei giocatori nel gioco in un punto di equilibrio approssimato: questo non può essere attualmente definito né come un equilibrio minmax né come un ϵ-Equilibrio di Nash, in quanto non può essere formulato, per questo particolare tipo di problema con spazio degli stati continuo, un equivalente problema MDP di Best Response. A tal proposito, si fanno le seguenti considerazioni:

- l'algoritmo non avrà garanzia di ottimalità della soluzione, ma avrà solo un bound sull'errore massimo di approssimazione della soluzione reale con quella calcolata;
- l'algoritmo iterativo, non potendo più contare sulla valutazione di un ϵ-equilibrio, dovrà avere una diversa forma di convergenza.

Si ricorda inoltre che la soluzione dell'algoritmo è considerata approssimata in quanto basata sulla risoluzione di un problema che non sfrutta tutto lo spazio continuo degli stati, ma solo una sua particolare campionatura.

Questo nuovo algoritmo sarà quindi composto di 3 fasi (invece che 4): una prima fase di inizializzazione dei valori preiterativa, quindi 2 fasi che compongono il corpo iterativo dell'algoritmo.

Inizializzazione dell'algoritmo In questa fase iniziale vengono inizializzati a zero due vettori ausiliari $\overline{v}_{copia}$ e $\overline{w}_{copia}$, che fungeranno successivamente da copie dei vettori dei valori approssimati calcolati tramite l'algoritmo dai 2

problemi primali CSP3 e CSP4, Quindi si fissa il valore della variabile $\gamma > 0$ che ci servirà successivamente per la nostra condizione di termine dell'algoritmo. Infine si inizializza il vettore $\overline{w}^0(s) = 0$, $\forall s \in S_2^*$ per la prima iterazione dell'algoritmo.

Calcolo delle soluzioni delle coppie di SDP primali e duali Questa rappresenta la prima delle 2 fasi del ciclo iterativo dell'algoritmo. Si risolva il problema primale CSP4 (4.24) ed il corrispondente problema duale CSD4 (4.26), assumendo $\widehat{w}(s) = \overline{w}^t(s)$, $\forall s \in S_2^*$, con t che rappresenta il numero di iterazioni già effettuate dall'algoritmo. Sia $(\overline{v}^t(s), \overline{\mu}^t(s), \overline{\nu}^t(s))$, per ogni $s \in S_1^*$, la soluzione ottima approssimata a questa coppia di problemi. Si risolva successivamente il problema primale CSP3 (4.23) ed il corrispondente problema duale CSD3 (4.25), assumendo $\widehat{v}(s) = \overline{v}^t(s)$. Sia $(\overline{w}^{t+1}(s), \overline{\mu}^{t+1}(s), \overline{\nu}^{t+1}(s))$, per ogni $s \in S_2^*$, la soluzione ottima approssimata a questa seconda coppia di problemi.

Verifica della condizione di termine dell'algoritmo In questa seconda fase si calcola la condizione di termine dell'algoritmo seguendo questo semplice schema:

1. si calcolano i vettori differenza $D_1 = \overline{v}^t - \overline{v}_{copia}$ e $D_2 = \overline{w}^t - \overline{w}_{copia}$;
2. si consideri il valore massimo $d_{\max}$ tra tutti gli elementi dei vettori differenza D_1 e D_2: tale valore rappresenta il massimo incremento di utilità che si è ottenuto dopo un ciclo iterativo dell'algoritmo. Poiché si vuole trovare un punto d'equilibrio che sia il più stabile possibile, questo valore dovrà essere molto piccolo per farci dire che siamo arrivati ad un punto di convergenza. Quindi, se il valore di $d_{\max}$ è inferiore a quello della nostra γ, allora l'algoritmo termina; in caso contrario, si incrementa t di un'unità, si copiano i valori attuali del vettore dei valori del gioco nei 2 vettori copia, cioè $\overline{v}_{copia} = \overline{v}^t$ e $\overline{w}_{copia} = \overline{w}^t$, quindi si riparte con una nuova iterazione dell'algoritmo.

4.3 Analisi dell'errore teorico di approssimazione della funzione d'utilità

Nelle precedenti sezioni sono state presentate forme di risoluzione per giochi stocastici, in particolare per gli MDP, con spazio degli stati e spazio delle azioni entrambi continui, e nel caso in cui abbiano sia proprietà di Single Controller sia proprietà di Switching Controller. Un'altra particolarità dei giochi da noi analizzati e risolti è che si assume che le funzioni di reward, di probabilità di transizione, e d'utilità siano tutte Lipschitz-continue. Ovviamente quest'assunzione si rappresenta, all'interno del problema, attraverso la soddisfazione di un vincolo, che quindi introduce un errore di approssimazione nella soluzione del problema stesso.

L'analisi che svolgeremo sarà quindi basata sulla valutazione dell'errore di approssimazione della soluzione di un gioco MDP polinomiale data l'assunzione di Lipschitz-continuità delle funzioni, in particolare ci concentreremo sull'errore relativo ai valori della funzione d'utilità restituiti nella soluzione dei giochi risolti in Sezione 4.1 e 4.2.

I risultati di questa sezione derivano principalmente dalle analisi teoriche presenti in [33] e [34].

Iniziamo quindi quest'analisi introducendo tutti gli elementi necessari alla successiva definizione dell'errore di approssimazione. Si consideri un problema MDP come quello definito in Sezione 4.2. Si definisca con v^* la funzione d'utilità reale del gioco, e con $\tilde{v}$ la sua versione approssimata, derivante dalla soluzione del problema MDP con spazio degli stati campionato. Si definisca poi il seguente operatore, chiamato Operatore di Bellman L.

Lemma 26. Siano $\mathbf{1}$ e $\overline{v}$ due vettori di dimensioni note. Allora si definisce l'operatore di Bellman L come segue:

$$L(\overline{v} + \epsilon\mathbf{1}) \;=\; \beta\epsilon\mathbf{1} + L\overline{v}$$

Si definisca poi il seguente lemma.

Lemma 27. Se $\overline{v}$ è una funzione d'utilità rappresentabile, allora anche $\overline{v}+\epsilon\mathbf{1}$ è una funzione rappresentabile, dove $\mathbf{1}$ è un vettore di tutti 1.

Relativamente a questo lemma, si presenti la seguente definizione.

Definizione 34 (Funzione ammissibile ϵ-transitiva)**.** L'insieme delle funzioni d'utilità ammissibili ϵ-transitive è definito, per ogni $\epsilon > 0$, come segue: $\mathcal{K}(\epsilon) \;=\; v \in \mathbb{R}^{|S|} \;:\; \overline{v} \geq L(\overline{v}) - \epsilon\mathbf{1}$

Lemma 28. Si consideri una funzione d'utilità ammissibile ϵ-transitiva $\overline{v}$, tale che $\overline{v} \geq L(\overline{v}) - \epsilon\mathbf{1}$. Allora vale la seguente disuguaglianza:

$$\overline{v} \geq \overline{v}^* - \frac{\epsilon}{1-\beta}\mathbf{1}. \tag{4.27}$$

Si definisca poi con L_{V^*} la costante di Lipschitz relativa alla funzione d'utilità reale, e con $L_{\tilde{V}}$ la costante di Lipschitz relativa alla funzione d'utilità approssimata. Come già detto, il problema MDP di cui abbiamo mostrato il metodo di risoluzione si basa sulla campionatura degli stati del problema originiale. Di conseguenza, quest'ultima introdurrà una componente d'errore intrinseca al problema, dovuta all'uso della funzione d'utilità approssimata, che definiremo come:

$$\epsilon = \frac{2}{1-\beta} \min_{\tilde{v} \in \mathcal{M}_{L_{\tilde{V}}}} \|\tilde{v} - v^*\|_\infty. \tag{4.28}$$

Relativamente a quest'ultima, si può presentare il seguente teorema, che definisce il valore massimo di ϵ in relazione alle caratteristiche specifiche del problema.

Teorema 25.

$$\epsilon \leq \frac{R_{\max} - R_{\min}}{(1-\beta)^2}(1 - \frac{L_{\tilde{V}}}{L_{V^*}})$$

Dimostrazione. Se $L_{\tilde{V}} \geq L_{V^*}$, allora $v^* \in \mathcal{M}_{L_{\tilde{V}}}$ e $\epsilon = 0$. Altrimenti, si definisce come di seguito la funzione d'utilità $v' \in \mathcal{M}_{L_{\tilde{V}}}$, prodotta traslando v^* in modo tale che $|v^*_{\max}| = |v^*_{\min}|$, quindi scalando il tutto per la componente $\frac{L_{\tilde{V}}}{L_{V^*}}$, infine ritraslando indietro il tutto:

$$v' =, \Big(v^* - \frac{v^*_{\max} + v^*_{\min}}{2}\Big)\frac{L_{\tilde{V}}}{L_{V^*}} + \frac{v^*_{\max} + v^*_{\min}}{2} \in \mathcal{M}_{L_{\tilde{V}}}.$$

Sapendo che, per costruzione del vincolo di Lipschitz, si ha che $v_{\max} \leq \frac{R_{\max}}{1-\beta}$, allora possiamo definire la distanza tra v' e v^* come segue:

$$\begin{aligned} \|v' - v^*\|_\infty &= \frac{v^*_{\max} - v^*_{\min}}{2}\Big(1 - \frac{L_{\tilde{V}}}{L_{V^*}}\Big) \\ &\leq \frac{R_{\max} - R_{\min}}{2(1-\beta)}\Big(1 - \frac{L_{\tilde{V}}}{L_{V^*}}\Big) \end{aligned}$$

La dimostrazione termina sostituendo quest'ultima equazione nell'espressione (4.28). □

Data la campionatura degli stati del problema originiale, si passerà da un numero infinito di stati ad un numero limitato,quindi anche le transizioni da uno stato all'altro risulteranno campionate, introducendo una nuova componente d'errore, definita come ϵ_p.

Definizione 35. La seguente disuguaglianza

$$\epsilon_p(L_{\tilde{V}}) \leq d_{\max}(L_{V^*} + L_{\tilde{V}})$$

limita la componente d'errore relativa alla possibile violazione dei vincoli di Bellman in tutti quei stati che non fanno parte dell'insieme di campionatura del problema, dove $d_{\max}$ che rappresenta la massima distanza da uno stato non presente nell'insieme degli stati campionati allo stato a lui più vicino che però risulti presente nell'insieme.

Date tutte queste nozioni, è ora possibile mostrare l'errore teorico di approssimazione della funzione d'utilità quando si risolve un problema MDP con spazio degli stati continuo attraverso un corrispondente MDP campionato.

Teorema 26. *Si definisca ϵ come in (4.28), ed ϵ_p come in Definizione 35. Sia $\tilde{v}$ la soluzione restituita dall'algoritmo mostrato in Sezione 4.2. Allora vale la seguente espressione:*

$$\|\tilde{v} - v^*\|_\infty \leq \epsilon + 2\frac{\epsilon_p(L_{\tilde{V}})}{1-\beta}. \tag{4.29}$$

Dimostrazione. Grazie al Lemma 28 sappiamo che

$$\overline{v} \geq \overline{v}^* - \frac{\epsilon_p}{1-\beta}\mathbf{1}.$$

Allora possiamo procedere come segue:

$$\begin{aligned}
\|\tilde{v} - v^*\|_\infty &\leq \|\tilde{v} - v^* + \frac{\epsilon_p}{1-\beta}\mathbf{1} - \frac{\epsilon_p}{1-\beta}\mathbf{1}\|_\infty \\
&\leq \|\tilde{v} - v^* + \frac{\epsilon_p}{1-\beta}\mathbf{1}\|_\infty + \frac{\epsilon_p}{1-\beta} \\
&\leq \|\tilde{v} - v^*\|_\infty + 2\frac{\epsilon_p}{1-\beta} \\
&\leq \epsilon + 2\frac{\epsilon_p}{1-\beta}
\end{aligned}$$

□

Capitolo 5

Valutazioni Sperimentali

In questo capitolo vengono presentate diverse valutazioni sperimentali riguardanti le soluzioni proposte nei precedenti capitoli per risolvere giochi stocastici sia con spazio delle azioni continuo e spazio degli stati discreto, sia con spazio delle azioni e degli stati entrambi continui.

La prima sezione di questo capitolo tratterà la valutazione dell'algoritmo di risoluzione di MDP polinomiali Switching Controller con spazio degli stati discreto e spazio delle azioni continuo presentato in Sezione 3.4.4: prima verrà mostrato il generatore creato per la generazione di questi particolari giochi stocastici, quindi verrà fatta un'analisi comparativa tra l'algoritmo da noi implementato con l'equivalente mostrato in [6], infine verrà analizzato il lavoro di ottimizzazione effettuato sui vincoli dei problemi che compongono l'algoritmo.

Nella seconda sezione invece si tratterà la valutazione del problema di risoluzione di giochi stocastici polinomiali Single Controller con spazio degli stati continuo e spazio delle azioni continuo presentato in Sezione 4.1.3: anche qui, prima verrà mostrato il generatore creato per la generazione di questi particolari giochi, quindi verrà fatta un'analisi comparativa tra la nostra soluzione approssimata, quella derivante da una discretizzazione del problema e quella in forma chiusa per il problema chiamato LQR, che poi andremo meglio a definire.

La terza ed ultima sezione tratterà la valutazione dell'algoritmo di risoluzione di giochi stocastici polinomiali Switching Controller con spazio degli stati continuo e spazio delle azioni continuo presentato in Sezione 4.2.3: qui verrà fatta un'analisi dei tempi di computazione e del numero di iterazioni necessari alla convergenza dell'algoritmo.

5.1 Valutazione dell'algoritmo di risoluzione di MDP polinomiali Switching Controller con spazio degli stati discreto

5.1.1 Presentazione del Generatore di MDP polinomiali Switching Controller a stati discreti

I giochi stocastici con spazio degli stati discreto sono giochi concenttualmente molto facili da descrivere e generare, soprattutto quando si hanno pochi stati in gioco. Tuttavia, all'aumentare degli stati in gioco, cresce la difficoltà di generazione, in quanto la funzione di probabilità di transizione deve rispettare alcuni vincoli per ogni stato s in gioco. Questa difficoltà poi cresce ulteriormente se la funzione di transizione deve essere descritta in una forma specifica, in questo caso in forma polinomiale nelle azioni dei giocatori in gioco.

Per risolvere queste difficoltà si è deciso di implementare un generatore automatico di giochi MDP polinomiali con proprietà di Switching Controller, che ora andremo a descrivere in dettaglio. Questo generatore riceverà in ingresso i seguenti parametri in modo da restituire poi una descrizione completa del gioco creato:

- modalità di creazione della funzione di reward da generare;
- modalità di creazione della funzione di transizione da generare;
- suddivisione del controllo sugli stati per ognuno dei 2 giocatori in gioco;
- l'intervallo $[c, d]$ su cui viene supportato lo spazio continuo delle azioni dei giocatori;
- il numero di stati S presenti nel gioco;
- il numero di transizioni T che partono da ogni stato verso gli altri presenti nel gioco, con $T \leq S$;
- il grado dei polinomi delle funzioni di reward e di transizione;
- il valore del fattore di sconto β;
- il file su cui verrà scritta la descrizione completa del gioco creato in base ai parametri precedenti.

Prima di proseguire, è necessaria un'osservazione sui parametri appena mostrati: la modalità di creazione delle funzioni indica, attraverso opportuni simboli, se le funzioni verranno inserite manualmente dall'utente, generate randomicamente attraverso metodi che poi andremo a descrivere, o se verranno considerate alcune funzioni di test.

Questo generatore, nonostante il gran numero di parametri d'ingresso, ha un'unica difficoltà implementativa, cioè la generazione automatica della funzione di transizione. L'altra funzione in gioco infatti, cioè quella di reward, può essere facilmente generata inserendo valori casuali all'interno della matrice superiore sinistra che la descrive per ogni stato del gioco, senza che debbano essere soddisfatti particolari vincoli.

Si consideri un MDP definito come in Sezione 3.4.2. Presentiamo quindi i vincoli che dovrà rispettare la funzione di transizione generata:

$$\sum_{s' \in S} p(s', s, a) \; = \; 1, \quad \forall s \in S, \forall a \in A = [c, d] \tag{5.1a}$$

$$p(s', s, a) \; \geq \; 0, \quad \forall s \in S, \forall s' \in S, \forall a \in A = [c, d] \tag{5.1b}$$

Il primo vincolo ci dice che, per ogni stato in gioco, la somma delle probabilità di transizione nella variabile d'azione verso gli altri stati sia normalizzata ad uno. Il secondo vincolo ci dice invece che ogni probabilità di transizione da uno stato s verso un altro stato s' deve essere non negativa nell'insieme delle azioni supportate.

Come si può subito notare, la generazione della funzione di transizione può essere riformulata come la soluzione del seguente problema di programmazione lineare:

$$\min_{p(s',s,a)} \sum_{s \in S} \sum_{a \in A} \sum_{s' \in S} c(s', s, a) p(s', s, a)$$

subject to

$$\sum_{s' \in S} p(s', s, a) \; = \; 1, \quad \forall s \in S, \forall a \in A = [c, d] \tag{5.2a}$$

$$p(s', s, a) \; \geq \; 0, \quad \forall s \in S, \forall s' \in S, \forall a \in A = [c, d] \tag{5.2b}$$

dove $c(s', s, a)$ rappresenta la funzione dei coefficienti, creata in modo automatico dal generatore, associata alla funzione di transizione.

Il problema precedente può essere riscritto in forma matriciale come segue:

$$\min_{\overline{p}(s',s)} \sum_{s\in S}\sum_{s'\in S} \overline{c}(s',s)\overline{p}(s',s)^T$$

subject to

$$\sum_{s'\in S} \overline{p}(s',s)^T \;=\; e_1^T, \quad \forall s \in S \tag{5.3a}$$

$$\overline{p}(s',s)^T \;\geq\; 0, \quad \forall s \in S, \forall s' \in S \tag{5.3b}$$

dove $\overline{c}(s',s)$ e $\overline{p}(s',s)$ rappresentano i vettori contenenti, rispettivamente, i coefficienti, indicizzati dal grado del polinomio, di $c(s',s,a)$ e di $p(s',s,a)$.

Infine, applicando le tecniche mostrate nei capitoli precedenti, possiamo creare il seguente problema di programmazione SDP, equivalente al precedente:

$$\min_{\overline{p}(s',s)} \sum_{s\in S}\sum_{s'\in S} \overline{c}(s',s)\overline{p}(s',s)^T$$

subject to

$$\sum_{s'\in S} \overline{p}(s',s)^T \;=\; e_1^T, \quad \forall s \in S \tag{5.4a}$$

$$\mathcal{H}^*(Z(s',s)+(c+d)\frac{1}{2}(L_1W(s',s)L_2^T+L_2W(s',s)L_1^T)-(c*d)L_1W(s',s)L_1^T-L_2W(s',s)L_2^T) =$$

$$= \overline{p}(s',s)^T, \quad \forall s \in S, \forall s' \in S \tag{5.4b}$$

$$W(s',s),\, Z(s',s) \;\succeq\; 0, \quad \forall s \in S, \forall s' \in S \tag{5.4c}$$

Per concludere, possiamo dire che grazie al generatore appena descritto è possibile generare in modo del tutto automatico problemi MDP polinomiali Switching Controller con spazio degli stati discreto di dimensioni e parametri variabili a piacere.

5.1.2 Analisi comparativa di 2 diverse implementazioni dell'algoritmo

Ora passiamo alla presentazione dell'analisi comparativa tra 2 diverse implementazioni dell'algoritmo descritto in Sezione 3.4.4, basato sul tempo di computazione per raggiungere la soluzione d'equilibrio di diverse istanze del problema. La prima implementazione, mostrata più dettagliatamente in [6] è stata realizzata utilizzando YALMIP [21], cioè un toolbox per MATLAB che

Tempo (millisec.)	2 stati	3 stati	4 stati
Grado 4	18150	18100	18050
Grado 6	18350	18200	18050
Grado 8	18250	18100	17950

Tabella 5.1: YALMIP: Tempi di computazione.

Tempo (millisec.)	2 stati	3 stati	4 stati
Grado 4	121	275	372
Grado 6	172	379	503
Grado 8	482	505	686

Tabella 5.2: SDPA: Tempi di computazione.

permette la modellizzazione efficiente di problemi di ottimizzazione; a questo toolobox viene poi affiancato un risolutore esterno chiamato SeDuMi, cioè un pacchetto software che risolve problemi di programmazione semidefinita positiva. La seconda implementazione, cioè quella che è stata effettivamente implementata durante questo progetto di tesi, è stata realizzata utilizzando SDPA [22], cioè un pacchetto software implementato in linguaggio C++ che risolve problemi in programmazione semidefinita positiva; a questo pacchetto vengono affiancati anche i pacchetti LAPACK e BLAS [23], il primo per la computazione delle operazioni sulle matrici ed il secondo per l'ottimizzazione delle operazioni algebriche lineari. La scelta di questo particolare pacchetto software è dovuta al fatto che le sue prestazioni risultavano tra le migliori nei confronti degli altri risolutori di problemi SDP presenti attualmente.

La comparazione delle due diverse implementazioni per quanto riguarda il numero di iterazioni dell'algoritmo per raggiungere la convergenza non viene mostrata perché poco significativa: infatti entrambe le implementazioni, come era giusto aspettarsi, convergono alla stessa soluzione d'equilibrio ϵ-Nash, con $\epsilon < 10^{-2}$, in un numero di iterazioni pari a 2, sia al variare degli stati di gioco sia al variare del grado del polinomio.

Mostriamo quindi la valutazione dei tempi di computazione dell'algoritmo per raggiungere la soluzione d'equilibrio nelle due diverse implementazioni, al variare del grado del polinomio e del numero di stati in gioco: nella Tabella 5.1 vengono mostrati i tempi di computazione, in millisecondi, per l'implementazione tramite YALMIP, mentre nella Tabella 5.2 vengono mostrati i tempi di computazione, in millisecondi, per l'implementazione tramite SD-

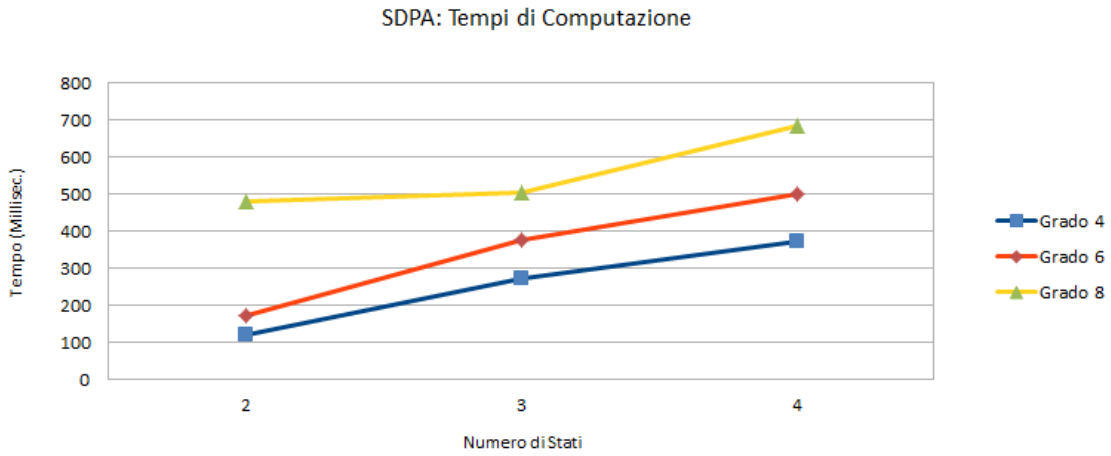

Figura 5.1: Analisi sperimentale del tempo di computazione dell'algoritmo al variare del numero di stati in gioco e del grado dei polinomi.

PA. I tempi mostrati rappresentano le medie su 20 computazioni dell'algoritmo su problemi MDP polinomiali Switching Controller caratterizzati, di volta in volta, da 2, 3, o 4 stati di gioco e da un grado dei polinomi pari a 4, 6 o 8. Da una prima analisi dei dati, si nota subito come la nuova implementazione tramite SDPA dia dei vantaggi notevoli in termini di tempo di esecuzione dell'algoritmo, circa 2 ordini di grandezza, rispetto alla vecchia implementazione tramite YALMIP. Questo è dovuto al fatto che quest'ultima ha un tempo di overhead elevatissimo, che và a mascherare il vero tempo di computazione dell'algoritmo, quindi non è possibile vedere come questo scala al variare dei parametri.

L'analisi di scalabilità quindi è effettuata analizzando i dati derivati da SDPA presenti in Tabella 5.2 e mostrati graficamente in Figura 5.1: da questi si nota che l'aumento del tempo di computazione dipende, come prevedibile, da entrambi i parametri, tuttavia l'influenza maggiore è data dalla variazione del grado dei polinomi scelto per rappresentare le funzioni in gioco. Si può infatti notare che, all'aumentare del grado dei polinomi, le curve passano da una forma quasi lineare ad una sempre più esponenziale.

Per concludere questa prima analisi, si può infine affermare che la nuova implementazione dell'algoritmo tramite SDPA presenta 2 vantaggi significativi:

1. azzera il tempo di overhead;

2. permette di fare un'analisi di scalabilità dell'algoritmo al variare dei parametri del problema.

5.1.3 Analisi relativa all'ottimizzazione dei vincoli del problema

Durante la riformulazione del problema su un intervallo $[c, d]$ generico dello spazio delle azioni A, sono stati analizzati nuovamente tutti i vincoli per cercare di trovare possibili ottimizzazioni. Grazie a questa analisi infatti sono stati individuati alcuni vincoli ridondanti che rallentavano le prestazioni dell'algoritmo in termini di tempi di computazione.

I vincoli ridondanti riguardavano i momenti delle misure di probabilità: infatti veniva inserito inizialmente nella formulazione un set di vincoli riguardanti i primi $m+1$ momenti per l'esplicitazione dell'insieme $\mathcal{M}(A)$, cioè il set di vincoli definito dal Lemma 5, quindi veniva aggiunto un ulterorire set di vincoli per i primi $m+2$ momenti per permettere la ricostruzione delle strategie, cioè il set di vincoli definito dal Lemma 6. Come si può notare dalla lettura dei due lemmi, usando il secondo di questi, cioè quello per la definizione dei primi $m+2$ momenti, si vincolano implicitamente anche i primi $m+1$ momenti ad essere una sequenza valida di momenti di misure di probabilità non negative sull'intervallo di supporto $[c, d]$. L'eliminazione di questi vincoli ci ha portato alla formulazione ottimizzata del problema, che equivale a quella già presentata nella Sezione 3.4.4.

L'analisi sperimentale dei tempi di computazione relativamente alle diverse versioni dell'algoritmo, cioè quella con tutti i vincoli e quella con i vincoli ottimizzati, è presentata in Figura 5.2, mentre in Tabella 5.3 vengono mostrati i guadagni percentuali dell'algoritmo nella sua versione ottimizzata rispetto a quella non ottimizzata. I tempi mostrati in Figura 5.2 rappresentano le medie su 20 computazioni dell'algoritmo nelle sue due versioni con gli stati di gioco che variano da 10 a 20, e il grado dei polinomi che variano con valori uguali a 2, 4 e 6. I dati mostrati in Tabella 5.3 invece rappresentano i

	Grado 2	Grado 4	Grado 6
10 Stati	13,75%	18,15%	14,93%
20 Stati	8,79%	13,22%	12,23%

Tabella 5.3: Problema ottimizzato: guadagno percentuale sui tempi di computazione rispetto al precedente problema con vincoli ridondanti.

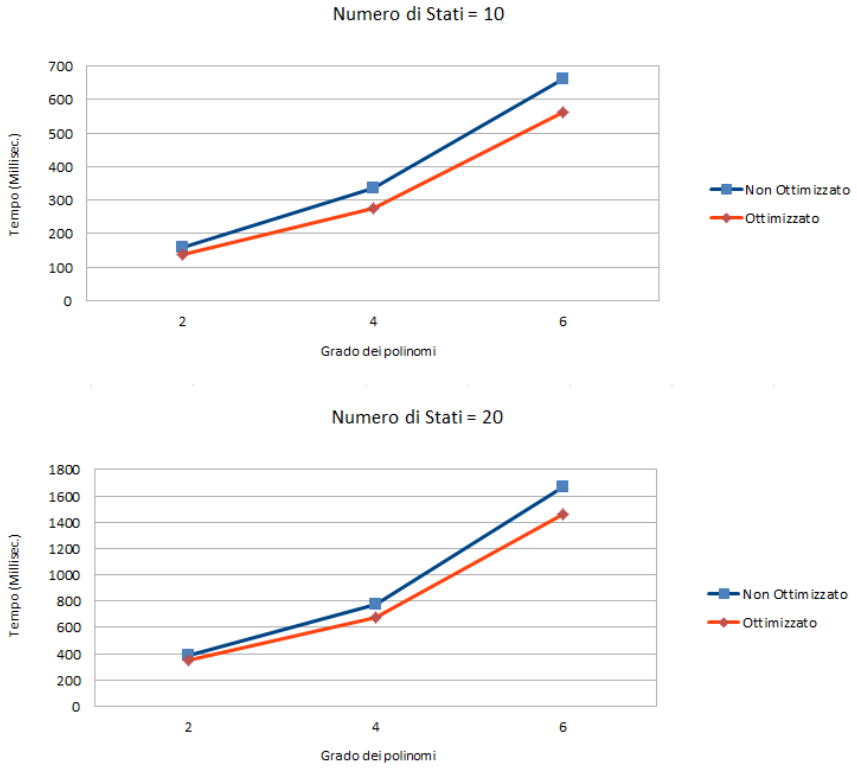

Figura 5.2: Analisi sperimentale dei tempi di computazione relativamente alle due versioni, ottimizzate e non, dell'algoritmo al variare del numero di stati in gioco e del grado del polinomio.

guadagni percentuali della versione ottimizzata rispetto a quella non ottimizzata, guadagni che derivano dalle medie utilizzate per realizzare i grafici in Figura 5.2.

Dall'analisi dei dati mostrati si può dedurre che:

- all'aumentare del numero di stati del gioco, il peso dei vincoli ridondanti eleminati è amplificato più che linearmente, in termini assoluti, sul tempo di computazione dell'algoritmo;

- all'aumentare del numero di stati del gioco, il peso dei vincoli ridondanti eleminati è sempre meno influente, in termini percentuali, sul tempo di computazione dell'algoritmo;

- all'aumentare del grado dei polinomi che descrivono il gioco, il peso dei vincoli ridondanti eleminati è amplificato in modo quasi esponenziale, in termini assoluti, sul tempo di computazione dell'algoritmo;

- all'aumentare del grado dei polinomi che descrivono il gioco, il peso dei vincoli ridondanti eleminati porta ad un guadagno quasi costante, in termini percentuali, sul tempo di computazione dell'algoritmo;

5.2 Valutazione del metodo di risoluzione per MDP polinomiali Single Controller con spazio degli stati continuo

5.2.1 Presentazione del Generatore di MDP polinomiali Single e Switching Controller a stati continui

Come già fatto precedentemente per problemi con spazio degli stati discreto, anche per i problemi descritti da MDP polinomiali con spazio degli stati continuo presenteremo un generatore di giochi automatico ad essa correlato. Rispetto al precedente generatore, le difficoltà relative alla creazione del problema stesso aumentano, in quanto ora le funzioni di probabilità di transizione sono descritte da distribuzioni continue di probabilità sugli infiniti stati di gioco, cioè non sono in forma polinomiale. Si ricorda comunque che gli stati del gioco saranno campionati in modo da creare una versione approssimata del problema.

Questo generatore riceverà in ingresso, oltre ai parametri già presentati in Sezione 5.1.1 per il precedente generatore, anche i seguenti parametri aggiuntivi, in modo da restituire poi una descrizione completa del gioco creato:

- parametro relativo alla modalità di campionatura degli stati nell'intervallo continuo che definisce lo spazio degli stati stesso;

- numero di stati $\tilde{S}$ d'arrivo, di cui non si calcoleranno le strategie dei giocatori, per ogni stato di partenza S^*, di cui invece si calcoleranno le strategie;

- parametro che descrive le proprietà della distribuzione continua di probabilità di transizione.

Alcune note sui parametri appena presentati: nella fase sperimentale, per convenzione, il primo parametro indicherà la distanza tra uno stato e l'altro sotto l'ipotesi di campionatura uniforme degli stati sull'intervallo $[c, d]$, mentre l'ultimo parametro indicherà la deviazione standard sotto l'ipotesi che la distribuzione di probabilità segua un modello gaussiano di media nulla.

Come nel precedente generatore, anche qui le difficoltà risiedono nella funzione di transizione: in questo caso bisognerà infatti prima approssimare la distribuzione di probabilità continua con una funzione polinomiale, quindi bisognerà generare una funzione equivalente a quella appena creata che però soddisfi tutti i vincoli per rappresentare una valida funzione di probabilità.

Per quanto riguarda la prima fase, cioè l'approssimazione polinomiale della distribuzione di probabilità, verrà inizialmente campionato lo spazio degli stati, ottendendo i due insiemi S^* ed $\tilde{S}$, quindi verrà implementato un approssimatore basato sulla tecnica di Chebyshev Approximation, mostrata nel dettaglio in Sezione 4.3.1. Grazie a questo approccio avremo anche garanzia sull'errore di approssimazione legata ad un limite superiore teorico dell'errore stesso.

Si consideri quindi un problema MDP come quello definito in Sezione 4.1.2. Sia definisca con $p_{cheb}(s', s, a)$ la funzione polinomiale approssimata ricavata tramite l'applicazione del metodo di Chebyshev Approximation. La funzione di transizione polinomiale che dovremo ricavare dovrà quindi sia presentare tutti i vincoli mostrati nell'espressione (5.1) sia basarsi sulla funzione polinomiale approssimata $p_{cheb}(s', s, a)$. Per fare quest'ultimo passaggio, dovremo aggiungere un ulteriore vincolo a quelli già presentati: il nuovo vincolo prevederà di minimizzare la distanza $d(s, s')$, in norma infinito, tra il polinomio $p_{cheb}(s', s, a)$ e l'equivalente funzione di transizione polinomiale approssimata $p(s', s, a)$, in modo che si possa mantenere sempre la garanzia sul massimo errore di approssimazione derivata dall'uso dei polinomi di Chebyshev, a meno di un termine additivo $\max d(s, s')$.

La nuova funzione di transizione quindi potrà essere generata tramite la risoluzione del seguente problema di ottimizzazione polinomiale:

$$\min_{d(s,s')} \sum_{s \in S^*} \sum_{s' \in \tilde{S}} d(s, s')$$

subject to

$$\sum_{s' \in \tilde{S}} p(s', s, a) \; = \; 1, \quad \forall s \in S^*, \forall a \in A = [c, d] \tag{5.5a}$$

$$p(s', s, a) \; \geq \; 0, \quad \forall s \in S^*, \forall s' \in \tilde{S}, \forall a \in A = [c, d] \tag{5.5b}$$

$$\|p_{cheb}(s', s, a) - p(s', s, a)\|_\infty \; \leq \; d(s, s'), \quad \forall s \in S^*, \forall s' \in \tilde{S}, \forall a \in A = [c, d] \tag{5.5c}$$

Il problema precedente sarà riscritto in forma matriciale come segue:

$$\min_{d(s,s')} \sum_{s \in S^*} \sum_{s' \in \tilde{S}} d(s, s')$$

subject to

$$\sum_{s' \in \tilde{S}} \overline{p}(s', s)^T \; = \; e_1^T, \quad \forall s \in S^* \tag{5.6a}$$

$$\overline{p}(s', s)^T \; \geq \; 0, \quad \forall s \in S^*, \forall s' \in \tilde{S}, \forall a \in A = [c, d] \tag{5.6b}$$

$$\|\overline{p_{cheb}}(s', s)^T - \overline{p}(s', s)^T\|_\infty \; \leq \; e_1^T d(s, s'), \quad \forall s \in S^*, \forall s' \in \tilde{S} \tag{5.6c}$$

dove $\overline{p_{cheb}}(s', s)$ e $\overline{p}(s', s)$ rappresentano i vettori contenenti, rispettivamente, i coefficienti, indicizzati dal grado del polinomio, di $p_{cheb}(s', s, a)$ e di $p(s', s, a)$.

Infine, applicando le tecniche mostrate nei capitoli precedenti, possiamo creare il seguente problema di programmazione SDP, equivalente al precedente problema di ottimizzazione polinomiale, che genererà i coefficienti dei polinomi rappresentanti la funzione di transizione approssimata:

$$\min_{d(s,\tilde{s})} \sum_{s \in S^*} \sum_{\tilde{s} \in \tilde{S}} d(s,\tilde{s}),$$

subject to

$$H^*(Z(s,\tilde{s})+(a+b)\frac{1}{2}(L_1 W(s,\tilde{s})L_2^T+L_2 W(s,\tilde{s})L_1^T)-(a*b)L_1 W(s,\tilde{s})L_1^T-L_2 W(s,\tilde{s})L_2^T) =$$

$$= \overline{p}(s,\tilde{s})^T, \quad \forall s \in S^*, \forall \tilde{s} \in \tilde{S} \tag{5.7a}$$

$$H^*(A(s,\tilde{s})+(a+b)\frac{1}{2}(L_1 B(s,\tilde{s})L_2^T+L_2 B(s,\tilde{s})L_1^T)-(a*b)L_1 B(s,\tilde{s})L_1^T-L_2 B(s,\tilde{s})L_2^T) =$$

$$= \overline{p}(s,\tilde{s})^T - \overline{p_{cheb}}(s,\tilde{s})^T + e_1^T d(s,\tilde{s}), \quad \forall s \in S^*, \tilde{s} \in \tilde{S} \tag{5.7b}$$

$$H^*(C(s,\tilde{s})+(a+b)\frac{1}{2}(L_1 D(s,\tilde{s})L_2^T+L_2 D(s,\tilde{s})L_1^T)-(a*b)L_1 D(s,\tilde{s})L_1^T-L_2 D(s,\tilde{s})L_2^T) =$$

$$= -\overline{p}(s,\tilde{s})^T + \overline{p_{cheb}}(s,\tilde{s})^T + e_1^T d(s,\tilde{s}), \quad \forall s \in S^*, \forall \tilde{s} \in \tilde{S} \tag{5.7c}$$

$$\sum_{\tilde{s} \in \tilde{S}} \overline{p}(s,\tilde{s})^T = e_1^T, \quad \forall s \in S^* \tag{5.7d}$$

$$W(s,\tilde{s}),\ Z(s,\tilde{s}),\ A(s,\tilde{s}),\ B(s,\tilde{s}),\ C(s,\tilde{s}),\ D(s,\tilde{s}) \succeq 0, \quad \forall s \in S^*, \forall \tilde{s} \in \tilde{S} \tag{5.7e}$$

Per concludere, possiamo dire che grazie al generatore appena presentato è possibile generare in modo del tutto automatico problemi MDP polinomiali Switching Controller, quindi anche Single Controller, con spazio degli stati continuo, ed opportunamente campionato, di dimensioni e parametri variabili a piacere.

5.2.2 Analisi sperimentale del metodo di risoluzione

Prima di iniziare questa analisi, presentiamo il modello di gioco su cui si è basato il nostro esperimento. Il modello in questione è il cosiddetto - Linear Quadratic Regulator (LQR), cioè un regolatore caratterizzato da una funzione di reward lineare quadratica. La scelta di questo modello è dovuta al fatto che è nota la sua soluzione in forma chiusa, cioè per ogni stato del gioco si sà esattamente qual'è il valore della funzione d'utilità ed il valore della policy ottima, quindi può essere usato come benchmark prestazionale.

Descriviamo nel dettaglio questo particolare modello di gioco, trattato in [24]. La funzione di reward si presenta, come già detto, in forma lineare quadratica, quindi polinomiale, come segue:

$$r(s,a) = -s^2 - a^2.$$

Per quanto riguarda la funzione di probabilità di transizione, questa segue il seguente modello di transizione:

$$s_{t+1} \;=\; s_t + a_t + rumore,$$

dove il rumore rappresenta una ditribuzione gaussiana definita dalla deviazione standard σ.

I valori ottimi in forma chiusa di questo modello sono dati dalle seguenti espressioni per quanto riguarda, rispettivamente, la policy ottima e il valore della funzione d'utilità:

$$a(s) = -k * s \tag{5.8}$$

$$v(s) \;=\; -\frac{1}{2}P_\beta s^2 - \frac{1}{2(1-\beta)}(2+\beta P_\beta)\sigma^2,$$

dove β è il fattore di sconto, k è definita come $k = 1 - \frac{2}{1+2\beta+\sqrt{4\beta^2+1}}$, e P_β rappresenta la soluzione dell'equazione di Riccati, dati i parametri descrittivi del problema.

Ora possiamo passare alla presentazione ed analisi dei test sperimentali. In questi test sono stati usati i seguenti valori dei parametri descrittivi del problema: $\beta = 0.5$, $\sigma = 0.5$, intervallo delle azioni e degli stati $[c, d] = [-4, 4]$, grado dei polinomi pari ad 8, valori di L_V e numero di stati S^* ed $\tilde{S}$ variabili.

Questa parte dei test è stata preparata attraverso 3 fasi consecutive tra loro: nella prima è stato risolto il problema LQR in forma chiusa, in modo da ottenere tutti i valori di benchmark esatti, nella seconda il problema LQR è stato prima campionato e discretizzato, quindi risolto tramite l'algoritmo per MDP a stati discreti presentato in Sezione 3.4.4, infine nella terza il problema LQR è stato campionato, quindi risolto tramite il metodo risolutivo per MDP polinomiali Single Controller a stati continui presentato in Sezione 4.1.3. Entrambe le implementazioni sono state fatte utilizzando SDPA, già citato nella Sezione 5.1.2.

I risultati sperimentali raccolti sono le medie dei dati forniti da 10 prove di risoluzione del problema LQR descritto all'inizio di questa sezione con entrambi i metodi di risoluzione, e sono mostrati nelle Tabelle 5.4 e 5.5. Dall'analisi di queste tabelle si può notare come, nell'algoritmo con spazio degli stati discreto, all'aumentare del numero di stati entrambi gli errori, sulla funzione d'utilità e sulla policy ottima, aumentano: questo è dovuto al fatto che il problema LQR completo non ha un intervallo limite $[c, d]$ per lo spazio degli stati, mentre il problema campionato è vincolato all'interno dell'intervallo $[-4, 4]$, quindi aumentando gli stati si overfitta il problema e si aumenta l'errore di approssimazione.

Nel solver con spazio degli stati continui invece si può notare come, all'aumentare del numero di stati di partenza, l'errore sulla policy migliori

S^*	$\mathbb{E}(\|v(s) - \tilde{v}(s)\|_\infty)$	$\mathbb{E}(\|\pi(s) - \tilde{\pi}(s)\|_\infty)$	Tempo(Sec.)
8	1,033	0,114	0,263
16	0,778	0,193	0,633
24	0,97	0,233	1,268
32	1,023	0,245	2,354
40	1,314	0,286	3,858

Tabella 5.4: Analisi prestazionale dell'algoritmo per MDP polinomiali SC con spazio degli stati discreto.

$S^*/\tilde{S}/L_V$	$\mathbb{E}(\|v(s) - \tilde{v}(s)\|_\infty)$	$\mathbb{E}(\|\pi(s) - \tilde{\pi}(s)\|_\infty)$	Tempo(Sec.)
8/320/8	1,51	0,169	3,135
16/384/8	0,822	0,147	8,353
24/192/10	0,913	0,083	6,166
32/128/10	0,785	0,084	6,305
40/80/10	0,76	0,085	5,959

Tabella 5.5: Analisi prestazionale del solver per MDP polinomiali SC con spazio degli stati continuo.

fino ad arrivare ad una sorta di stabilità, mentre l'errore sull'utilità degli stati continui a migliorare: questo è dovuto al fatto che, nonostante gli stati di partenza siano campionati sempre nell'intervallo $[-4, 4]$, gli stati d'arrivo (ausiliari) sono campionati senza vincoli, quindi possono spaziare potenzialmente su tutto l'asse reale, riducendo così gli effetti negativi dell'overfit. Questo solver quindi fornisce sperimentalmente una soluzione approssimata con un errore inferiore a quello dell'algoritmo discreto, tuttavia la maggior precisione è pagata in maggior tempo di computazione, infatti l'inserimento nella formulazione di nuovi vincoli e variabili ausiliarie allunga i tempi di calcolo della soluzione.

Per quanto riguarda il solver con spazio degli stati continuo, presentiamo anche un'ulteriore analisi sperimentale per capire come migliora, in termini di accuratezza della soluzione, l'errore di approssimazione al variare del numero degli stati d'arrivo e della costante di Lipschitz L_V relativa alla funzione d'utilità. Questa ulteriore analisi è stata fatta per poter capire quanto influiscono sull'accuratezza della soluzione le nuove variabili e i nuovi vincoli

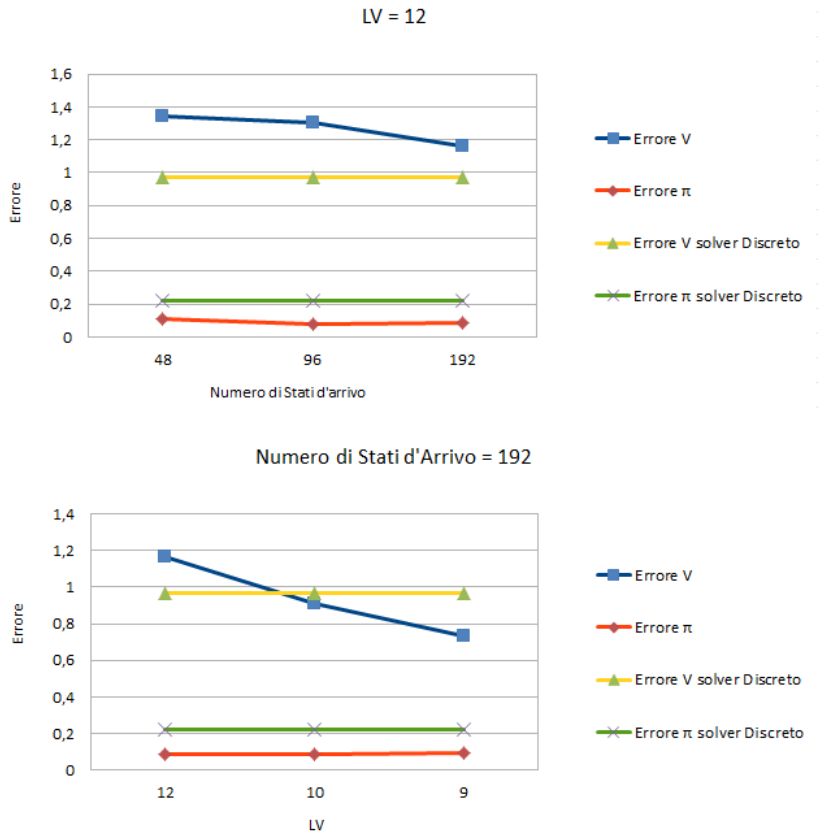

Figura 5.3: Analisi sperimentale dell'errore di approssimazione al variare del numero di stati d'arrivo e della costante di Lipschitz della funzione d'utilità, supponendo un numero di stati di partenza fissato a 24.

imposti sul problema rispetto all'algoritmo con spazio degli stati discreto, ed è mostrata in Figura 5.3. Qui viene scelto un numero di stati di partenza fissato a 24 per due motivi: il primo è dovuto al fatto che solitamente, per casi riguardanti problemi continui, si vogliono valutare le prestazioni del solver quando ha a disposizione pochi dati sicuri, che in questo caso sono rappresentati dagli stati di partenza del nostro MDP; il secondo è dovuto al fatto che l'effetto sulle prestazioni del solver in relazione al variare dei parametri già citati è più visibile quando abbiamo pochi stati di partenza. Il numero di stati ausiliari d'arrivo sarà variabile tra 48 e 192, numeri scelti per motivi di campionatura, invece il valore della costante di Lipschitz relativa alla funzione d'utilità sarà variabile tra 9 e 12, valori scelti perché si sà che per

questo particolare problema il valore reale della costante, cioè L_V^*, è intorno a 10.

Dall'analisi della Figura 5.3 si può dedurre che:

- l'errore sulla policy calcolata rispetto a quella ottima rimane costante sia al variare degli stati d'arrivo sia al variare della costante di Lipschitz;
- all'aumentare del numero di stati d'arrivo del gioco, il miglioramento dell'errore sui valori dell'utilità degli stati è poco significativo;
- al variare del valore della costante di Lipschitz, il miglioramento dell'errore sui valori dell'utilità degli stati è tanto significativo quanto più ci si avvicina al valore della vera costante di Lipschitz, cioè intorno a 10.

Si può concludere quindi che il solver per MDP polinomiali SC con spazio degli stati continuo ha buone performance di approssimazione se paragonato ad un equivalente basato su uno spazio degli stati discreto, questo a patto che si consideri di avere un tempo di computazione superiore e che si possa stimare il giusto intervallo su cui far variare i valori della costante di Lipschitz.

5.3 Valutazione dell'algoritmo di risoluzione di MDP polinomiali Switching Controller con spazio degli stati continuo

Iniziamo quest'ultima sezione presentando il modello di gioco su cui si è basato il nostro esperimento. Il modello in questione è il cosiddetto Linear Quadratic Regulator (LQR), cioè un regolatore caratterizzato da una funzione di reward lineare quadratica, già mostrato nel dettaglio nella Sezione 5.2.2.

Rispetto al caso precedente, però, la funzione di reward sarà definita in forma polinomiale come segue:

$$r(s, a) = -2s^2 + (a_1 - a_2)^2.$$

Lo spazio degli stati invece sarà diviso nel seguente modo: tutti gli stati campionati in posizione compresa nell'intervallo $[-2, 2]$ avranno transizioni governate dal secondo giocatore, mentre quelli campionati al di fuori dell'intervallo $[-2, 2]$ avranno transizioni governate dal primo giocatore, cioè

$$s \in S_1, \forall -2 \leq s \leq 2;$$

$$s \in S_2, \forall s \leq -2 \vee s \geq 2.$$

Diversamente da quanto fatto per il caso degli MDP Single Controller, per la valutazione dell'algoritmo di risoluzione di MDP polinomiali Switching Controller con spazio degli stati continuo non si presenterà un'analisi comparativa, ma solo una valutazione delle prestazioni generali dell'algoritmo rispetto al problema presentato. L'analisi comparativa infatti non è possibile in quanto, per il caso Switching Controller, non esiste una soluzione in forma chiusa del problema, quindi non si avrebbe un termine di confronto per valutare l'errore di approssimazione sulla soluzione trovata.

Procediamo quindi con la presentazione dei test effettuati per valutare le prestazioni dell'algoritmo. In questi test sono stati usati i seguenti valori dei parametri descrittivi del problema: $\beta = 0.9$, $\sigma = 0.5$, intervallo delle azioni e degli stati $[c, d] = [-4, 4]$, grado dei polinomi pari ad 8, valore di L_V fissato a 16, numero di stati ausiliari d'arrivo $\tilde{S}$ fissati a 128, numero di stati di partenza S^* e condizione di convergenza γ variabili.

La scelta di effettuare i test variando, tra i parametri descrittivi del problema, solo il numero di stati di partenza S^* è dovuta al fatto che la valutazione dell'algoritmo sarà basata solo su tempo di computazione e numero di iterazioni per raggiungere la convergenza, quindi dall'analisi dei dati presentati nella precedente sezione sul solver con Single Controller, di cui l'algoritmo con Switching Controller è una naturale estensione, sappiamo che il parametro che influirà maggiormente in questa valutazione sarà appunto quello relativo agli stati di partenza S^*.

L'implementazione dell'algoritmo è stata fatta utilizzando sempre SDPA, già presentato nella Sezione 5.1.2.

I risultati sperimentali raccolti sono le medie dei dati forniti da 10 prove di risoluzione tramite l'algoritmo del problema LQR prima descritto, e sono mostrati in Figura 5.4.

Dall'analisi dei dati in Figura 5.4 si può dedurre che:

- all'aumentare del numero di stati di partenza S^*, il numero di iterazioni dell'algoritmo per raggiungere la convergenza diminuisce linearmente. Questo risultato è motivato dal fatto che, aumentando la campionatura degli stati, le informazioni che l'algoritmo ha a disposizione sono maggiori, quindi ci si aspetta che quest'ultimo impieghi meno iterazioni per raggiungere la convergenza.

- all'aumentare del numero di stati di partenza S^*, il tempo di computazione dell'algoritmo per raggiungere la convergenza aumenta quasi esponenzialmente. Questo risultato è motivato dal fatto che, aumentando la campionatura degli stati, la dimensione totale del problema

aumenta in modo esponenziale, in termini di vincoli e variabili da gestire e valutare, quindi ci si aspetta che l'algoritmo ci metta molto più tempo per eseguire i singoli problemi SDP, e di conseguenza per raggiungere la convergenza.

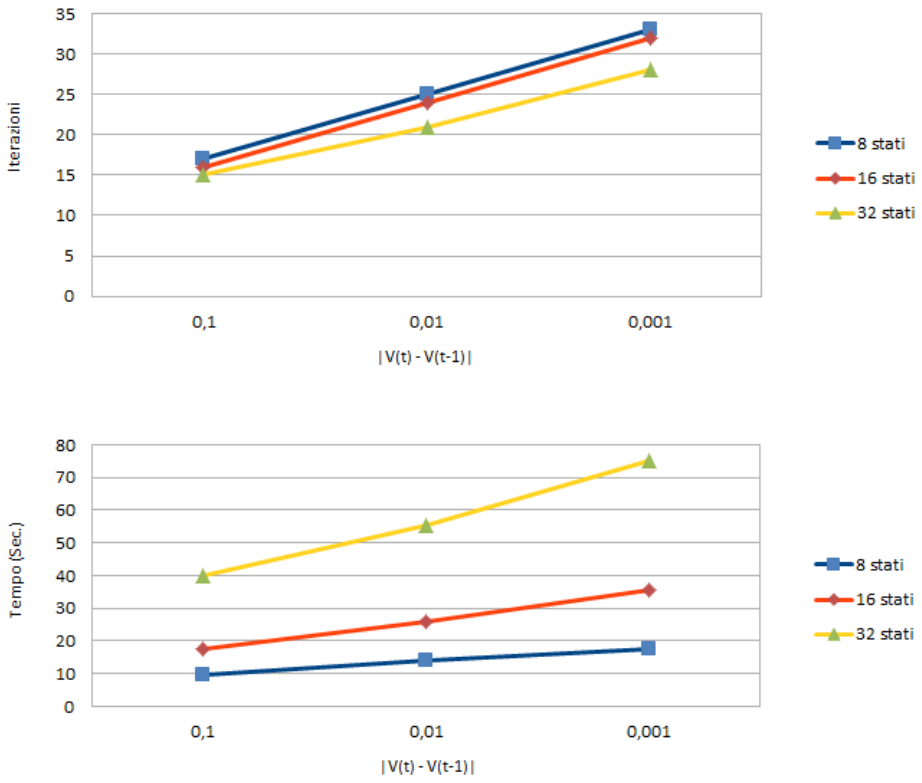

Figura 5.4: Valutazione sperimentale del tempo di computazione e del numero di iterazioni per la convergenza dell'algoritmo per MDP con Switching Controller al variare del numero di stati di partenza S^* e della condizione di termine γ.

Capitolo 6

Conclusioni e Sviluppi Futuri

6.1 Conclusioni

In questa tesi è stata studiata la classe dei giochi polinomiali stocastici a due giocatori a somma zero con spazio delle azioni continuo con Switching Controller, partendo dal caso più semplice in cui si ha uno spazio discreto degli stati, per poi passare al caso di uno spazio continuo degli stati; in particolare per quest'ultimo caso è stato presentato un algoritmo iterativo in grado di calcolare un vettore dei valori ed un profilo di strategie miste approssimati del gioco.

Inizialmente è stata presentata la classe dei giochi polinomiali con spazio delle azioni continuo e spazio degli stati discreto, partendo dai classici giochi in forma normale per poi passare ai giochi stocastici, in particolare alla sottoclasse dei giochi chiamata MDP, con proprietà di Single Controller e di Switching Controller. Relativamente a questi giochi, è stata discussa l'esistenza e l'unicità di una soluzione d'equilibrio, e sono state presentate le formulazioni ottimizzate per uno spazio delle azioni continuo supportato nell'intervallo $[c, d]$. Quindi sono stati presentati gli algoritmi, formulati in programmazione semidefinita positiva, in grado di risolvere questi giochi: in particolare, l'algoritmo basato su giochi Single Controller li risolve restituendo una soluzione d'equilibrio ottima, mentre l'algoritmo basato sui giochi Switching Controller li risolve restituendo una soluzione d'equilibrio ϵ-Nash. In particolare, relativamenete a quest'ultimo, sono state discusse le garanzie di convergenza dell'algoritmo e di ottimalità della soluzione ϵ-Nash restituita. Infine, è stato presentanta un'analisi dell'errore massimo di approssimazione nel caso si voglia ricondurre un gioco con funzioni in forma non polinomiale ad un equivalente gioco approssimato con funzioni in forma polinomiale, in

modo da poter usare le tecniche di risoluzione per giochi polinomiali finora presentate.

Successivamente, si è presentata la classe dei giochi stocastici polinomiali con spazio delle azioni e degli stati entrambi continui, in modo da estendere la trattazione precedentemente mostrata. Partendo da una semplice assunzione di regolarità sulle funzioni del gioco, rappresentata dall'assunzione di Lipschitz-continuità sulle funzioni di reward, di probabilità di transizione e di utilità, si è estesa la formulazione presentata per il caso degli MDP con spazio degli stati discreti a quella per il caso degli MDP con spazio degli stati continuo, cioè con un numero infinito e non numerabile di elementi. La nuova formulazione si è basata quindi sull'introduzione di nuovi vincoli rappresentanti l'assunzione di Lipschitz-continuità delle funzioni e sulla campionatura dello spazio degli stati, in modo da ottenere un problema equivalente all'originale ma con un numero di stati finito (campionato).

E' stato quindi presentanto un algoritmo, formulato sempre in programmazione semidefinita positiva, in grado di risolvere giochi stocastici polinomiali con Switching Controller e con spazio delle azioni e degli stati entrambi continui: grazie ai vincoli aggiuntivi rappresentanti l'assunzione di Lipschitz-continuità delle funzioni, questo algoritmo riesce a trovare un vettore dei valori approssimati del gioco ed un profilo di strategie miste approssimate dei due giocatori nel gioco, in quanto derivanti dall'approssimazione del gioco reale ad infiniti stati ad un equivalente gioco con stati campionati finiti.
Relativamente a questo algoritmo, è stata poi discussa l'analisi dell'errore teorico di approssimazione relativo ai valori della funzione d'utilità: sempre grazie all'assunzione di Lipschitz-continuità fatta sulle funzioni, è stato possibile garantire che quest'errore di approssimazione è limitato superiormente.

Alla fine è stata discussa la valutazione sperimentale dei diversi algoritmi presentati in questa tesi, presentando per ognuno di questi il relativo generatore automatico di giochi, basato sempre sulla risoluzione di un problema formulato in programmazione semidefinita positiva.

La valutazione relativa all'algoritmo di risoluzione di MDP polinomiali con Switching Controller, spazio degli stati discreto e spazio delle azioni continuo ha permesso di dimostrare come la nuova implementazione tramite SDPA ha permesso di azzerare il tempo di overhead, che si aveva con la precedente implementazione in YALMIP, e di fare un'analisi di scalabilità dell'algoritmo al variare dei parametri descrittivi del problema, dimostrando come i tempi di computazione dell'algoritmo stesso sono dipedenti dai parametri d'ingresso, in particolare dal grado dei polinomi delle funzioni, in relazione quasi esponenziale. Sempre relativamente a questo algoritmo, è stato dimostrato come l'ottimizzazione dei vincoli effettuata in questa tesi rispetto a lavori precedenti permette un risparmio di più del 12% del tempo

di computazione per problemi di dimensione elevate.

La valutazione relativa all'algoritmo di risoluzione di MDP polinomiali con Single Controller, spazio degli stati e spazio delle azioni entrambi continui ha poi permesso di dimostrare come quest'ultimo, grazie all'introduzione dell'assunzione di regolarità sulle funzioni descrittive del gioco, abbia prestazioni migliori, in termini di accuratezza della soluzione restituita, rispetto all'algoritmo con spazio degli stati discreto, soprattutto al crescere del numero di stati campionati per descrivere il gioco approssimato; questa maggiore accuratezza è però pagata in termini di tempo di computazione, che risulta più che raddoppiato se paragonato all'algoritmo con spazio degli stati discreto. Questa valutazione infine ha permesso anche di dimostrare come una buona scelta del valore della costante di Lipschitz $L_{\tilde{V}}$ relativa alla funzione d'utilità approssimata sia fondamentale per accuratezza della soluzione restituita dall'algoritmo.

Infine, La valutazione relativa all'algoritmo di risoluzione di MDP polinomiali con Switching Controller, spazio degli stati e spazio delle azioni entrambi continui ha permesso di dimostrare come le prestazioni di quest'ultimo siano fortemente dipendenti dal numero di stati di partenza S^* campionati per descrivere il problema, in quanto influenzano in rapporto quasi esponenziale i tempi di computazione dell'algoritmo.

6.2 Sviluppi Futuri

Una possibile direzione di ricerca potrebbe essere quella di estendere il numero di giocatori in gioco ad un numero superiore a 2, cosa che però comporterebbe il cambiamento delle tecniche basate sulla ricerca di un equilibrio minmax presentate in questa tesi a favore di tecniche di risoluzione più generiche basate sulla ricerca di equilibri di Nash, di cui gli equilibri minmax sono una versione particolare nel caso di giochi a due giocatori a somma zero.

Un'altra possibile prospettiva di ricerca futura potrebbe essere quella di estendere il controllo delle probabilità di transizione in ogni stato del gioco dal caso Switching Controller al caso Double Controller, dove entrambi i giocatori influenzerebbero le transizioni in ogni stato, quindi la funzione polinomiale diventerebbe multivariata, richiedendo l'applicazione di tecniche più complesse basate sulla risoluzione di gerarchie di problemi SDP.

Bibliografia

[1] E. Altman, *Constrained markov decision processes*, In Chapman and Hall, Boca Raton, Florida, 1999.

[2] Martin L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, 1994.

[3] J. A. Filar and K. Vrieze, *Competitive Markov decision processes*, Springer, New York, 1997.

[4] Samuel Karlin Dresher, Melvin and Lloyd S. Shapley. *Polynomial Games. In Contributions to the Theory of Games*, pages 161–180, 1950.

[5] Pablo A Parrilo. Polynomial Games and Sum of Squares Optimization. *Decision and Control, 2006 45th IEEE Conference on*, pages 2855 – 2860, 2006.

[6] Guido Bonomi, *Giochi stocastici polinomiali a somma zero con Switching Control*, Tesi di Laurea Magistrale in Ingegneria Informatica, Politecnico di Milano, Anno Accademico 2010-2011.

[7] R. Meziat, *The Method of Moments in Global Optimization*, Journal of Mathematical Sciences, Vol. 116, No. 3, pag 3303 - 3324. 2003.

[8] Bruce Schmeiser and Luc Devroye, *Non-Uniform Random Variate Generation*, Journal of the American Statistical Association, 83(403):906, September 1988.

[9] J.A. Shohat and J.D. Tamarkin. The Problem of Moments. *American Mathematical Society Mathematical surveys*, 2, 1943.

[10] Parikshit Shah and Pablo a. Parrilo. Polynomial Stochastic Games via Sum of Squares Optimization. *2007 46th IEEE Conference on Decision and Control*, pages 745–750, 2007.

[11] L S Shapley. Stochastic Games. emphProceedings of the National Academy of Sciences of the United States of America, 39(No 10):1095–1100, October 1953.

[12] J. a. Filar. Ordered Field Property for Stochastic Games when the Player who Controls Transitions changes from State to State. *Journal of Optimization Theory and Applications*, 34(No 4):503–515, August 1981.

[13] J. A. Filar, *Algorithms for Solving some Undiscounted Stochastic Games*, Ph.D. Dissertation, Univeristy of Illinois, Chicago, 1979.

[14] Nagarajan Krishnamurthy, T. Parthasarathy, and G. Ravindran. Orderfield Property of Mixtures of Stochastic Games. *Sankhya A*, 72(No 1):246–275, June 2010.

[15] J. A. Filar O. J. Vrieze, S. H. Tijs, T. E. S. Raghavan. A Finite Algorithm for the Switching Control Stochastic Game. *OR Spektrum 5*, pages 15–24, 1983.

[16] F. Thuijsman and T.E.S Raghavan. Stochastic Games with Switching Control or ARAT Structure. *Technical Report M94-06*, University of Limburg, Maastricht, The Netherlands, 1997.

[17] O.J. Vrieze. Stochastic Games with Finite State and Action Spaces. *Centrum voor Wiskunde en Informatica*, 1987.

[18] T. E. S. Raghavan S. R. Mohan. An Algorithm for Discounted Switching Control Stochastic Games. *OR Spektrum 9*, pages 41–45, 1987.

[19] Guido Bonomi and Nicola Gatti and Fabio Panozzo and Marcello Restelli, *Computing equilibria in two–player zero–sum continuous stochastic games with switching controller*, Politecnico di Milano, 2012.

[20] Remez, Y. L. On a method of Chebyshev type approximation of functions. *Ukrain Ann.*, 1935.

[21] J. Lofberg. YALMIP : a toolbox for modeling and optimization in MATLAB. *2004 IEEE International Symposium on Computer Aided Control Systems Design*, (4):284 – 289, 2004.

[22] K. Fujisawa, K. Nakata, M. Yamashita, and M. Fukuda. SDPA project: Solving large-scale semidefinite programs. *Journal of Operations Research Society of Japan 50 (2007)*, pag. 278–298, 2007.

[23] E. Anderson, Z. Bai, C. Bischof, L. S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. LAPACK Users' Guide, Third Edition. *Society for Industrial and Applied Mathematics*, Philadelphia, PA, 1999.

[24] Jan Peters, *Policy Gradient Methods for Control Applications*, 2003.

[25] Yoav Shoham and Kevin Leyton-brown, *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*, 2010.

[26] Roberto Lucchetti, *A Primer in Game Theory*, Prima Ed. Ottobre 2011.

[27] S. Russell and P. Norvig, *Artificial Intelligence - A Modern Approach*, 3rd ed. Prentice-Hall, 2010.

[28] Constantinos Daskalakis, Aranyak Mehta and Christos Papadimitriou, *A Note on Approximate Nash Equilibria*, University of California, Berkeley, USA, 2006.

[29] S Raghavan. Finite-step Algorithms for Single-Controller and Perfect Information Stochastic Games. *Introduction to Games and Economic Theory Selected contribution in Honor of Robert J. Aumann*, pages 227–251, 1995.

[30] Lihong Li and Michael L. Littman, *Lazy Approximation: A New Approach for Solving Continuous Finite-Horizon MDPs*, Department of Computer Science, Rutgers University, June , 2005.

[31] Janusz Marecki, Zvi Topol and Milind Tambe, *A Fast Analytical Algorithm for MDPs with Continuous State Spaces*, Computer Science Department, University of Southern California, Los Angeles, 2006.

[32] Milos Hauskrecht and Branislav Kveton *Linear Program Approximations for Factored Continuous-State Markov Decision Processes*, Department of Computer Science and Intelligent Systems Program, University of Pittsburgh, 2003.

[33] Jason Pazis and Ronald Parr, *Non-parametric Approximate Linear Programming for MDPs*, Department of Computer Science, Duke University, 2011.

[34] Marek Petrik, Gavin Taylor†, Ron Parr† and Shlomo Zilberstein, *Feature Selection Using Regularization in Approximate Linear Programs for Markov Decision Processes*, Department of Computer Science, University of Massachusetts, Department of Computer Science, Duke University, 2010.