UNIVERSITÀ DELLA CALABRIA

UNIVERSITA' DELLA CALABRIA

Dipartimento di Ingegneria Informatica, Modellistica, Elettronica e Sistemistica

Dottorato di Ricerca in

INFORMATION AND COMMUNICATION ENGINEERING FOR PERVASIVE INTELLIGENT ENVIRONMENTS

CICLO

XXIX

Problemi di Allocazione in Giochi Cooperativi: Approssimazioni e Casi Trattabili per il Calcolo del Valore di Shapley

Settore Scientifico Disciplinare ING-INF/05 SISTEMI DI ELABORAZIONE DELLE INFORMAZIONI

Coordinatore: Ch.mo Prof. Crupi Felice

Firma

Supervisore/Tutor: Ch.mo Prof. Scarcello Francesco

Firma

Dottorando: Dott. Mendicelli Angelo

Firma

UNIVERSITÀ DEGLI STUDI DELLA CALABRIA
UNICAL

SCUOLA DI DOTTORATO

INFORMATION AND COMMUNICATION ENGINEERING FOR PERVASIVE INTELLIGENT ENVIRONMENTS – XXIX ciclo

Tesi di Dottorato

Problemi di Allocazione in Giochi Cooperativi: Approssimazioni e Casi Trattabili per il Calcolo del Valore di Shapley

Mendicelli Angelo

Coordinatore del corso di dottorato
prof. Felice Crupi

Relatore:
prof. Francesco Scarcello

Luglio 2017

Sommario

Capitolo 1

Introduzione

La Teoria dei Giochi permette di modellare ed analizzare secondo un rigoroso formalismo matematico le interazioni tra agenti razionali, ognuno dei quali persegue i propri obiettivi. In letteratura esistono due macro-categorie di giochi, chiamati giochi cooperativi e giochi non-cooperativi, in funzione della propensione degli agenti a collaborare o meno tra loro per raggiungere un qualche obiettivo generale, ad esempio massimizzare il bene comune (*social welfare*).

In questo lavoro di tesi ci concentreremo sui giochi cooperativi, in particolare sui giochi di coalizione, nei quali la ricchezza (*worth*) a cui possono ambire i vari agenti è determinata dalle varie coalizioni a cui possono partecipare. Formalmente un gioco di coalizione è una coppia $\mathsf{G} = \langle N, v \rangle$, in cui N è un insieme di agenti e v è una funzione che associa un valore o un insieme di possibili configurazioni a ciascun insieme $C \subseteq N$ di agenti (o giocatori), chiamato coalizione. In particolare in questo lavoro saranno considerati i cosiddetti giochi a utilità trasferibile, nei quali la funzione v associa ad ogni coalizione un valore numerico, che possiamo interpretare per semplicità come una quantità di denaro e che può appunto essere liberamente distribuita tra gli agenti. Nei giochi ad utilità non-trasferibile, invece, la funzione descrive solo un insieme discreto di possibili distribuzioni della ricchezza disponibile.

Il problema fondamentale in questo tipo di giochi è individuare una distribuzione della ricchezza agli agenti che soddisfi una serie di proprietà desiderabili. Ad esempio, si richiede tipicamente che la ricchezza disponibile $v(N)$ per l'insieme di tutti i giocatori N, chiamata anche *grand coalition*, sia distribuita in modo completo (*budget balance*). Inoltre, per motivi di stabilità, si richiede usualmente che ad ogni gruppo di agenti C sia riconosciuta almeno la quantità $v(C)$. In particolare ciò implica la cosiddetta *individual rationality*, che stabilisce che ogni agente abbia almeno la ricchezza a cui può ambire da solo, senza collaborare con altri.

Vari concetti di soluzione "fair" sono stati descritti in letteratura e, tra questi, uno dei principali è il valore di Shapley [33]. In base a tale modalità di distribuzione, la ricchezza di ogni agente i è calcolata in base al proprio contributo ad ogni possibile coalizione di

agenti C. Più precisamente occorre considerare il cosiddetto contributo marginale di i, cioè la differenza tra ciò che C può ottenere con i e ciò che può ottenere senza di lei/lui. Formalmente, il valore di Shapley per i si ottiene considerando la seguente media pesata dei vari contributi marginali di i:

$$\phi_i(\mathsf{G}) = \sum_{C \subseteq N \setminus \{i\}} \frac{|C|!(n - |C| - 1)!}{n!} \Big(v(C \cup \{i\}) - v(C) \Big).$$

Vi sono numerose tipologie di giochi di coalizione, in base alle diverse possibili classi di funzioni di worth v. Il presente lavoro di tesi riguarda in particolare i cosiddetti *giochi di allocazione*, in cui occorre allocare un insieme di beni indivisibili agli agenti in modo da massimizzare il *social welfare*. Ad ogni agente può essere assegnato un bene (o anche più di uno, in alcune varianti di tali giochi), e a ciascun assegnamento bene-agente corrisponde un certo beneficio per la comunità. Pertanto la worth $v(C)$ associata a ciascuna coalizione C è determinata dalla migliore possibile allocazione di beni agli agenti in C e la ricchezza totale $v(N)$ è determinata dall'allocazione ottima (globale) dei beni a tutti gli agenti. Nonostante i beni siano indivisibili, è possibile ottenere delle suddivisioni "fair" , essendo il gioco ad utilità trasferibile. Ad esempio può essere conveniente associare ad un certo agente un bene scadente (per massimizzare l'utilità complessiva) ma poi assegnargli una compensazione monetaria che tenga conto del suo contributo reale alla comunità. Questo tipo di giochi emerge naturalmente in diversi domini applicativi, che spaziano dall'allocazione di beni reali (es. appartamenti) allo scheduling (cooperativo) e all'allocazione di processi, fino ad arrivare ai protocolli per la comunicazione nelle reti wireless (si veda, ad esempio, [61, 62, 63] e le referenze contenute in questi lavori).

1.1 Contributi

Calcolare il valore di Shapley è un problema difficile in generale, precisamente *#P-Hard*, e rimane difficile se consideriamo i soli giochi di allocazione ed anche se i beni possono assumere solo due possibili valori. È quindi evidente che calcolare tale valore con un approccio *brute-force* non sia fattibile, a meno che gli agenti in gioco non siano pochissimi. In questo lavoro di tesi siamo interessati invece ad applicazioni reali dei problemi di allocazione in cui sono coinvolti molti agenti e, pertanto, sono stati esplorati due possibili approcci algoritmici:

1. schemi di approssimazione polinomiale, eventualmente randomizzati;
2. individuazione di classi di giochi trattabili, in base alla struttura delle interazioni tra gli agenti.

Purtroppo, nonostante la presenza in letteratura di algoritmi per l'approssimazione (randomizzata) del valore di Shapley (FPRAS) [25], in casi reali con molti agenti il tempo richiesto per ottenere una soluzione di buona qualità (ancorché in valore atteso e non

garantito) è spesso inaccettabile. Un primo contributo della tesi è stato quindi quello di individuare proprietà di modularità e possibili semplificazioni del problema che ci permettessero di scomporre una data istanza in sottoproblemi di dimensioni inferiori e quindi più facilmente affrontabili.

Un secondo importante contributo riguarda la definizione di un limite inferiore ed un limite superiore per il valore di Shapley che sia calcolabile in modo efficiente, almeno per quegli agenti che hanno poche situazioni di conflitto con altri agenti. La disponibilità di tali *bound* ci permette di avere dei range precisi (e non approssimati) per la qualità delle soluzioni ottenute con gli algoritmi proposti e, soprattutto, ci permette di stimare in modo più accurato il numero di campioni da considerare negli algoritmi approssimati. Inoltre in molti casi capita che tali limiti coincidano e che quindi abbiamo la garanzia di conoscere il valore di Shapley esatto per un certo agente.

Tutti gli algoritmi proposti sono stati implementati e testati su un problema di allocazione reale che modella la Valutazione della Qualità della Ricerca in Italia (VQR), utilizzando come caso di studio la valutazione presso l'Università di Roma "La Sapienza" che coinvolge migliaia di agenti e di beni, nella fattispecie i ricercatori/professori afferenti a tale università ed i loro prodotti di ricerca.

Il lavoro di tesi si è infine concentrato sull'individuazione di classi trattabili del problema, in particolare le classi di giochi in cui il grafo agenti-beni ha un basso grado di ciclicità, più precisamente una bassa *treewidth* [32], e a ciascun agente possa essere allocato al più un prodotto. In quest'ambito, sono state adattate alcune proprietà interessanti dei giochi di allocazione descritte recentemente in letteratura ed è stato proposto un nuovo algoritmo per il calcolo del valore di Shapley. Tale algoritmo, basato su formule della logica monadica del secondo ordine (MSO), è in grado di calcolare le soluzioni desiderate in tempo polinomiale, se il grafo agenti-beni ha appunto bassa treewidth. Questo risultato di trattabilità generalizza il recente risultato descritto in [14], che era invece basato sulle proprietà strutturali del grafo agenti-agenti, i cui rami rappresentano conflitti di interesse tra gli agenti. L'implementazione è basata sulla definizione di un problema di conteggio del numero di soluzioni di una data cardinalità di una formula MSO. Per valutare tali formule è stato usato il framework Sequoia [51], la cui attuale implementazione permette di contare le soluzioni di una formula MSO con variabili libere (cioè tutte le istanziazioni distinte di tali variabili libere) in tempo polinomiale su classi di grafi di treewidth limitata.

Questo algoritmo, insieme alle altre semplificazioni descritte nel presente lavoro, è stato in grado di risolvere in modo esatto l'istanza completa del problema di allocazione VQR dell'Università "La Sapienza", ristretto al caso di un singolo bene allocabile a ciascun agente.

1.2 Organizzazione

Il lavoro di tesi è organizzato come segue:

- nel capitolo 2 si introducono i concetti fondamentali della teoria dei giochi cooperativi ed alcuni concetti di soluzione;
- nel capitolo 3 si analizzano le proprietà dei giochi di allocazione e l'applicazione al caso di studio della valutazione della qualità della ricerca italiana (VQR);
- nel capitolo 4 si individuano alcune proprietà dei giochi di allocazione che permettono di semplificare il calcolo del valore di Shapley; si propone inoltre un algoritmo per il calcolo di un lower bound e di un upper bound del valore di Shapley;
- nel capitolo 5 si descrive come approssimare il valore di Shapley e come valutare il numero di campioni di coalizioni da considerare, in modo che il risultato atteso rientri in un bound prefissato (con alta probabilità);
- nel capitolo 6 si riportano i risultati sperimentali per gli algoritmi proposti condotti su un dataset reale, con migliaia di agenti e beni, relativo al problema della valutazione della ricerca (VQR) per l'Università di Roma "La Sapienza";
- nel capitolo 7 si presenta il risultato di trattabilità, basato sull'uso di formule MSO, per i giochi di allocazione i cui grafi agenti-beni abbiano treewidth limitata;
- nel capitolo 8 si descrive l'implementazione del software per il calcolo del valore di Shapley basato sulle formule MSO e se ne presentano i risultati sperimentali per l'istanza VQR di Roma, ristretta al caso del singolo bene allocabile a ciascun autore.
- nel Capitolo 9 si riportano le conclusioni del lavoro.

Capitolo 2

Teoria dei giochi

2.1 Giochi cooperativi

La teoria dei giochi è uno degli argomenti più studiati nei settori dell'economia e della matematica. Essa si occupa di modellare il comportamento dei partecipanti al gioco (agenti) all'interno di sistemi intelligenti, in cui ciascun agente agisce al fine di massimizzare il proprio profitto; tale fine può essere raggiunto attraverso l'applicazione di strategie di tipo *cooperativo* o *non cooperativo*. Nel primo caso gli agenti si alleano al fine di perseguire un obiettivo comune, mentre nel secondo caso essi decidono di non cooperare, perseguendo il loro obiettivo individualmente. La teoria dei giochi non cooperativi analizza dunque questi ultimi scenari, che vengono scelti per ragioni che possono sostanzialmente riassumersi nel non trarre vantaggio dalla cooperazione. È opportuno osservare che il termine "non cooperativo" può generare confusione. La teoria dei giochi non cooperativa non si applica solo in situazioni in cui gli interessi degli agenti collidono. Equivalentemente, la teoria dei giochi cooperativi, non studia soltanto le situazioni in cui gli interesso degli agenti sono comuni.

Nell'ambito dei giochi non cooperativi, un ruolo molto importante è assunto dal concetto di equilibrio di Nash. Questo concetto è sicuramente il metodo più diffuso per predire il risultato di una interazione strategica nel campo delle scienze sociali. Più precisamente, tale concetto modella il comportamento razionale dei giocatori che, avendo a disposizione diverse strategie a cui sono associate delle funzioni di utilità (che stabiliscono i payoff che ogni giocatore riceve), raggiungono una condizione di stabilità in cui ognuno di loro ha selezionato una strategia ottimale (rispetto all'azione di ogni altro giocatore) e di conseguenza nessuno questi giocatori è incentivato ad abbandonarla. Si noti che decidere se un gioco ammette un equilibrio di Nash e calcolarlo efficientemente è un problema di grande interesse per la comunità scientifica. Nel caso di giochi in cui sono ammesse strategie miste (cioè regolate da distribuzioni di probabilità) l'esistenza di un equilibrio è garantito dal famoso teorema di Nash e, recentemente la complessità

computazione del calcolo di un equilibrio di Nash è stata completamente caratterizzata.

Tuttavia, questo lavoro di tesi è focalizzato sullo studio dei giochi cooperativi, ovvero quella tipologia di giochi in cui gli agenti possono ottenere massimizzare il loro benessere collaborando tra di loro rispetto all'azione isolata. In molte applicazioni del mondo reale, non è facile per un agente agire in modo ottimale. Ciò è dovuto al fatto che in un certa situazione l'agente non agisce in modo isolato ma deve confrontarsi in generale con altri agenti i quali cercano di perseguire i propri obiettivi. Nel momento in cui tali obiettivi sono comuni, allora la teoria dei giochi cooperativi è in grado di fornire tutti gli ingredienti per lo studio dei meccanismi di cooperazione, che permettono ad ogni singolo agente di ottenere benefici maggiori dalla cooperazione in gruppi di agenti rispetto a quello che riuscirebbe ad ottenere il singolo.

In questo capitolo procederemo come segue. Prima di tutto introdurremo la classe dei giochi di coalizione e discuteremo alcune importanti proprietà nell'ambito di tale classe di giochi. Successivamente discuteremo la nozione di concetto di soluzione; discuteremo i più importanti concetti di soluzione usati nella teoria dei giochi cooperativa e studieremo in particolare il concetto di valore di Shapley.

2.2 Giochi di coalizione

Una importante classe di giochi cooperativi è quella dei giochi di coalizione (coalition games). Un gioco di coalizione è un gioco in cui gruppi di giocatori possono allearsi e coordinare le loro azioni garantendosi maggiori vantaggi rispetto all'azione isolata. In effetti, la formazione di alleanze o coalizione è presente in moltissimi scenari del mondo reale, in cui vi sono diversi individui che decidono di collaborare allo scopo di perseguire un obiettivo comune.

Esempio 2.1. *Come esempio di gioco di coalizione, si consideri il gioco di maggioranza. In questo gioco, abbiamo un insieme N di giocatori e per ogni coalizione $C \subseteq N$, C è detta coalizione* vincente *(cioè a dire, un gruppo di giocatori che possono allearsi per "vincere"), se tutti i membri di tale coalizione sono sufficienti per trasformare una proposta di legge in legge se tutti decidono di farla passare. Le coalizioni che decidono di non far passare la legge sono dette coalizioni* perdenti. *Per una coalizione* vincente C *abbiamo $v(C) = 1$, viceversa $v(C) = 0$. Di conseguenza è facile vedere che ogni coalizione composta da un solo giocatore è* perdente *mentre la Grand-Coalition che comprende tutti i giocatori è* vincente.

I giochi di coalizioni possono essere formalizzati come tuple $G = (N, v)$ dove N è l'insieme dei giocatori e v è una funzione detta **funzione caratteristica** che associa ad ogni coalizione $S \subseteq N$ un valore reale $v(S)$ che codifica l'utilità che i giocatori in S ottengono dalla loro collaborazione. Si noti che possono esistere $2^n - 1$ sottoinsiemi $S \subseteq N$ tali che S possa essere visto ai fini del gioco come un unico giocatore. Un

aspetto fondamentale dei giochi di coalizione riguarda il valore $v(S)$ che una coalizione S assume all'interno di un gioco. Nello specifico, le diverse formulazioni della funzione caratteristica v determinano differenti tipologie di gioco.

Giochi con utilità trasferibile. Nella sua forma più comune o *forma caratteristica* il valore v di una coalizione S dipende unicamente dai membri della coalizione, e possono essere di fatto trascurati eventuali altri accordi o legami che sussistono tra i restanti $N \setminus S$ membri. Il concetto di forma caratteristica è stato introdotto da von Neuman e Morgenstern [1], insieme ad una nuova categoria dei giochi conosciuti come giochi con *Utilità Trasferibile (TU)*. Nell'ambito di questi giochi, il valore della coalizione (ovvero i guadagni ottenuti dalla collaborazione dei membri della coalizione) può essere diviso liberamente tra i suoi membri. Diversamente, nei giochi senza utilità trasferibile (NTU) le coalizioni possono dividere i loro guadagni solo secondo alcune particolari regole del gioco dette conseguenze [52]. In effetti, il problema cruciale è quello di trovare delle regole di distribuzione (del valore associato con la *Grande-Coalizione* N) desiderabili, che possano essere percepite come giuste e stabili. Su questo punto torneremo più in dettaglio nel seguito di questo capitolo. In un gioco di coalizione $G = (N, v)$ la funzione v è detta super-modulare (risp., sub-modulare) se vale che $v(S \cup T) + v(S \cap T) \geq v(S) + v(T)$, (risp., $v(S \cup T) + v(S \cap T) \leq v(S) + v(T)$) per ogni coppia di coalizioni $S, T \subseteq N$.

Nella teoria dei giochi di coalizione, una proprietà interessante che si assume spesso per la funzione v è la superadditività. Formalmente v è superadditiva se $v(S \cup T) \geq v(S) + v(T), \forall S, T \subset N, S \cap T = \emptyset$. In questi casi, si parla di *gioco superadditivo*. In un gioco superadditivo, la cooperazione porta ad ottenere utilità più alte per ogni giocatore della coalizione rispetto all'azione individuale. Si noti che ciò è vero solo nel caso in cui le coalizioni lavorino senza che queste interferiscano tra di loro. È inoltre utile osservare che, ogni gioco può essere trasformato nella sua forma superadditiva. Intuitivamente, nel caso peggiore i giocatori possono mettersi d'accordo per non cooperare al fine di ottenere dei payoff più elevati. Un'importante implicazione della superadditività riguarda il meccanismo della formazione delle coalizioni. Infatti, in un gioco superadditivo il valore della *Grande-Coalizione* è sempre maggiore rispetto al valore di qualunque altra coalizione e, di conseguenza, la *Grande-Coalizione* si formerà sempre. Si consideri invece una situazione in cui le coalizioni non possono mai interferire tra di loro, sia in modo positivo che in modo negativo. In questo caso, otteniamo un'altra forma di gioco chiamato *gioco additivo*. Formalmente un gioco si dice additivo se $v(S \cup T) = v(S) + v(T), \forall S, T \subset N, S \cap T = \emptyset$. Nei prossimi capitoli, tutte le volte che faremo riferimento alla funzione v, assumeremo che v sia superadditiva. Concludiamo questa sezione riportando per completezza qualche altra sottoclasse rilevante di giochi di coalizione.

Definizione 2.1 (Gioco a somma costante)**.** *Un gioco* $G = (N, v)$ *è* a somma costante *se,* $\forall S \subset N, v(S) + v(N \setminus S) = v(N)$.

Tutti i giochi additivi sono necessariamente a somma costante, ma non è vero il viceversa. I giochi a somma costante più studiati sono quelli a somma zero.

Definizione 2.2 (Gioco convesso)**.** *Un gioco* $G = (N, v)$ *è* convesso *se,* $\forall S, T \subset N, v(S \cup T) \geq v(S) + v(T) - v(S \cap T)$.

Questi giochi sono una sottoclasse dei giochi superadditivi. Si noti che la proprietà di "convessità" è una condizione più forte della superadditività, tuttavia le situazioni del mondo reale in cui si manifestano questi giochi sono poche.

Definizione 2.3 (Gioco semplice)**.** *Un gioco* $G = (N, v)$ *è* semplice *se,* $\forall S \subset N, v(S) \in \{0, 1\}$.

Questa classe di giochi è utile per modellare i giochi di maggioranza discussi nell'Esempio 2.1 . In genere, vi si aggiunge il requisito che se una colazione vince, allora tutti i sovrainsiemi della coalizione sono coalizioni vincenti, quindi, se $v(S) = 1, \forall T \supset S, v(T) = 1$. Questa condizione non implica la superadditività. Quando i giochi semplici sono anche a somma costante vengono chiamati *propriamente semplici*. In questi giochi, se S è una coalizione vincente, allora $N \setminus S$ è una coalizione perdente.

Giochi senza utilità trasferibile. Esistono molti scenari in cui il valore di una coalizione non può essere rappresentato semplicemente da un numero reale, o in cui in generale esistono delle rigide restrizioni sulla distribuzione del valore di una coalizione tale per cui è impossibile modellare un gioco G con una funzione caratteristica di tipo (TU). Tali giochi sono conosciuti ed ampiamente descritti da Aumann e Peleg in [4]. Si tratta dei giochi di coalizione senza utilità trasferibile (NTU). In un gioco (NTU), il *payoff* assegnato a ogni giocatore in relazione ad una coalizione S dipende dalle diverse azioni effettuate da tutti i giocatori che appartengono a S. Il valore $v(S)$ di una data coalizione S, è una funzione il cui codominio è un insieme di vettori di *payoff*, $v(S \subseteq R^{|S|})$, dove ogni elemento x_i del vettore $x \in v(S)$ rappresenta un particolare *payoff* che il giocatore $i \in S$ può ottenere all'interno della coalizione S se assume una specifica strategia. Possiamo quindi affermare che la classe dei giochi *TU* è un caso particolare della classe dei giochi (NTU).

Dopo questa breve descrizione formale, ricordiamo che in un gioco di coalizione, non siamo interessati nel determinare il modo in cui gli agenti interagiscono all'interno di una coalizione, cioè a dire, non siamo interessati ai dettagli degli accordi, ma piuttosto siamo interessati ai payoffs della coalizione. Da questo punto di vista, in questo lavoro di tesi siamo interessati a capire (i) tra tutte le possibili coalizioni quale coalizione si formerà e (ii) come tale coalizione dovrebbe distribuire il suo valore tra i suoi membri. Spesso la risposta al primo problema è la *Grand-Coalition*. Tuttavia, in molte situazioni tale risposta dipende anche dalla scelta effettuata tenendo in considerazione il secondo problema.

2.3 Concetti di soluzione

Abbiamo visto che una questione centrale nella teoria dei giochi cooperativi consiste nel determinare come dividere il guadagno della grande coalizione tra i suoi agenti. Ci si concentra sulla grande coalizione perché molti giochi sono superadditivi e, come già detto, in questi giochi è proprio la grande coalizione ad avere il guadagno più alto e ci si può aspettare che venga formata. Inoltre, gli agenti non potrebbero avere altra scelta che formare la grande coalizione, come nel caso dei progetti pubblici, nei quali la legge richiede che tutti i partecipanti siano coinvolti. Il problema fondamentale per un gioco di coalizione $G = (N, v)$ diventa dunque quello di trovare un metodo desiderabile di divisione del guadagno di una coalizione tra i propri membri. Questi metodi di distribuzione prendono il nome di *concetti di soluzione* e sono vettori di payoffs (pagamenti) della forma $(x_1, ..., x_{|N|}) \in \mathbb{R}^{|N|}$ dove $x_i + \cdots + x_{|N|}$ è uguale al valore associato con l'intera coalizione N di agenti. Un vettore di payoff x è una *imputazione* se sono vere le seguenti condizioni: la condizione di *razionalità individuale* i.e., per tutti i giocatori $i \in N$ si ha che $x_i \geq v(i)$ e, la condizione di *efficienza* i.e., $\sum_{i \in N} x_i = v(N)$. Efficienza in questo contesto significa che l'intero valore della coalizione è diviso tra i giocatori, mentre la condizione di razionalità individuale indica che ad ogni giocatore è garantito almeno il valore che avrebbe potuto ottenere da solo. In letteratura sono stati studiati diversi concetti di soluzioni e, tra tutti, quelli più conosciuti e largamente accettati sono il *nucleo*, il *kernel*, il *nucleolo* e il *valore di Shapley*. Nel resto di questa sezione richiamiamo le nozioni di *nucleo*, *kernel*, *nucleolo*, *valore di Shapley* e *Indice di Banzhaf* e discuteremo brevemente delle loro proprietà.

2.3.1 Il nucleo

Il nucleo [64, 8] (o core) è probabilmente insieme al valore di Shapley il concetto di soluzione più importante nel campo della teoria dei giochi di coalizione. Il nucleo è interessante perchè fornisce indicazioni in merito alla stabilità della *Grande-Coalizione*.

Il nucleo di un gioco di coalizione $G = (N, v)$ è l'insieme delle allocazioni ammissibili che disincentivano i propri membri ad abbandonare la *Grande-Coalizione* per formare coalizioni più piccole. Tale definizione è strettamente correlata alla proprietà di super-additività, in quanto in sua presenza i giocatori sono incentivati a formare coalizioni sempre più grandi fino a formare una coalizione che li contiene tutti. Di seguito si specifica la struttura del nucleo nelle diverse tipologie di giochi:

Dato un gioco $G = (N, v)$ in forma (TU), il nucleo di G è un vettore in cui i singoli elementi $x \in R^{|N|}$ definiscono come ripartire il valore $v(N)$ della *Grande-Coalizione* tra i giocatori in modo che vengano rispettati i seguenti vincoli:

- **Razionalità della coalizione**: La divisione è definita equa se:

$$\sum_{i \in N} x_i = v(N)$$

- **Razionalità individuale**: La divisione garantisce che un singolo giocatore può ottenere un vantaggio nell' unirsi ad una coalizione piuttosto che rimanere da solo

$$\forall i \;\; x_i \geq v(\{i\})$$

A tal fine il nucleo C di un gioco di coalizione (TU) è definito nel modo seguente:

$$C_{TU} = \{x : \sum_{i \in N} x_i = v(N) \quad e \quad \sum_{i \in S} x_i = v(S) \; \forall \; S \; \subseteq N)\} \tag{2.1}$$

In altre parole, il nucleo è l'insieme delle soluzioni in cui nessuna coalizione $S \subseteq N$ è incentivata a rifiutare il valore che gli viene assegnato. Chiaramente, quando si è in grado di trovare un'allocazione ottima che sta nel nucleo, allora la *Grande-Coalizione* è una soluzione stabile e ottimale per il gioco.

Per applicare ai giochi (NTU) questo concetto di soluzione è necessario che la funzione caratteristica v soddisfi le seguenti proprietà per qualsiasi coalizione S

- Il valore $v(S)$ di qualsiasi coalizione S deve essere un sottoinsieme chiuso e convesso di $R^{|S|}$.
- Il valore $v(S)$ deve essere globale, questo implica che se $x \in v(S)$, $y \in R^{|S|}$ e $y \leq x$, allora $y \in v(S)$.
- L'insieme $\{x \mid x \in v(S) \; e \; x_i \geq z_i, \; \forall i \; \in S\}$ con:

$$z_i = max\{y_i \mid y \in v(\{i\})\} < \infty, \forall i \in n$$

  deve essere un sottoinsieme limitato di $R^{|S|}$.

Le proprietà descritte implicano che se esiste un certo profitto x raggiungibile dai componenti di una coalizione S, allora, se i membri di S cambiassero la propria strategia otterrebbero un profitto y dove $y \leq x$. L'ultima proprietà implica che per una coalizione S ciascun giocatore in S riceve un valore non inferiore rispetto al valore massimo che potrebbe avere ottenuto non cooperando, a conferma che z_i è un insieme limitato.

Il nucleo di un gioco di coalizione (NTU) è definito come segue:

$$C_{(NTU)} = \{x \in v(N) \quad | \quad \forall s \nexists y \in v(S), \; s.t, \; y_i > x_i, \forall i \in S\} \tag{2.2}$$

Questa definizione garantisce che esiste una *Grande-Coalizione* stabile. L'idea di base è che qualsiasi allocazione presente nel nucleo di un gioco (NTU) garantisce che non esiste

una coalizione $S \subset N$ che ha una ripartizione del proprio valore più conveniente per tutti i suoi membri. A differenza dei giochi (TU) in cui la stabilità è data dalla somma degli elementi in posizione i dei vettori "payoff", nei giochi (NTU) la stabilità è data solo dagli elementi dei vettori "payoff".

Di seguito discutiamo alcune interessanti proprietà del nucleo.

Dato un gioco in forma (TU) o (NTU) l'esistenza del nucleo non è affatto scontata, infatti per molti giochi il nucleo risulta vuoto e di conseguenza la *Grande-Coalizione* non risulta essere una soluzione stabile. In tali casi sono definiti concetti di soluzione alternative come la soluzione ϵ-core in cui, dato un gioco di coalizione $G = \langle N, v \rangle$, una sua soluzione è ϵ-core se vale la seguente proprietà:

$$\forall S \subseteq N, \sum i \in S x_i \geq v(S) - \epsilon$$

Tuttavia nella teoria dei giochi vi sono diverse tipologie di giochi dove esiste una garanzia teorica che il nucleo non è vuoto. Formalmente il nucleo può essere modellato come un sistema di disequazioni lineari deboli. Per tale ragione esso è chiuso e convesso ed è risolvibile come un problema di programmazione lineare. L'esistenza del nucleo per giochi (TU) è strettamente legata al fatto che il suo corrispettivo problema di programmazione lineare abbia una soluzione ammissibile. In generale verificare questa proprietà è un problema NP-completo a causa del numero di vincoli che è esponenziale nel numero di agenti presenti in N. Tuttavia, per determinare che il nucleo è non-vuoto, oltre a calcolare le allocazioni che effettivamente si trovano nel nucleo, esistono diverse tecniche:

- **Approccio Grafico**: Un approccio grafico può essere utilizzato per individuare il nucleo di giochi (TU) con al più 3 giocatori. L'idea consiste nel rappresentare i vincoli su un sistema di assi cartesiani; l'insieme di tali vincoli definisce una regione dello spazio i cui punti rappresentano il nucleo del gioco.

- **Teoria della Dualità** Utilizzando la teoria della dualità la condizione necessaria e sufficiente che deve essere garantita affinché il nucleo non sia vuoto, è il teorema di Bondareva-Shapley che impone che il gioco sia bilanciato.

  Un gioco (TU) si definisce bilanciato se e solo se la disuguaglianza:

  $$\sum_{S \in N} \nu(S) v(S) \leq v(N) \tag{2.3}$$

  è soddisfatta per tutte le collezioni che hanno un peso non negativo

  $$\nu = \nu(S)_{S \subseteq N} \tag{2.4}$$

  dove ν è il vettore dei pesi associato ad ogni coalizione $S \subseteq N$ contenente valori compresi nell'intervallo $[0, 1]$,

$$\sum_{S \subseteq i} \nu(S) = 1 \; \forall i \in N \tag{2.5}$$

Per giochi (NTU) è definito il concetto di gioco equilibrato [64] [8]. Per questa tipologia di giochi la funzione caratteristica v del gioco non è più una funzione ma un insieme. In generale dato un gioco (TU) o un gioco (NTU) equilibrato è possibile applicare il seguente teorema:

Teorema 2.3.1 (Teorema Bondareva-Shapley)**.** *Il nucleo di un gioco è non vuoto se e solo se il gioco è equilibrato.*

Pertanto si può sempre dimostrare che un gioco di coalizione, il cui nucleo è non vuoto, è anche bilanciato.

- **Verifica di Convessità**: Un gioco di coalizione G convesso ha sempre un nucleo non vuoto. Nel caso di giochi (TU) in forma canonica la proprietà di convessità è valida se:

$$v(S_1) + v(S_2) \leq v(S_1 \cup S_2) + v(S_1 \cap S_2) \forall S_1, S_2 \subseteq N.$$

Tale proprietà, se verificata, implica che la funzione caratteristica v del gioco G, è super modulare. In alternativa un gioco di coalizione G convesso è definito come qualsiasi gioco di coalizione che soddisfa la seguente equazione:

$$v(S_1 \cup \{i\}) - v(S_1) \leq v(S_2 \cup \{i\}) - v(S_2),$$

Ogni qualvolta che si verifica la condizione $S_1 \subseteq S_2 \subseteq N \setminus \{i\}$, allora il gioco è convesso se e solo se per ogni giocatore $i \in N$ il contributo marginale del giocatore non è decrescente rispetto all'operazione di inclusione di insiemi.

La proprietà di convessità può essere estesa anche ai giochi (NTU) come descritto nel lavoro [8]. Per entrambe le tipologie di giochi (TU) e (NTU) , un gioco è convesso ed equilibrato se ha un nucleo non vuoto. L'inverso non è detto che sia sempre vero.

Infine è possibile concludere che i giochi convessi costituiscono un importante classe di giochi in cui il nucleo è non vuoto.

Esempio 2.2. *Sia $G = \langle h, N \rangle$ un gioco di coalizione, la cui funzione caratteristica v è data dalle seguente distribuzione*

$$\begin{array}{ll} v(\{a\}) = 0 & v(\{a,b\}) = 20 \\ v(\{b\}) = 0 & v(\{a,b\}) = 30 \;\; v(\{a,b,c\}) = 45 \\ v(\{c\}) = 0 & v(\{b,b\}) = 40 \end{array}$$

Le regola di divisione per questo gioco è una funzione che deve garantire che le triple (x_a, x_b, x_c) *soddisfino i seguenti vincoli:*

$$\begin{cases} x_a + x_b + x_c = 45, \\ x_a \geq 0 \\ x_b \geq 0 \\ x_c \geq 0 \end{cases}$$

Se si considera la ripartizione x *tale che:* $x_a = 5$*,* $x_b = 15$*,* $x_c = 25$ *si vede facilmente che* x *fa parte del nucleo di* G*.*

Esempio 2.3. *Sia* $G = \langle N, v \rangle$ *un gioco di coalizione, con* $N = \{a, b, c\}$ *e la seguente distribuzione*

$$\begin{aligned} &v(\{a\}) = v(\{b\}) = v(\{c\}) = 0 \\ &v(\{a, b\}) = 20 \\ &v(\{a, c\}) = 30 \\ &v(\{b, c\}) = 40 \\ &v(\{a, b, c\}) = 42 \end{aligned}$$

Se si considera la ripartizione x *tale che:* $x_a = 4$*,* $x_b = 14$*, e* $x_c = 24$*. Si può osservare che tale soluzione* $v(\{b, c\}) = 40 > x(\{b, c\}) = 38$ *non è nel nucleo di* G*. Inoltre è possibile dimostrare che per questo gioco il nucleo è vuoto. Infatti basta osservare che* $v(\{a, b, c\}) = 42$ *e considerare coalizioni* $S_1 = \{a, b\}, S_2 = \{a, c\}$ *ed* $S_3 = \{b, c\}$*. Dalla definizione del gioco si ricava che* x *deve soddisfare le seguenti disuguaglianze:*

$$\begin{cases} x_a + x_b \geq 20 \\ x_a + x_c \geq 30 \\ x_c + x_c \geq 40 \end{cases}$$

Ne consegue che $(x_a + x_b) + (x_a + x_c) + (x_b + x_c) \geq 20 + 30 + 40$ *cioè che* $x_a + x_b + x_c \geq 45$*, quindi la grande coalizione deve ottenere un valore complessivo di 45 invece di 42 al fine di soddisfare gli agenti. Di conseguenza, la grande coalizione non potrà mai costituirsi per questo gioco.*

2.3.2 Il kernel

Il kernel è un concetto di soluzione basato su una funzione di ridistribuzione in cui, data una coppia di membri di una coalizione i e j, è possibile verificare che il massimo *surplus* della stessa coalizione senza il membro j non può essere superiore al *surplus* che j

otterrebbe nella coalizione senza i. È possibile definire il kernel di un gioco di coalizione come il massimo surplus che un giocatore i può avere su un giocatore j rispetto ad una allocazione x tale che:

$$s_{ij}^{v}(x) = \max\{v(S) - \sum_{k \in S} x_k : S \subseteq N \setminus \{j\}, i \in S\} \tag{2.6}$$

Il surplus massimo rappresenta un modo per misurare il peso di un giocatore i rispetto ad giocatore j. Il kernel di v è l'insieme delle allocazioni x che soddisfano le seguenti formule:

$$(s_{ij}^{v}(x) - s_{ji}^{v}(x)) \times (x_j - v(j)) \leq 0$$
$$(s_{ji}^{v}(x) - s_{ij}^{v}(x)) \times (x_i - v(i)) \leq 0$$

2.3.3 Il nucleolo

Un ulteriore concetto di soluzione per i giochi di coalizione, introdotto principalmente per i giochi (TU), è il nucleolo. La logica alla base di questa soluzione è di rilasciare alcuni vincoli garanti l'equità di una formula di divisone, e definire quindi una nuova formula di ridistribuzione v che riduce al minimo l'insoddisfazione dei giocatori.

Data una coalizione S, la misura di insoddisfazione dell'assegnazione $x \in R^N$ è definita come *excess*

$$e(x, S) = v(S) - \sum_{j \in S} x_j$$

Chiaramente un'allocazione stabile in questo contesto è una allocazione in grado di garantire che gli *excess* di tutti gli autori siano ridotti al minimo. È importante osservare che un'allocazione appartiene al nucleo se e solo se tutti i suoi *excess* sono minori o uguali a zero. Una definizione alternativa di questo concetto di soluzione è la seguente: Sia $G = \langle N, v \rangle$ un gioco di coalizione e sia x un regola di ridistribuzione. Questa appartiene al nucleolo del gioco G se x è soluzione di un insieme di problemi di ottimizzazione $O_1, O_2, ..., O_{|N|}$ definiti come segue:

$$O_i = \begin{cases} min\ \epsilon \\ \sum_{i \in S} x_i \geq v(S) - \epsilon & \forall S \subseteq S_1 \\ \vdots \\ \sum_{i \in S} x_i \geq v(S) - \epsilon - 1 & \forall S \subseteq S_{i-1} S_{i-2} \\ \sum_{i \in S} x_i \geq v(S) - \epsilon & \forall S \subseteq 2^N S_{i-1} S_{i-2} \end{cases}$$

Dato un vettore $O(x)$ contenente tutti gli *excess*, meno quello relativo alla *Grande-Coalizione*, sono disposti in ordine crescente, dato un vettore $y = (y_1, \ldots, y_k)$ e un vettore

$z = (z_1, \ldots, z_k)$, dove i singoli elementi del vettore y sono disposti in modo da garantire che $y <_{lex} z$ se

$$\exists \ l \in \{1, \ldots, k\}$$

tale che:

$$\begin{aligned} y_1 &= z_1 \\ y_2 &= z_2 \\ &\vdots \\ y_{L-1} &= z_{L-1} \\ y_l &< z_l \end{aligned}$$

Un'allocazione x appartiene al nucleolo se per ogni altra possibile allocazione δ vale che:

$$O(x) <_{lex} O(\delta)$$

Alla luce di questo è possibile definire il nucleolo di un gioco G come un'allocazione x che minimizza tutti gli *excess*. Tale funzione gode delle seguenti proprietà:

- Dato un gioco di coalizione G l'esistenza del nucleolo è sempre garantita ed è unica.
- Dato un gioco G se il nucleo non è vuoto, allora il nucleolo è contenuto nel nucleo.
- Dato il nucleolo di un gioco G, esso appartiene al kernel di G.
- Il nucleolo di un gioco G è simmetrico e soddisfa l'assioma fittizio di Shapley.
- Il nucleolo è la migliore allocazione sotto il criterio min-max.

Per calcolare il nucleolo è necessario inizialmente trovare le imputazioni del gioco che distribuiscono il valore della *Gran Caolition* in modo tale che sia minimizzata l'insoddisfazione dei singoli giocatori. Nel caso in cui il processo di minimizzazione presenti un'unica soluzione, questa è il nucleolo del gioco; in caso contrario si cercano le imputazioni che minimizzano la seconda soluzione con *excess* più grande. La procedura viene ripetuta iterativamente fino a trovare l'unica soluzione che sta nel nucleolo. I singoli problemi di minimo sono risolti utilizzando tecniche note di programmazione lineare come il metodo del simplesso.

2.3.4 Il valore di Shapley

Abbiamo visto che il nucleo rappresenta l'insieme delle possibili allocazioni in cui il valore di gioco non può essere ulteriormente migliorato. Il nucleo generalmente presenta diversi inconvenienti, uno di questi è che, tale insieme, può risultare vuoto (come nel caso nei giochi di maggioranza) o viceversa quando non è vuoto può contenere diverse allocazioni e quindi non ci offre una "singola soluzione" ma ci fornisce piuttosto un modo per scartare tutte quelle allocazioni instabili (cioè quelle allocazioni in cui il singolo ha interesse ad abbandonare la grande coalizione N). Infine, a questi due problemi se ne aggiunge un altro: alcune volte la divisione dei payoffs non è equa e, di nuovo, il singolo giocatore potrebbe essere incentivato ad uscire dalla grande coalizione se si insiste su quella divisione. A tutti questi problemi si risponde con un altro concetto di soluzione, il valore di Shapley, appunto. Abbiamo già visto nella Sezione 2.2 che un gioco di coalizione $G = (N, v)$ è composto da un insieme finito di giocatori $N = \{1,2,3...,n\}$ e da una funzione v che assegna all'insieme delle possibili coalizioni $2^N + 1$ un insieme di numeri reali indicante il valore che una data coalizione $S \subseteq N$ può ottenere nel gioco G. Si ricordi che il valore di una coalizione vuota è pari a $v(\emptyset) = 0$.

Definizione 2.4 (Valore di Shapley)**.** *Sia $G = (N, v)$ un gioco cooperativo. Il* valore di Shapley *di un giocatore i è dato da*

$$\phi_i(G) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(n-|S|-1)!}{n!} \Big(v(S \cup \{i\}) - v(S) \Big), \textit{ dove } v(S \cup \{i\}) - v(S)$$

è il contributo marginale *di i alla coalizione $S \cup \{i\}$.*

Intuitivamente, il valore di Shapley $\phi(G)$ ridistribuisce il punteggio ottenuto dalla coalizione tra i suoi membri in funzione del contributo marginale che ogni singolo elemento apporta alla coalizione. Più precisamente, il valore di Shapley di ogni giocatore i è determinato dalla media dei contributi marginali di i su tutte le possibili coalizioni alle quali i può partecipare. Alcune proprietà ben note (si veda [52, 57]) del valore di Shapley di un qualunque gioco cooperativo $G = (N, v)$ sono le seguenti:

1. **Efficienza**: $\sum_{i \in N} \phi_i(G) = v(N)$;
2. **Stabilità**: Se v è *supermodulare*, il valore di Shapley appartiene al nucleo del gioco;
3. **Monotonia**: Se $G' = (N, v')$ è un gioco cooperativo tale che $v'(S) \geq v(S), \forall S \subseteq N$, allora $\phi_i(G') \geq \phi_i(G), \forall i \in N$.
4. **Simmetria**: Dati due giocatori i e j tali per che $v(S \cup \{i\}) = v(S \cup \{j\})$ allora per tutte le coalizioni S che non contengono i giocatori i e j, deve valere che $\phi(i, v) = \phi(j, v)$. Questo implica che giocatori con lo stesso contributo marginale ricevono una stessa ricompensa.

5. **Giocatore fittizio**: Dato un giocatore i tale che $v(S) = v(S \cup \{i\})$ per ogni coalizione S che non contiene i, vale che $\phi(i, v) = 0$.

6. **Additività**: Dati due giochi $G_1 = (N, v)$ e $G_2 = (N, u)$ le cui funzioni caratteristiche sono rispettivamente u e v, la combinazione dei due giochi ha come funzione caratteristica
$$\phi(i, v + u) = \phi(i, v) + \phi(i, u) \ \ \forall i \in N$$
L'additività afferma quindi che il valore di ciascun giocatore in un gioco che è la somma di due giochi è pari alla somma dei valori di quel giocatore nei singoli giochi.

7. **Anonimia**: Per tutti i giochi (N, v) e per tutte le permutazioni σ di N si ha che: $\phi_{\sigma(i)} = \phi(N, \sigma)$ dove il gioco σ_v è definito come $\sigma_s(S) = v(\sigma(S))$ per tutti i $S \subseteq N$.

8. **Trasferimento**: Per tutti i giochi $(N, p), (N, w)$ si ha che
$$\phi(N, p) + v(N, w) = \phi(N, p \vee w) + \phi(N, p \wedge w)$$
dove $(N, p \vee w), (N, p \wedge w)$ è definita per tutti gli insiemi $S \subseteq N$ come:
$$(p \vee w) = max\{p(S), w(S)\}$$
$$(p \wedge w) = min\{p(S), w(S)\}$$

9. **Contributi Marginali**: Data una coppia di giochi $(N, p), (N, w)$ e per tutti gli agenti $i \in N$ tale che: $p(S \cup i) - p(S) \leq w(S \cup i) - w(S)$ per tutti $S \subseteq N \setminus i$ si ha che $v_i(N, v) \leq v_i(N, w)$.

Teorema 2.3.2 (Shapley (1953), Young (1985), Feltkamp (1995))**.**
Il valore di Shapley è l'unico valore che soddisfa le proprietà di efficienza, simmetria, additività, giocatore fittizio, monotonia, trasferimento, stabilità.

Abbiamo visto che la formula per il calcolo del valore di Shapley è basata sul concetto di contributo marginale $[v(S \cup \{i\}) - v(S)]$ che il giocatore i apporta alla coalizione S premoltiplicato per il coefficiente

$$\frac{|S|!(N - |S| - 1)!}{N!} \tag{2.7}$$

che è la probabilità che il giocatore i si unisca alla coalizione S in cui gli agenti sono disposti casualmente. Nel contesto in questione vi sono $|S|!$ possibili modi di disporre i membri di S, e $(N - |S| - 1)!$ modi per disporre i restanti giocatori (eccetto i). La probabilità che si verifichi un tale ordinamento, basandosi sull'ipotesi che ogni ordinamento

è equi probabile, è espressa dalla formula 2.7. In generale, il valore di Shapley non è correlato al nucleo del gioco, tuttavia è stato dimostrato che per giochi convessi, il valore di Shapley appartiene al nucleo del gioco [4] [1]. Questo è un risultato molto interessante, in quanto si possono sfruttare gli assiomi del nucleo che garantiscono stabilità e la correttezza del valore Shapley.

Si noti che il numero di operazioni necessarie per il calcolo del valore di Shapley utilizzando direttamente la definizione è esponenziale nel numero degli agenti del gioco. È stato in effetti dimostrato da Deng-Papadimitriou [10] e da Matsui-Matsui [11] che determinare se un dato giocatore ha un valore Shapley diverso da zero è un problema *NP-hard*, e che il calcolo del valore esatto è *#P-hard*. Un altro risultato importante ottenuto da Matsui-Matsui [12] è il seguente: è possibile calcolare il valore esatto di Shapley con un algoritmo pseudo-polinomiale che fa uso della programmazione dinamica.

2.3.5 Indice di Banzhaf

L'analisi della ridistribuzione del potere in una comunità è un problema centrale nello studio delle scienze politiche. In generale, è molto difficile determinare il peso di un agente all'interno di un dato gioco. Per i giochi di maggioranza, sono stati definiti diversi indici per rispondere a questa domanda. Il primo di questi è il valore di Shapley, di cui abbiamo discusso nella sezione precedente. Tuttavia esiste un altro concetto, anch'esso basato sul concetto di contributo marginale, che permette di misurare il potere di voto di un agente o di una coalizione, chiamato indice di Banzhaf [53]. In questa sezione si introduce l'indice di Banzhaf, mostrandone le relazioni e le similitudini con il valore di Shapley. Dato un gioco (N, v) e una funzione f che assegna al gioco un vettore di numeri reali $f(N, v) \in \mathbb{R}^n$ e dato un giocatore $i \in N$ si definisce la distribuzione di probabilità $\{p_i | S \subseteq N \setminus \{i\}\}$ sull'insieme delle coalizioni che non contengono i, inoltre si definisce la funzione f come:

$$f_i(N, v) = \sum_{S \subseteq N \setminus i} p_S^i \cdot [v(S \cup i) - v(S)]$$

dove i è un agente che decide di unirsi a tutte le coalizioni $S \subseteq N$ ricevendo come ricompensa il suo contributo marginale $v(S \cup \{i\}) - v(S)$. Per ogni $S \subseteq N \setminus \{i\}$ si definisce p_S^i la probabilità che il giocatore i si unisca alla coalizione S, quindi la funzione $f_i(N; v)$ rappresenta semplicemente il payoff del giocatore i. Osservando la formula per il calcolo del valore di Shapley 2.4 è possibile osservare che questa può essere riscritta come una particolare istanza della formula f:

$$p_i^i = \frac{n-1}{2}^{-1} \binom{n}{k} = \frac{|S|!(N - |S| - 1)!}{N!}$$

D'altra parte, il valore Shapley si basa sul presupposto che per ogni giocatore le coalizioni di una data dimensione a cui lui aderisce, sono equiprobabili.

L'indice di Banzhaf è definito dalla formula:

$$\beta_i(V) = \frac{1}{2^{n-1}} \sum_{S \subseteq N \setminus \{i\}} [v(S \cup \{i\}) - v(S)] \tag{2.8}$$

Come il valore di Shapley, anche l'indice di Banzhaf può essere visto come una particolare istanza della formula f dove

$$p_i^i = \frac{1}{2^{n-1}}$$

per ogni $S \in N \setminus \{i\}$ ma, diversamente dal valore di Shapley, l'indice di Banzhaf si basa sul presupposto che ogni giocatore ha la stessa probabilità di partecipare a qualsiasi coalizione. Le proprietà del valore di Shapley descritte nella sezione precedete risultano tutte valide per il valore di Banzhaf ad eccezione della proprietà relativa all'efficienza. Inoltre l'indice di Banzhaf presenta proprietà aggiuntive qui riportate:

- **2-Efficiency**: Dato un gioco (N, v) per tutte le coppie di giocatori $i, j \in N$ si ha che: $f_i(N, v) = f_j(N, v) \geq f_p(N^{i,j}, v^{i,j})$ dove $(N^{i,j}, v^{i,j})$ è il gioco ottenuto da (N, v) dove per ogni coppia di giocatori i, j si costruisce un nuovo giocatore p che ne è la fusione:

$$N^{i,j} = (N \setminus \{i, j\}) \cup \{p\}$$

$$v^{i,j} = \begin{cases} v(S) & \text{se } p \notin S \\ v((S \setminus p) \cup i \cup j) & \text{se } p \in S \end{cases}$$

- **2-Efficiency***: Dato un gioco (N, v) per tutte le coppie di giocatori $i, j \in N$ si ha che:

$$f_i(N, v) = f_j(N, v) \geq f_p(N^{i,j}, v^{i,j}) \tag{2.9}$$

- **Potere totale**: Dato un gioco (N, v) si ha che:

$$\sum_{i \in N} f_i(N, v) = \frac{1}{2^{n-1}} \sum_{i \in N} \sum_{S \subseteq N \setminus \{i\}} [v(S \cup \{i\}) - v(S)]$$

Il valore di Banzhaf, come anche quello di Shapley, è calcolato mediante la formula 2.8. Tuttavia nei contesti in cui la cardinalità dell'insieme dei giocatori è grande, il problema risulta intrattabile, in quanto la complessità computazionale è legata al numero di operazioni da effettuare sulle possibili coalizioni, che è esponenziale rispetto al numero degli agenti. Basti osservare che il numero di permutazioni è pari a $|n|!$ dove n è il numero di agenti del gioco. Utilizzando l'approssimazione di Stirling, è possibile verificare che il numero di coalizioni da calcolare è di circa $O(2^{n \log(n)})$. È stato inoltre dimostrato che il calcolo dell'indice di Banzhaf è un problema #P-hard.

2.4 Algoritmi approssimati per il valore di Shapley

In questo lavoro di tesi siamo particolarmente interessati alla valutazione del valore di Shapley. Considerato che si tratta di un problema intrattabile, oltre alla ricerca di isole di trattabilità, è importante ricordare le principali tecniche di approssimazione descritte in letteratura.

In primo luogo, si ricorda il metodo proposto in [56] ed elaborato ulteriormente in [55], dove è stata presentata una tecnica per giochi di coalizione con voto ponderato. Questa tipologia di giochi prevede che ogni giocatore controlli un certo numero di voti, o più in generale, che abbia un peso. L'obiettivo del gioco è costruire coalizioni che permettano di massimizzare gli interessi degli agenti. Una coalizione risulta *vincente* se il numero complessivo di voti dei suoi membri supera una soglia predefinita o, in caso contrario, *perdente*. Fatima propone un metodo per approssimare il valore di Shapley per questa particolare sottoclasse di giochi evitando di calcolare i contributi marginali per le 2^n coalizioni e considerandone solo n selezionate in modo del tutto casuale. Le n coalizioni sono scelte in modo da garantire che ve ne sia una per ogni possibile dimensione della coalizione che varia tra 1 e n. Solo per queste n coalizioni vengono calcolati i contributi marginali attesi di ogni giocatore. Il valore di Shapley approssimato per ogni giocatore è pari alla media dei suoi contributi marginali alle diverse coalizioni.

Un metodo diverso è stato proposto in [65] nel contesto di giochi di coalizione semplici in cui la funzione caratteristica è binaria. Per questi giochi, Bachrach ha proposto di estendere l'approccio suggerito da Mann e Shapley (1960) cercando di fornire un'analisi statistica più rigorosa. In particolare Mann e Shapley descrivono un approccio basato su tecnica Monte Carlo volta a stimare il valore di Shapley partendo da un campione casuale di coalizioni. Bachrach utilizza questa tecnica per calcolare l'indice di Banzhaf e quindi suggerisce l'utilizzo di un campione casuale di permutazioni di tutti i giocatori al fine di calcolare l'indice di Shapley-Shubik per i giochi di coalizione semplici. Quello che si osserva è che il calcolo dell'intervallo di confidenza risulta cruciale per tale approccio ed è inoltre strettamente legato alla forma binaria della funzione caratteristica. Questo metodo è più generale rispetto a quello proposto da Fatima, e applicabile ai giochi di maggioranza ponderato che sono un sottoinsieme dei giochi di coalizione semplici.

A differenza dei primi due, il metodo presentato in [58] è applicabile a tutte le tipologie di giochi di coalizione in cui la logica del gioco è cablata nella sua funzione caratteristica. Assumendo che il valore di ogni coalizione può essere calcolato in tempo polinomiale, l'algoritmo richiede la generazione di un insieme di permutazioni che coinvolgono tutti i giocatori. Per ciascuna di queste permutazioni l'algoritmo calcola il contributo marginale di ogni giocatore alla coalizione considerata. Il valore di Shapley è calcolato come la media di questi contributi. La precisione del metodo aumenta (statisticamente) ogni qual volta viene considerata una nuova permutazione. Gli autori, nel loro lavoro, mostrano come stimare il numero delle permutazioni affinché il risultato finale presenti un errore accettabile. Data la sua ampia applicabilità, questo metodo viene

analizzato più nel dettaglio e specificato per il gioco in questione.

Nel Capitolo 5 sarà introdotto un ulteriore algoritmo approssimato utile per giochi di coalizione che siano monotoni supermodulari [25], di cui si proporrà una variante.

Capitolo 3

Giochi di allocazione

Tra tutte le classi di giochi di coalizione, in questo lavoro di tesi consideriamo i giochi di allocazione, che forniscono un modello matematico per analizzare problemi di divisione di risorse tra i diversi agenti in modo tale che tutti gli agenti possano ritenere di aver ricevuto una giusta ricompensa, secondo una qualche nozione di equità (o fairness) . In questa tesi, analizzeremo un setting in cui è dato uno scenario di allocazione che comprende un insieme di beni e un insieme di agenti e, ad ogni agente è assegnato al più un bene di suo interesse.

È ben noto che il calcolo del valore di Shapley di questi giochi è un problema difficile, infatti è #P-hard anche sotto il vincolo che i beni possano avere al più due valori diversi [14]. Come vedremo nei prossimi capitoli, il nostro obiettivo sarà quello di affrontare questo problema attaccando istanze che coinvolgono migliaia di agenti e beni. Si noti che per questi scenari non esiste in letteratura un algoritmo in grado di fornire una soluzione esatta. Tuttavia, recentemente sono stati fatti dei passi avanti nell'identificazione di isole di trattabilità per il problema: è stato infatti osservato che tutte le istanze avente un grafo di interazione tra agenti di treewidth limitata da qualche costante (cioè aventi un piccolo grado di ciclicità) possono essere risolte in modo esatto in tempo polinomiale [14]. Tale risultato è basato su dei recenti passi avanti fatti nella teoria delle basi di dati e in particolare nel calcolo delle soluzioni delle interrogazioni congiuntive con variabili esistenziali [54]. Sfortunatamente, nel caso in cui la struttura del grafo delle interazioni è molto intricata e il numero di agenti e beni è molto elevato, questa tecnica non può essere applicata, poichè la sua complessità computazionale è dipendente dalla treewidth.

È utile osservare che in alcune applicazioni tali per cui il calcolo del valore di Shapley non può essere eseguito in modo efficiente, potremmo comunque essere soddisfatti con qualche approssimazione del valore. A riguardo, in linea di principio le cose vanno abbastanza bene, dal momento che sappiamo che esiste uno *schema di approssimazione in tempo pienamente polinomiale* (o fully polynomial-time randomised approximation scheme) per il calcolo del valore di Shapley nei giochi monotoni supermodulari. L'algoritmo può essere messo a punto per ottenere un errore atteso desiderato. Ad ogni modo, e ciò

non sorprende, per istanze di grandi dimensioni, occorre considerare un elevato numero di campioni per poter rimanere al di sotto di un errore atteso ragionevole.

3.1 Formalizzazione del problema di allocazione

Nel setting considerato in questo lavoro di tesi, un gioco è definito da uno scenario di allocazione $\mathscr{A} = \langle N, \mathbb{G}, \Omega, \texttt{val}, k \rangle$ che comprende un insieme di agenti N e un insieme di beni $\mathbb{G}$, i cui valori sono dati dalla funzione a valori reali $\texttt{val}$. La funzione Ω associa ad ogni agente un insieme di beni a cui l'agente è interessato. Inoltre, il numero naturale k fornisce il massimo numero di beni che può essere assegnato ad ogni agente. Si osservi che i beni sono indivisibili e non è possibile condividerli.

Per una coalizione $C \subseteq N$, un'allocazione (ammissibile) $\pi_{\mathscr{A}}[C]$ è un mapping da C a insiemi di beni presi da $\mathbb{G}$ in modo tale che: ogni agente $i \in C$ riceve un insieme di beni $\pi_{\mathscr{A}}(i) \subseteq \Omega(i)$ con $|\pi_{\mathscr{A}}(i)| \leq k$, e $\pi_{\mathscr{A}}(i) \cap \pi_{\mathscr{A}}(j) = \emptyset$, per ogni altro agente $j \in C$ (ogni bene può essere assegnato al più ad un agente).

Denotiamo con $\texttt{img}(\pi_{\mathscr{A}}[C])$ l'insieme di tutti i beni nell'immagine di $\pi_{\mathscr{A}}[C]$, i.e., $\texttt{img}(\pi_{\mathscr{A}}[C]) = \bigcup_{i \in C} \pi_{\mathscr{A}}[C](i)$. Con un lieve abuso di notazione, denotiamo con $\texttt{val}(S)$ la somma dei valori di un insieme di beni con $S \subseteq \mathbb{G}$, e con $\texttt{val}(\pi_{\mathscr{A}}[C])$ il valore $\texttt{val}(\texttt{img}(\pi_{\mathscr{A}}[C]))$. Un'allocazione $\pi_{\mathscr{A}}[C]$ si dice ottima se non esiste un'altra allocazione $\pi'_{\mathscr{A}}[C]$ avente $\texttt{val}(\pi'_{\mathscr{A}}[C]) > \texttt{val}(\pi_{\mathscr{A}}[C])$. Il valore totale di una tale allocazione ottima per la coalizione C è denotata con l'espressione $\texttt{opt}_{\mathscr{A}}(C)$. Il budget disponibile per $\mathscr{A}$, noto anche con il nome di (massimo) benessere sociale (o maximum social welfare) è $\texttt{opt}_{\mathscr{A}}(N)$, cioè a dire, il valore di una qualunque allocazione ottima per l'intero insieme di agenti N (la grande coalizione). Il gioco di coalizione definito dallo scenario $\mathscr{A}$ è la coppia $\langle N, \texttt{opt}_{\mathscr{A}} \rangle$, ovvero, il gioco in cui il valore di una qualunque coalizione è dato dal valore di una qualsiasi tra le sue allocazioni ottime. Si noti che $\texttt{opt}_{\mathscr{A}}(C) \geq 0$ è vera, per ogni $C \subseteq N$, dal momento che l'allocazione in cui non è assegnato alcun bene a nessuno degli agenti è un'allocazione (ammisibile) (il valore di un insieme vuoto di beni è 0). La definizione diventa banale per $C = \emptyset$, con $\texttt{opt}_{\mathscr{A}}(\emptyset) = 0$.

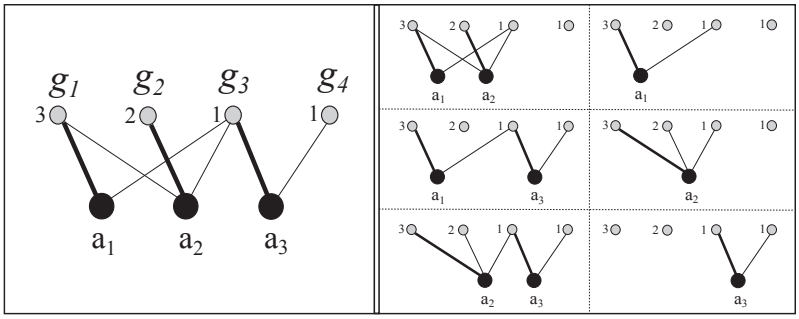


Figura 3.1: Scenario di allocazione $\mathscr{A}_0$ nell'Esempio 3.1.

Esempio 3.1. Si consideri lo scenario di allocazione $\mathscr{A}_0$, illustrato graficamente in Figura 3.1. Abbiamo un insieme $\{g_1, g_2, g_3, g_4\}$ di beni che devono essere allocati a tre agenti. Per semplicità si assuma che è possibile allocare solo un bene ad ogni agente, di conseguenza ogni arco connette un agente ad un bene a cui lui è interessato. Gli archi in grassetto identificano un'allocazione ottima, i.e., un allocazione (ammissibile) la cui somma di valori dei beni allocati è la massima possibile. Il valore di questa allocazione è $\texttt{val}(g_1) + \texttt{val}(g_2) + \texttt{val}(g_3) = 3 + 2 + 1 = 6$.

Per ogni $C \subset \{1,2,3\}$ con $C \neq \emptyset$, riportiamo anche un'allocazione ottima ristretta agli agenti in C. Concludendo, il gioco di coalizione associato è $\mathsf{G}_{\mathscr{A}_0} = \langle \{1,2,3\}, v_{\mathscr{A}_0} \rangle$, dove $v_{\mathscr{A}_0}(\{1,2,3\}) = 6$, $v_{\mathscr{A}_0}(\{1,2\}) = 5$, $v_{\mathscr{A}_0}(\{1,3\}) = v_{\mathscr{A}_0}(\{2,3\}) = 4$, $v_{\mathscr{A}_0}(\{1\}) = v_{\mathscr{A}_0}(\{2\}) = 3$, e $v_{\mathscr{A}_0}(\{3\}) = 1$. ◁

3.2 Intrattabilità del calcolo del valore di Shapley

In precedenza è stato osservato che calcolare il valore Shapley è un problema difficile (formalmente #P-completo) per diverse classi di giochi, inclusi i giochi di allocazione. Tale risultato è stato dimostrato in [13] dove si fa riferimento a casi in cui è possibile allocare al più 3 beni; il risultato è stato ulteriormente migliorato in [14] dove gli autori mostrano che non vi è alcun vantaggio in termini di complessità computazionale nel considerare scenari in cui il numero di beni da allocare si riduce a 1.

Teorema 3.2.1 ([14]). *Il calcolo dell'indice di Banzhaf è un problema #P-hard nei giochi di allocazione, anche in scenari* $\mathscr{A} = \langle N, G, \Omega, \texttt{val} \rangle$ *tali che* $|\{\texttt{val}(g) \mid g \in G\}| = 1$.

Teorema 3.2.2 ([14]). *Il calcolo del valore di Shapley è #P-hard nei giochi di allocazione, anche in scenari* $\mathscr{A} = \langle N, G, \Omega, \texttt{val} \rangle$ *tali che* $|\{\texttt{val}(g) \mid g \in G\}| = 1$.

Da questi risultati, emerge che agendo sui valori dei beni non si identificano delle classi di istanze trattabili.

3.3 Un caso di studio: La valutazione della qualità della ricerca italiana

Nel 2012, l'agenzia nazionale per la valutazione delle università e degli Istituti di Ricerca ANVUR ha promosso il programma **VQR** per valutare la qualità dei lavori di ricerca italiani nel periodo 2004-2010. Il programma di valutazione è stato poi ripetuto per il periodo 2011-2014, con alcune variazioni ininfluenti per gli scopi del presente lavoro. Ogni struttura di ricerca R deve selezionare un insieme di pubblicazioni e sottoporli all'ANVUR. Nel farlo, la struttura R è in concorrenza con tutte le altre strutture di ricerca

italiane, l'esito della valutazione sarà utilizzato per trasferire in modo proporzionale i fondi stanziati dal Ministero per sostenere le attività di ricerca fino al successivo processo di valutazione. Ogni struttura R deve quindi selezionare e presentare i prodotti di ricerca migliori.

Il programma VQR è diviso in tre fasi di seguito riportate:

- **Autovalutazione**: Gli autori affiliati ad una struttura R sono invitati ad auto-valutare i loro prodotti, secondo dei criteri definiti da gruppi di esperti nominati dall'ANVUR. Si presume che avendo a disposizione tali criteri, ogni autore sia in grado di assegnare ad ogni suo prodotto un punteggio in base a delle linee guida definite dalla VQR. Per questo lavoro di tesi i dettagli del calcolo sono volutamente tralasciati in quanto irrilevanti e inoltre si assume che per ogni prodotto di ricerca sia definito un punteggio rappresentato come un numero reale. Nel caso di valutazioni basate su criteri puramente bibliometrici questa fase può anche essere effettuata centralmente in modo automatico.

- **Sottomissione** Sulla base delle autovalutazioni raccolte, la struttura R seleziona e sottopone all'ANVUR un massimo di k prodotti per ognuno dei suoi autori in modo tale che ogni prodotto sia formalmente associato ad un solo autore. Nel primo esercizio della VQR si aveva $k = 3$, nell'ultimo invece $k = 2$ (anche perché il periodo temporale interessato è stato più breve).

- **Calcolo** L'ANVUR verifica internamente le valutazioni dei prodotti di ricerca che la struttura sottomette e calcola la qualità della ricerca della struttura R sommando i punteggi assegnati a tali pubblicazioni. Infine la struttura R riceverà nei successivi anni una quantità di fondi proporzionale al punteggio ottenuto. Con le medesime informazioni, ANVUR può inoltre valutare tutte le sotto strutture o tutti i dipartimenti, nel caso la struttura R sia una Università (per semplicità assumeremo nel seguito che sia questo il caso, anche se la VQR esamina anche gli istituti di ricerca).

Sebbene l'obiettivo principale del programma VQR sia la valutazione delle diverse strutture di ricerca italiane e le loro sottostrutture, esso è stato anche usato per altri fini. In particolare, il Ministero ha usato le informazioni della VQR per attribuire parte del fondo di finanziamento ordinario delle università in base alla valutazione VQR dei neo-assunti e, più recentemente, per valutare i collegi di dottorato. Nei prossimi anni la VQR sarà addirittura usata per l'accreditamento dei corsi di laurea magistrale, sulla base dei risultati VQR dei docenti che vi insegnano.

È quindi fondamentale definire delle regole corrette per ripartire il punteggio complessivo della struttura di ricerca alle sue sottostrutture, o a gruppi di ricerca o addirittura singoli autori (si pensi al caso dei neo-assunti), in modo da evitare attribuzioni ingiuste che favoriscano alcuni a scapito di altri [20].

Regole imparziali per la ripartizione delle risorse. Sia R una struttura di ricerca, e sia R' l'insieme di ricercatori affiliati ad R. Per ogni ricercatore $r \in R$, si definisce $product(r, R)$ l'insieme dei prodotti della struttura R in un periodo prefissato.

Dato un insieme di ricercatori $S \subseteq R$, si definisce una funzione di allocazione ψ che assegna ad ogni ricercatore $R \in S$ un insieme di prodotti (pubblicazioni) tale che: i prodotti assegnati siano un sottoinsieme dei prodotti $\psi(r) \subseteq product(r)$, il numero di prodotti allocati per ogni ricercatore sia $|\psi(r)| \leq 2$ e infine che ogni prodotto g sia assegnato ad un solo agente i ovvero $\psi(r) \cap \psi(r') = \emptyset$ per ogni $r' \in S \setminus \{r\}$.

Un'allocazione ammissibile per una struttura R è un'allocazione che coinvolge tutti i ricercatori di R, ricorsivamente un'allocazione per una sotto struttura S di R è un'allocazione che coinvolge solo i ricercatori affiliati alla sotto struttura S. Il programma della VQR prevede che ogni struttura di ricerca R presenti all'ANVUR un insieme di prodotti P_ψ tale che $P_\psi = \bigcup_{r \in R} \psi(r)$ dove $\psi(R)$ è un'allocazione che coinvolge tutti i ricercatori affiliati alla struttura.

Per ogni $p \in P_\psi$, l'ANVUR quantifica la qualità per la struttura R calcolando il punteggio:

$$score_{VQR}(\psi) = \sum_{p \in P_\psi} score_{VQR}(p) \tag{3.1}$$

Sulla base di questo punteggio la struttura R riceverà proporzionalmente una certa quantità di fondi.

Definizione 3.1. *Una regola di divisione γ per R è una funzione reale che, dato un ricercatore $r \in R$ e un'allocazione ψ per R, restituisce il punteggio per il ricercatore r rispetto all'allocazione ψ, con $\gamma_{\psi(r,R)} \geq 0$.*

Detto questo è possibile definire per qualsiasi sotto struttura $S \subseteq R$ un punteggio $\gamma_\psi(S, R)$ dove:

$$\gamma_\psi(S, R) = \sum_{r \in S} \gamma_\psi(r, R) \tag{3.2}$$

Per questioni di comodità ogni volta che R è sottinteso è possibile scrivere direttamente $\gamma_{\psi(r)}$ e $\gamma_{\psi(S)}$, al posto di $\gamma_{\psi(r,R)}$ e $\gamma_{\psi(S,R)}$. Inoltre si assume che $S_1, ..., S_n$ siano le sotto strutture di R, che coprono in modo esaustivo tutti ricercatori di R, ovvero:

$$\bigcup_{i=1}^{n} S_i = R \tag{3.3}$$

$$S_i \cap S_j = \emptyset \tag{3.4}$$

$$\text{per ogni } i \neq j \tag{3.5}$$

Si noti che ogni ricercatore r è affiliato ad una sola sotto struttura.

Proprietà fondamentali per una regola di divisione imparziale. Una delle più semplici regole di divisione è quella che assegna a ciascun ricercatore $r \in R$ la somma dei punteggi VQR dei prodotti a lui assegnati nell'allocazione ottima (globale per la struttura):

$$proj_{\psi}(r) = \sum_{p \in \psi(r)} score_{VQR}(p) \tag{3.6}$$

La regola $proj$ non soddisfa in generale alcune proprietà desiderabili tipiche delle regole di divisione. Ad esempio questa regola è difficilmente percepibile come equa, poiché r potrebbe magari ottenere un miglior punteggio se vi fosse stata una diversa allocazione (utilizzando qualche prodotto migliore assegnato a qualche suo coautore).

Di seguito si riporta l'elenco di queste proprietà desiderabili e l'eventuale verifica di validità per la formula $proj_{\psi}(r)$

Definizione 3.2 (P1 budget-balance)**.** *Una regola divisione γ deve distribuire con precisione il punteggio VQR della struttura R su tutti i suoi membri, vale a dire,*

$$\sum_{r \in R} \gamma_{\psi(r)} = score_{VQR}(R) \tag{3.7}$$

Definizione 3.3 (P2 L'indipendenza della ripartizione dei prodotti)**.** *Si suppone che ψ e $\overline{\psi}$ siano due allocazioni che coinvolgono i medesimi prodotti, vale a dire, $P\psi = P\overline{\psi}$. Una regola di divisione γ deve essere indipendente dalla ripartizione delle risorse scelte, cioè, per ogni $r \in R$, $\gamma_{\psi}(r) = \gamma_{\overline{\psi}}(r)$. Questo implica che, per ogni sotto struttura S_i vale che: $\gamma_{\psi}(S_i) = \gamma_{\overline{\psi}}(S_i)$ che è la proprietà desiderabile.*

Tuttavia, uno sguardo più attento rivela che anche questo non è sufficiente, in quanto vi è un modo semplice per aggirare questa proprietà: basta considerare la regola di assegnazione del punteggio complessivo della VQR per un dato ricercatore r_i (indipendentemente dal suo contributo effettivo), cioè

$$trivial_{\psi}(r) = \begin{cases} \sum_{p \in P_{\psi}} score_{VQR}(p), & \text{se } r = r_1 \\ 0 & \text{se} r \in R \setminus \{r_i\} \end{cases} \tag{3.8}$$

Il problema di questa regola, è l'assenza di simmetria, in quanto data una coppia di ricercatori r_j e r_i che hanno pubblicato esattamente lo stesso insieme di pubblicazioni, il processo di valutazione li tratta in modo del tutto diverso solo perché presentano nomi diversi. Per risolvere questo problema è necessario garantire che il sistema di valutazione offra una parità di trattamento per i ricercatori, che è un altro requisito fondamentale e formalizzato come segue: sia $\pi : R \mapsto \Re$ una permutazione dei ricercatori in R. Si definisce R_{π} una nuova struttura di ricerca definita su R dove, per ogni $r \in R$ vale che $product(r, R_{\pi}) = product(\pi(r), R)$. Inoltre, se ψ è un'allocazione valida per la struttura R, allora ψ_{π} è un'allocazione in cui $\psi_{\pi}(r, R_{\pi}) = \psi(\pi(r), R)$ per ogni $r \in R$. Così R_{π}

e ψ_π derivano applicando una permutazione π, il cui ruolo è solo quello di rinominare i ricercatori in R.

Con queste nozioni ora è possibile definire la proprietà di imparzialità.

Definizione 3.4 (P3 Imparzialità)**.** *Sia π una permutazione arbitraria dei ricercatori di R e γ una regola divisione che garantisce che per ogni $r \in R$ e per ogni allocazione ψ deve valere che $\gamma_{\psi_\pi}(r, R_\pi) = \gamma_\psi(\pi(r), R)$*

Per capire la difficoltà di definire una regola di divisione soddisfacente, si consideri la regola di distribuzione uniforme

$$uniform_\psi(r) = \frac{score_{VQR}(R)}{|R|} \tag{3.9}$$

Questa regola è simmetrica ma nello stesso tempo risulta essere in generale non imparziale, in quanto il punteggio complessivo di una coalizione verrebbe distribuito uniformemente tra i vari ricercatori non riflettendo di fatto l'effettivo contributo che il singolo ricercatore apporta alla valutazione complessiva della struttura.

Per formalizzare meglio questo concetto, nella seguente sezione si descrive una ulteriore proprietà proposta nel lavoro [20] e basata sul concetto di contributo marginale.

Contributi marginali. Sia R una struttura di ricerca e si supponga che ψ è una data allocazione per la struttura R e P_ψ è l'insieme di prodotti che sono stati sottomessi e valutati dall'ANVUR. Sia $S \subseteq R$ un qualsiasi insieme di ricercatori e sia ψ_S un'allocazione valida per S, ossia $\psi_S(r) \subseteq P_\psi$ per ogni $r \in S$. In generale un'allocazione è considerata valida se e solo se è composta da un insieme di prodotti già valutati dall'ANVUR, inoltre un'allocazione ψ_S è un'allocazione ottima se non vi è alcuna allocazione valida ψ'_S tale che:

$$\sum_{r\ inS} \sum_{p \in \psi'_S(r)} score_{VQR}(p) \geq \sum_{r \in S} \sum_{p \in \psi_S(r)} score_{VQR}(p) \tag{3.10}$$

Date le nozioni di cui sopra, è possibile definire una nuova nozione di punteggio per la VQR basata sul punteggio complessivo ottenuto dai singoli ricercatori di S senza considerare eventuali coautori esterni ad S dove ψ_S è un'allocazione ψ*-ottima*.

$$best_\psi(S) = \sum_{r \in S} \sum_{p \in \psi_S(r)} score_{VQR}(p) \tag{3.11}$$

Si noti che nel caso estremo in cui $S = R$ si ha che $best_\psi(R) = score_{VQR}(R)$. Cioè, quando sono considerati tutti i ricercatori di R, $best_\psi$ restituisce esattamente il punteggio complessivo della VQR.

Detto ciò è possibile formalizzare la nozione di contributo marginale.

Definizione 3.5 (Contributo Marginale)**.**
Sia ψ un'allocazione per R e siano $S_1, S_2 \subseteq R$ con $S_1 \subseteq S_2$, due gruppi di ricercatori; il contributo marginale di S_1 su S_2 (in R e ψ) è definito da:

$$marg_\psi(S_1, S_2) = best_\psi(S_2) - best_\psi(S_2 \setminus S_1). \tag{3.12}$$

Intuitivamente, il contributo marginale $marg_\psi(S_1, S_2)$ quantifica il contributo di un gruppo di ricercatori S_2 che apportano ad S_1. In particolare, $marg_\psi(r, R)$ misura quanto apporta il ricercatore r all'intera struttura R. Detto questo è possibile definire l'ultima proprietà di equità, che quantifica il contributo reale degli individui e dei gruppi ad una sovra struttura.

Definizione 3.6 (P4 Marginalità)**.** *Data una regola divisione ψ tale che, per ogni gruppo di ricercatori $S \subseteq R$ e per ogni allocazione ψ, vale la seguente relazione:*

$$\psi(S) \geq marg_\psi(S, R) \tag{3.13}$$

Pertanto ad ogni gruppo è assegnato almeno il suo contributo marginale che apporta alla struttura R.

Definizione di gioco di allocazione per la valutazione della qualità della ricerca italiana VQR. Una soluzione al problema della valutazione della ricerca italiana è basata sulla definizione di un gioco di coalizione dove i singoli agenti possono collaborare al fine di ottenere risultati migliori rispetto all'agire in modo solitario. Si suppone l'ipotesi che il gioco sia ad utilità trasferibile (TU) e che il valore di una coalizione può essere liberamente ridistribuito tra i suoi membri. Il gioco in questione può essere modellato in forma astratta come una coppia $G = (N, \phi)$, dove $N = \{1, ..., n\}$ è un insieme finito di giocatori e ϕ è una funzione super modulare che quantifica il valore che gli agenti possono ottenere coalizzandosi in C. La funzione ϕ associa ad ogni coalizione $C \subseteq N$ un numero reale $\phi(C) \in \mathfrak{R}$ che rappresenta il valore, con $\phi(\{\emptyset\}) = 0$. Il risultato del gioco G è un vettore di payoff $x = (x_1, ..., x_n) \in \mathfrak{R}^n$ che quantifica come viene redistribuito il valore totale della coalizione $\phi(N)$ tra i suoi membri affinché sia garantita la proprietà di efficienza ovvero:

$$\sum_{i \in N} x_i = \phi(N)$$

Tuttavia, dato un gioco di coalizione, possono esservi diverse tipologie di soluzioni come il valore Shapley, il nucleo, il kernel, e il nucleolo. Ognuna di queste definisce una serie proprietà desiderabili per il gioco in questione. Il concetto di soluzione che meglio si adatta al gioco della valutazione della ricerca italiana è il concetto di soluzione del valore di Shapley

$$\phi(G) = \sum_{C \subseteq N} \frac{(|N| - |C|)(|C| - 1)!}{|N|!} \pi(C) - \pi(C \setminus \{i\})) \text{ per ogni } i \in N \tag{3.14}$$

In [20] è stato mostrato che il valore di Shapley ha tutte le proprietà desiderabili che sono state elencate.

A questo punto il problema della valutazione della ricerca VQR può essere naturalmente modellato come un gioco di allocazione $A = \langle \mathscr{R}, \mathscr{P}, product, val, k \rangle$ dove $\mathscr{R}$ è l'insieme dei ricercatori affiliati ad una università o a una struttura di ricerca R, P è l'insieme di pubblicazioni della struttura R da sottomettere per la valutazione, $product$ è una funzione che assegna a ciascun autore un insieme di pubblicazioni e infine la funzione val è una funzione che assegna un valore ad ogni pubblicazione. Per il programma di valutazione corrente, la funzione val assegna ad ogni pubblicazione un valore compreso nell'insieme $\{0, 0.1, 0.4, 0.7, 1\}$, dove 1 è riservato alle pubblicazioni ottime. Il valore k di prodotti da considerare per ciascun autore è stato 3 nel primo esercizio e 2 nel secondo. Nel resto di questo capitolo useremo per gli esempi $k = 3$, mentre i dati del caso di studio reale che vedremo nel prosieguo fanno riferimento all'ultima VQR e sarà quindi usato $k = 2$.

Come già anticipato nella fase di presentazione sono gli autori a stimare la qualità dei loro prodotti di ricerca basandosi sui criteri pubblicati dall'ANVUR, questo implica che i valori delle singole pubblicazioni possono subire delle variazioni di punteggio quando vengono sottoposte a revisione dagli esperti dell'ANVUR. Come già anticipato questo è comunque irrilevante per questo lavoro, in quanto si assume che i valori proposti dalle strutture nella fase preliminare coincidono con quelli degli esperti dell'ANVUR, per tutti i prodotti.

Il primo step per risolvere il gioco proposto consiste nel risolvere un problema combinatorio, ovvero un problema di matching, volto ad individuare la migliore allocazione per la struttura R. Il problema combinatorio prevede di selezionare un insieme di pubblicazioni P da presentare, il cui valore complessivo è il massimo possibile tra tutte le possibili combinazioni di pubblicazioni prodotte dalla struttura R nel periodo considerato.

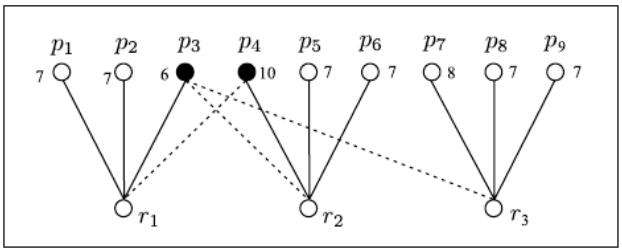


Figura 3.2: Autori e prodotti dell'esempio 3.2.

Esempio 3.2. Si consideri il grafo bipartito della figura 3.2, i cui vertici rappresentano i ricercatori $R = \{r_1, r_2, r_3\}$ di un'università R e P tutte le pubblicazioni da loro prodotte. Gli archi del grafo codificano la relazione paternità, e i loro pesi codificano i valori pubblicazioni. Si consideri l'assegnazione ψ tale che $\psi(r_1) = \{p_1, p_3\}$, $\psi(r_2) = \{p_2, p_4\}$ e $\psi(r_3) = \{p_6, p_7\}$, codificata dagli archi più spessi presenti nella figura. L'insieme delle

pubblicazioni presentate per la valutazione è $P_\psi = \{p_1, p_2, p_3, p_4, p_6, p_7\}$. Si noti che p_2 è co-autore di r_1, r_2 e r_3, mentre p_3 è co-autore di r_1 e r_2. Analizzando i valori di ciascun prodotto, si ottiene il valore ottimo per la struttura R è pari a $\texttt{opt}(R) = 45$.

Come già anticipato la soluzione al problema della valutazione della ricerca italiana si basa sul calcolo del valore di Shapley sul gioco G_ψ:

$$\gamma^*_\psi(r) = \sum_{S \subseteq R} \frac{(|R| - |S|)(|S| - 1)!}{|R|!} marg(r, S) \text{ per ogni } r \in R \tag{3.15}$$

Nel processo di valutazione della ricerca la fase di selezione preliminare può essere vista come una fase in cui vengono recuperate le informazioni sui prodotti di ricerca. Terminata questa fase la struttura R può avere un'allocazione ottima ψ^*, che massimizza il punteggio della struttura

$$\sum_{r \ in S} \sum_{p \in \psi'_S(r)} score_{VQR}(p) \geq \sum_{r \in S} \sum_{p \in \psi_S(r)} score_{VQR}(p) \tag{3.16}$$

per ogni possibile assegnazione ψ.

Si noti che il problema di ottimizzazione si basa sui punteggi dichiarati dai ricercatori e questo implica che se i ricercatori hanno valutato correttamente i loro prodotti, allora la struttura riceverà come punteggio quanto calcolato.

Esempio 3.3. Continuando l'esempio precedente, è ora possibile calcolare le allocazioni ottime per tutte le possibili coalizioni:

- $best_\psi(\{r_1\}) = 24$, $best_\psi(\{r_2\}) = 24$, e $best_\psi(\{r_3\}) = 22$; in questo caso si considera che la struttura è composta solo da r_1 o solo da r_2 o solo da r_3, ed il valore della struttura R è pari al $best_\psi(r_i)$ $i \in \{1,2,3\}$, poiché l'autore può scegliere tutti i migliori prodotti, ivi compresi quelli condivisi con altri autori.

- $best_\psi(\{r_1, r_2\}) = 44$, $best_\psi(\{r_1, r_3\}) = 46$, e $best_\psi(\{r_2, r_3\}) = 46$; nel caso in cui la struttura sia composta da solo due ricercatori si può osservare che la pubblicazione eccellente p_4 è condivisa tra r_1 e r_2. Tuttavia tale pubblicazione al momento della ripartizione $best_\psi(r_1, r_2)$ verrà assegnata ad un solo autore.

- $best_\psi(\{r_1, r_2, r_3\}) = 66$. Nel caso in cui la struttura sia composta da $\{r_1, r_2, r_3\}$, come nel caso precedente, la pubblicazione p_4 verrà assegnata ad uno solo degli autori e gli altri saranno costretti a dividersi le restanti pubblicazioni non eccellenti e la struttura otterrà un valore complessivo di 20 + 24 = 44.

sempre per le stesse coalizioni è possibile calcolare i loro contributi marginali rispetto a tutte le possibili coalizioni di R :

- $marg_\psi(\{r_1\}, R) = 66 - 46 = 20$;
- $marg_\psi(\{r_2\}, R) = 66 - 46 = 20$;
- $marg_\psi(\{r_3\}, R) = 66 - 44 = 22$
- $marg_\psi(\{r_1, r_2\}, R) = 66 - 22 = 44$;
- $marg_\psi(\{r_1, r_3\}, R) = marg_\psi(\{\}r_2, r_3, R) = 66 - 24 = 42$
- $marg_\phi(\{r_1\}, \{r_1, r_2\}) = marg_\psi(\{r_2\}, \{r_1, r_2\}) = 44 - 24 = 20$
- $marg_\phi(\{r_1\}, \{r_1, r_3\}) = marg_\psi(\{r_2\}, \{r_2, r_3\}) = 46 - 22 = 24$;
- $marg_\phi(\{r_3\}, \{r_1, r_3\}) = marg_\psi(\{r_3\}, \{r_2, r_3\}) = 46 - 22 = 24$;

Come si può osservare i primi tre punti permettono di determinare quanto un singolo ricercatore apporta in termini di punteggio al valore della struttura R. Si osserva che per come è strutturato il grafo, se r_3 non venisse considerato la struttura R perderà il valore dei prodotti migliori di questo ricercatore. Inoltre visto che

$$\begin{aligned} marg_\psi(r_2, R) = marg_\psi(r_1, R) = \\ marg_\psi(\{r_1\}, \{r_1, r_2\}) = \\ marg_\psi(\{r_2\}, \{r_1, r_2\}) = 20 \end{aligned}$$

il contributo marginale per ciascuno di questi due giocatori è inferiore al valore dei prodotti a loro assegnati. Questo perché, se r_2 non era parte della struttura, r_1 avrebbe potuto utilizzare anche i migliori prodotti di cui lui è coautore, e viceversa. Infine, si osservi che nell'esempio in questione r_1 e r_2 sono completamente intercambiabili.

Esempio 3.4. *Si consideri lo scenario illustrato in 3.3 e si supponga lo scenario in cui siano presenti solo i ricercatori r_1 e r_2, affiliati alla struttura R, e che $product(r_1) = \{p_1, ..., p_5\}$ e $product(r_2) = p_4, ..., p_8$, r_1 e r_2 siano entrambi coautori dei prodotti p_4 e p_5. Per ognuno dei prodotti $p_i \in product(R1)$ e $p_2 \in product(r_2)$ è definito un punteggio $score_{r_1}(p_i)$ e $score_{r_2}(p_i)$ come raffigurato in figura. In questo scenario i ricercatori r_1, e r_2 agiscono come segue:*

- *r_1 dichiara che i sui prodotti sono $\{p_1, p_2, p_3, p_4\}$ ed hanno come punteggio rispettivamente $score_{r_1}(p_2) = score_{r_1}(p_3) = 7$ e $score_{r_1}(p_1) = 10$ e $score_{r_1}(p_4) = score_{r_1}(p_5) = 8$.*
- *r_2 dichiara che i sui prodotti sono $\{p_4, p_5, p_6, p_6, p_8\}$ e hanno un punteggio pari a $score_{r_2}(p_6) = 7$, $score_{r_2}(p_7) = 8$, $score_{r_2}(p_8) = 10$.*

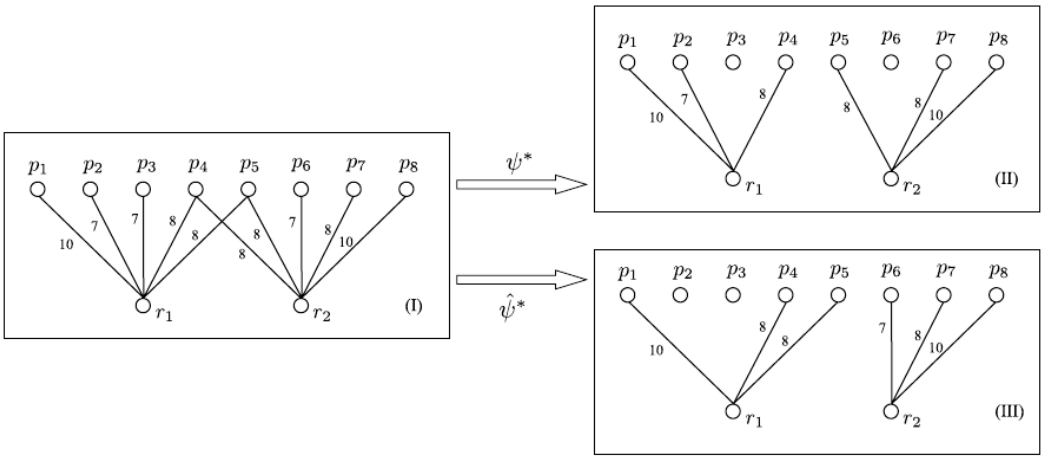


Figura 3.3: Autori e prodotti dell'esempio 3.4.

In questo scenario si evince che le ripartizioni ottime per R e ψ^* sono:

- $\psi^*(r_1) = \{p_1, p_2, p_4\}$ *e* $\psi^*(r_2) = \{p_5, p_7, p_8\}$ *di cui* $score_\psi(r_1) = 25$ *e* $score_\psi(r_2) = 26$
- $\psi'^*(r_1) = \{p_1, p_5, p_4\}$ *e* $\psi'^*(r_2) = \{p_6, p_7, p_8\}$ *di cui* $score_\psi(r_1) = 26$ *e* $score_\psi(r_2) = 25$

Complessivamente si evince che le allocazioni ottime per la struttura sono:

- $\psi^*(R) = \{p_1, p_2, p_4, p_5, p_7, p_8\}$
- $\psi'^*(R) = \{p_1, p_5, p_4, p_6, p_7, p_8\}$

In base queste ipotesi la struttura R nel complesso calcola un'allocazione ottima il cui punteggio è pari a $score'^*_\psi(R) = 51$

Dall'esempio precedente si evince che per la struttura R esistono diverse possibili selezioni ottime dei prodotti da sottomettere, che sono indistinguibili per R. Le diverse scelte possono però avere conseguenze importanti per i ricercatori (o per gruppi e sotto-strutture), che possono trovarsi nella situazione in cui una specifica selezione li avvantaggia o nel caso peggiore li svantaggia. Una regola di divisione davvero equa dovrebbe quindi anche essere indipendente dalla specifica allocazione ottima per R in base alla quale sono stati scelti i prodotti da sottomettere.

P5 indipendenza della selezione prodotti: Siano ψ_1, ψ_2 una qualsiasi coppia di allocazioni ottime per R, e γ una regola divisione. Per ogni $r \in R$ deve risultare valida la sequente proprietà

$$\gamma_{\psi_1}(r) = \gamma_{\psi_2}(r)$$

Anche questa proprietà è soddisfatta dal valore di Shapley [20].

In tale lavoro sono anche discussi alcuni elementi strategico-opportunistici dei ricercatori che, in presenza di regole di divisione ingiuste, potrebbero portarli ad alterare le scelte dell'Università per favorire la propria valutazione (o del proprio gruppo), col possibile effetto di mancare la scelta ottima globale e quindi danneggiare tutta la struttura.

Capitolo 4

Limiti inferiori e superiori e proprietà del valore di Shapley

Si ricorda che il calcolo del valore di Shapley è un problema #P-Hard per molte classi di giochi [23, 22, 24, 29], compresi quelli di allocazione anche se i beni hanno solo due possibili valori [21]. In problemi di grosse dimensioni un approccio "brute-force" non è praticamente utilizzabile, poiché avrebbe bisogno di risolvere, per determinare il valore di ciascun agente $i \in N$, 2^n problemi di ottimizzazione, dove $n = |N|$ è il numero di agenti. Ciò avviene in particolare nel caso di studio considerato, in cui occorre calcolare il valore di Shapley anche per i ricercatori di grandi università o centri di ricerca per i quali n è dell'ordine delle migliaia, così come il numero di beni (qui pubblicazioni) da allocare.

Per mitigare la complessità del problema, in questo capitolo dimostreremo alcune proprietà del valore di Shapley per i giochi di allocazione che ci permettono di semplificare le istanze in una fase di pre-processing. Inoltre introduciamo un algoritmo per il calcolo di un limite inferiore ed un limite superiore per il valore di Shapley. Tali limiti possono essere sfruttati negli algoritmi esatti per il calcolo del valore di Shapley o anche per valutare in modo più preciso la qualità dell'output di algoritmi di approssimazione.

4.1 Proprietà utili per affrontare problemi reali

In questa sezione descriviamo alcune proprietà del valore di Shapley in giochi di allocazione per i quali assumiamo che i beni abbiano tutti valori non-negativi. In questo tipo di giochi il contributo marginale di ciascun giocatore è sempre non-negativo e quindi ciò vale anche per il valore di Shapley. Nel prosieguo della tesi manteniamo implicitamente questa assunzione di non-negatività dei beni.

Definizione 4.1. *Dato uno scenario di allocazione* $A = \langle N, \{G\}, \omega, \texttt{val}, k\rangle$*, il grafo degli agenti è un grafo non orientato* $G(A) = (N, E)$ *dove i nodi corrispondono agli*

agenti del gioco ed ogni arco rappresenta una relazione di collaborazione tra di essi. Più precisamente, $E = \{\{i, j\} \mid \omega(i) \cap \omega(j) \neq \emptyset\}$.

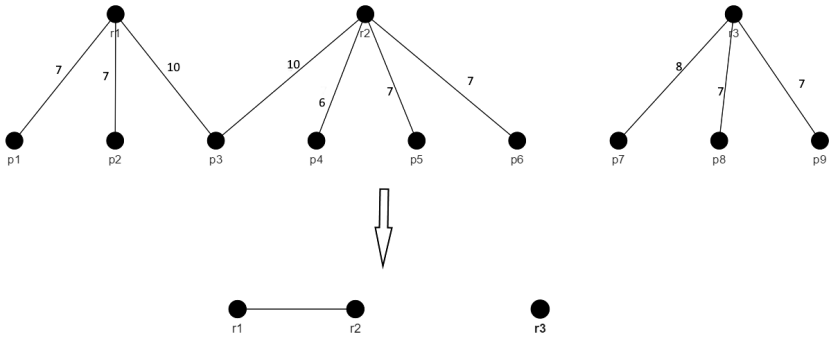


Figura 4.1: Grafo degli agenti

Si noti che, nel caso specifico, il grafo degli agenti rappresenta il grafo delle collaborazioni tra i ricercatori di una struttura R. I giochi di allocazione generalmente sono definiti su un grafo bipartito agenti/prodotti come raffigurato nella figura 4.1. Dato un grafo bipartito agenti/prodotti si costruisce il grafo degli agenti che ne sintetizza le relazioni. Nella figura 4.1 ci sono tre agenti $N = \{r_1, r_2, r_3\}$, gli agenti r_1 e r_2 presentano un bene condiviso p_3 mentre r_3 non presenta nessun bene condiviso. Nel grafo degli agenti costruito per questo scenario vi sono nodi che rappresentano r_1 e r_2 collegati da un arco, mentre r_3 rappresenta un nodo disconnesso.

Si consideri un gioco di allocazione $G = \langle N, v\rangle$, in cui il grafo degli agenti è $G = (N, E)$; per tale scenario mostriamo le seguenti proprietà che permettono di semplificare il gioco senza alterare il valore di Shapley di nessun giocatore: *Modularità*, *Agenti disconnessi*, *Separabilità*, *Beni di valore nullo*.

4.1.1 Modularità

Sia $\{C_1, C_2\}$ una partizione degli agenti di N tale che non esiste nel grafo degli agenti nessun arco tra qualche agente in C_1 e qualche agente in C_2. Siano rispettivamente $G_1 = \langle C_1, v_1\rangle$ e $G_2 = \langle C_2, v_2\rangle$ i giochi di coalizione ristretti agli agenti della componente C_1 e della componente C_2. Allora per il valore di Shapley di ogni agente $i \in N$ si ha $sv_i(G) = sv_i(G_1) + sv_i(G_2)$.

Dimostrazione. Siano $G'_1 = \langle N, v'_1\rangle$ e $G'_2 = \langle N, v'_2\rangle$ i due giochi di coalizione tali che, per ogni $C \subseteq N$, $v'_1(C) = v_1(C \cap C_1)$ e $v'_2(C) = v_2(C \cap C_2)$. Poiché gli agenti di C_1

e C_2 non interagiscono tra di loro, il valore totale dell'allocazione ottima dei beni a C è data dalla somma delle allocazioni ottime ristrette ai due insiemi $C \cap C_1$ e $C \cap C_2$. Si ha quindi $v(C) = v'_1(C) + v'_2(C)$.

Per la proprietà di additività del valore di Shapley, per ogni $i \in N$, $sv_i(G) = sv_i(G'_1) + sv_i(G'_2)$. Si noti inoltre che ogni giocatore in $N \setminus C_1$ è *dummy* per il gioco G'_1 e può essere eliminato poiché avrà un valore di Shapley nullo e non influenzerà il valore di nessun altro giocatore. Di conseguenza, per ogni giocatore $i \in C_1$, si ha $sv_i(G'_1) = sv_i(G_1)$ ed il risultato segue immediatamente (ovviamente il ragionamento è identico per G_2). □

Esempio 4.1. *Si consideri la figura 4.1 e si consideri uno scenario in cui ogni autore può sottoporre al massimo due prodotti*

- $marg_\psi(\{r_1\}, R) = 44 - 17 = 27$*;*
- $marg_\psi(\{r_2\}, R) = 44 - 17 = 27$*;*
- $marg_\psi(\{r_3\}, R) = 44 - 13 = 31$
- $marg_\psi(\{r_1, r_2\}, R) = 44 - 17 = 27$*;*
- $marg_\psi(\{r_1, r_3\}, R) = 44 - (17 + 13) = 27$
- $marg_\psi(\{r_2, r_3\}, R) = 44 - (17 + 13) = 30$

Si osservi che il contributo marginale che r_3 apporta ad ogni possibile coalizione è sempre pari alla sua allocazione ottima $best(r_3) = 13$ poiché r_3 non condivide nessun bene con i restanti agenti. Per tale motivo il suo valore di Shapley è pari alla sua allocazione ottima ovvero 17.

Il caso riportato nell'esempio è legato ad un solo autore ma può essere esteso ad una coalizione isolata, come descritto precedentemente. La modularità è un'importante proprietà del valore di Shapley in quando permette di dividere il problema di partenza in diversi sotto problemi, uno per ciascuna componente connessa del grafo agente/agente, risolvibili in modo separato. In particolare eventuali piccoli gruppi isolati possono essere risolti facilmente anche con l'algoritmo *brute force*.

4.1.2 Beni di valore nullo

È facile vedere che i beni di valore nullo non impattano sul calcolo dell'allocazione ottima (avendo assunto di considerare giochi con beni non negativi). Tuttavia la presenza di beni di valore nullo condivisi tra più agenti induce connessioni tra gli agenti e quindi complica la struttura del grafo.

Si consideri ad esempio uno scenario di allocazione in cui vi sono tre agenti $N = \{r_1, r_2, r_3\}$ in cui ognuno di questi ha un insieme di beni di cui l'unico condiviso tra tutti

ha peso nullo. In tale scenario, la proprietà di *Modularità* non è applicabile, in quanto risulta che il grafo delle interazioni agente/agente presenta un'unica componente connessa. Se tale bene non fosse condiviso il grafo degli agenti risulterebbe invece completamente disconnesso e sarebbe quindi possibile calcolare banalmente i valori di Shapley applicando la proprietà di *Modularità*.

Per tali situazioni, è tuttavia possibile sfruttare la seguente proprietà: sia A uno scenario di allocazione $\langle N, \{G\}, \omega, \texttt{val}, k\rangle$. Sia g un bene con $\texttt{val}(g) = 0$ e sia $X = \{a \in N \mid g \in \omega(a)\}$ l'insieme degli agenti interessati a g. Allora A è equivalente (rispetto al valore di Shapley) allo scenario di allocazione in cui g è sostituito da $|X|$ beni $g_1, \ldots g_{|X|}$ tali che ognuno di essi è di interesse di un unico agente in X (non vi sono quindi connessioni nel grafo a causa di tali beni).

Per vedere che la proprietà vale, è sufficiente ricordare che, per gli scenari considerati in cui non vi sono beni aventi valore negativo, nel calcolo del contributo marginale di un qualche agente i ad una coalizione C, non vi è alcuna convenienza per C nell'usare un bene di valore nullo in $\omega(i)$.

4.1.3 Separabilità

La seguente proprietà ci offre un potente strumento di semplificazione per giochi di allocazione. Intuitivamente essa afferma che qualunque insieme di agenti C che non mostri sinergie effettive col resto degli agenti può essere eliminata dal gioco e risolta separatamente.

Teorema 4.1.1 (separabilità)**.** *Data una coalizione C tale che $opt(C) + opt(N \setminus C) \leq opt(N)$, è possibile definire due diversi scenari di allocazione rispettivamente per le coalizioni C ed $N \setminus C$ risolvibili separatamente. Per ogni giocatore $i \in N$ è quindi possibile calcolare il suo valore di Shapley considerando solo il gioco ristretto all'insieme in cui i occorre.*

Dimostrazione. Sia $\overline{C} = N \setminus C$ e si considerino i giochi $G_1 = \langle C, V_1\rangle$ e $G_2 = \langle \overline{C}, V_2\rangle$ limitati rispettivamente agli agenti di C e di $\overline{C}$. Dal momento che i giochi di allocazione sono supermodulari (si veda ad esempio [21]), in generale si ha $opt(C) + opt(\overline{C}) \geq opt(N)$. Combinata con l'ipotesi riguardo la coalizione C, tale espressione implica $opt(C) + opt(\overline{C}) = opt(N)$, cioè la somma dei valori dei beni non utilizzati in una (qualunque) allocazione ottima di $\overline{C}$ è uguale alla somma dei valori dei migliori beni selezionati per gli agenti della coalizione C.

Mostriamo di seguito che, per ogni allocazione ottima π per N, l'insieme di beni $S \subseteq \omega(C)$ assegnato da π a C è tale che $val(S) = opt(C)$ e l'analoga proprietà vale per $\overline{C}$. Tali insiemi ottengono cioè i migliori beni a cui possono aspirare. Per dimostrare tale affermazione, si consideri il valore $v = val(S) \leq opt(C)$ ed il valore $\overline{v} \leq opt(\overline{C})$. Per quanto sopra osservato, si ha $opt(C) + opt(N \setminus C) = opt(N)$ e, poiché π è ottima, vale anche $v + \overline{v} = opt(N)$.

Si consideri ora una qualsiasi coalizione $C' \subseteq N$, e siano $C_a = C' \cap C$ e $C_b = C' \cap \overline{C}$. Consideriamo un'allocazione ottima π' per C'. È possibile affermare che esiste una allocazione ottima π_a che mappa i beni da S a C dove $val_{\pi_a}(C_a) = val_{\pi'}(C_a)$, ed una allocazione ottima di beni π_b che mappa i beni non in S a $\overline{C}$,con $val_{\pi_b}(C_b) = val_{\pi'}(C_b)$. Assumiamo per contraddizione che quanto affermato non sia vero. In tal caso almeno una di queste allocazioni presenta un valore più piccolo di π' (si noti infatti che l'allocazione π' non può essere peggiore, in quanto l'unione delle due allocazioni limitate costituisce una mappatura ammissibile per C'). Si supponga che C_a ottenga un valore complessivo più piccolo (l'altro caso è simmetrico), vale a dire, $val_{\pi_a}(C_a) < val_{\pi'}(C_a)$. Allora esiste un qualche agente i ed un bene $p \notin S$ tale che $p \in \pi'(i)$. Utilizzando il teorema 4.4 del lavoro [21] si può concludere che ciò sarebbe in contraddizione con il fatto che $val(S) = opt(C)$. Infatti, beni come p che sono condivisi con gli agenti esterni alla coalizione C e che permettono di ottenere un valore migliore per gli agenti di $C_a \subseteq C$, potrebbero essere utilizzati per migliorare la scelta dei beni disponibili di S per l'insieme C.

A questo punto, visto che è sufficiente usare solo i beni di S per C ed i restanti per $\overline{C}$, è possibile definire un gioco equivalente nel quale i prodotti presenti in S siano di interesse solo per gli agenti in C ed i restanti solo per gli agenti in $\overline{C}$. Nel nuovo gioco C ed $\overline{C}$ sono pertanto insiemi di agenti senza connessioni in comune ed il teorema segue immediatamente dalla proprietà di modularità sopra descritta. □

Un caso frequente e importante nelle applicazioni è quello delle coalizioni di tipo singoletto che ricadono nel caso considerato da quest'ultima proprietà. In questi casi accade cioè che l'allocazione ottima per tale coalizione, diciamo $\{i\}$, coincide con il contributo marginale di i ad $N \setminus \{i\}$. Per la proprietà sopra enunciata l'insieme $\{i\}$ può quindi essere eliminato e risolto separatamente e si ha immediatamente $\phi(i) = opt(\{i\})$.

4.1.4 Agenti disconnessi

Concludiamo la sezione con una semplice proprietà che non serve a semplificare il grafo, ma solo ad evitare il calcolo di allocazioni ottime inutili nell'ambito della valutazione dei contributi marginali.

Si considerino il grafo degli agenti $G(A) = (N, E)$, un agente $i \in N$ ed una coalizione $C \subseteq N$ tale che i non abbia alcun bene condiviso con i membri della coalizione C, cioè tale che $Neigh(i) \cap C = \emptyset$. Vale allora $opt(\{i\} \cup C) = opt(\{i\}) + opt(C)$ e quindi il contributo marginale di i a C è semplicemente $opt(\{i\})$.

4.2 Lower bound e upper bound per il valore di Shapley

Anche calcolare solo una approssimazione accettabile del valore di Shapley può risultare un'operazione molto onerosa se la dimensione dell'istanza del problema è molto grande.

In questa sezione si descrive un algoritmo per determinare due bound (Upper e Lower) di un dato gioco di allocazione $G_A = \langle N, v_A \rangle$. La disponibilità di tali bound può essere di aiuto a quantificare meglio il vero livello di errore di un algoritmo approssimato (come vedremo spesso tali algoritmi funzionano meglio rispetto ai loro bound teorici), a ridurre il numero di iterazioni di tali algoritmi, ma può anche permettere di calcolare esattamente il valore di Shapley per quegli agenti per i quali i due bound coincidano.

Preliminarmente si ricordi che due bound molto semplici possono essere facilmente ottenuti analizzando la definizione del valore di Shapley per ogni giocatore i. Infatti è possibile osservare che tale valore è sempre compreso nell'intervallo $[marg(\{i\}, N), \texttt{opt}(\{i\})]$. Per ottenere bound più ristretti occorre considerare coalizioni un po' più ampie del semplice singoletto. A tal proposito si osservi che, intuitivamente, i vicini di i in una coalizione C sono gli agenti che hanno un'influenza maggiore sul contributo marginale di i nella coalizione C. Sono infatti proprio questi agenti i primi interessati a utilizzare i beni di i quando si cerca un'allocazione ottima per C (senza i). L'idea è quindi di calcolare un valore in cui, per ogni $i \in N$, ci si basa solo sugli effetti determinati dalla presenza dei suoi vicini (in tutte le loro combinazioni) per stimare i valori di tutte le coalizioni senza restrizioni.

Sia P' un sottoinsieme di vicini di i e siano

$$\begin{aligned} Z &= Neigh(i) \setminus P' \\ C &= N \setminus (Neigh(i) \cup \{i\}) \\ l &= |C| \end{aligned} \tag{4.1}$$

Il principio alla base dell'algoritmo è quello di calcolare il contributo marginale di i alla coalizione $C \cup P'$, per ogni tale sottoinsieme P', e di utilizzare questo valore al posto dei contributi marginali di i a ciascuna coalizione $C' \subseteq N$ tale che $C' \cap Neigh(i) = P'$, cioè a ciascuna coalizione con la stessa configurazione P' di vicini di i. Si utilizza poi un opportuno fattore moltiplicativo y per pesare adeguatamente tale valore come se provenisse da tutte le coalizioni avente tale forma C'.

Lo stesso ragionamento di applica dualmente al limite superiore, ma considerando il caso più favorevole in cui si usa il contributo marginale di i a P' al posto dei contributi marginali di i a ciascuna coalizione $C' \subseteq N$ tale che $C' \cap Neigh(i) = P'$.

Di seguito è riportato l'algoritmo per il calcolo del limite inferiore LB_i e il limite superiore UB_i del valore di Shapley per ogni agente i.

Teorema 4.2.1. *Sia (LB, UB) la coppia di vettori restituita dall'Algoritmo 1. Per ogni agente $i \in N$, si ha $LB_i \leq sv(i) \leq UB_i$ ed il calcolo di tali valori ha complessità $O(2^{|Neigh(i)|}|N|^3)$.*

Dimostrazione. L'algoritmo si basa sul calcolo di ogni possibile combinazione P' dei vicini di ciascun agente i nel grafo degli agenti.

Algorithm 1 Calcolo dei bound per il valore di Shapley per giochi di allocazione

Input: Un gioco di allocazione $\mathsf{G}_{\mathscr{A}} = \langle N, v_{\mathscr{A}} \rangle$;
Output: Una coppia di vettori (LB, UB) che codificano, rispettivamente, un limite inferiore e un limite superiore del valore di Shapley di $\mathsf{G}_{\mathscr{A}}$;

1: **for all** $i \in N$ **do**
2: $P := Powerset(Neigh(i))$;
3: $C := N \setminus (Neigh(i) \cup \{i\})$;
4: $l = |C|$;
5: **for all** $P' \in P$ **do**
6: $Z := Neigh(i) \setminus P'$;
7: $y = \sum_{k=0}^{l} \frac{(l-k+|P'|)! \cdot (|Z|+k)!}{|N|!} \cdot \binom{l}{k}$;
8: $LB_i \mathrel{+}= y \cdot (v_{\mathscr{A}}(C \cup P' \cup \{i\}) - v_{\mathscr{A}}(C \cup P'))$;
9: $UB_i \mathrel{+}= y \cdot (v_{\mathscr{A}}(P' \cup \{i\}) - v_{\mathscr{A}}(P'))$;
10: **end for**
11: **end for**
12: **return** (LB, UB);

Per quanto riguarda il calcolo del lower bound, per ogni tale profilo P', l'algoritmo considera una coalizione $C \cup P'$ ottenuta completando P' con tutti gli altri agenti in $N \setminus \{i\}$ che non sono vicini di i. L'algoritmo usa poi il valore del contributo marginale di i a tale coalizione, cioè il valore $\delta = v_{\mathscr{A}}(C \cup P' \cup \{i\}) - v_{\mathscr{A}}(C \cup P')$, al posto dei contributi marginali di i a ciascuna coalizione $C' \subseteq N$ tale che $C' \cap Neigh(i) = P'$. Poiché $C' \subseteq C$, per la proprietà di antimonotonicità dei contributi marginali nei giochi di allocazione, si ha infatti immediatamente $marg(\{i\}, C) \leq marg(\{i\}, C')$. L'algoritmo pesa poi in modo opportuno tale valore δ in modo che sia considerato nel calcolo del limite inferiore al posto dei valori che avremmo dovuto calcolare per ogni coalizione C' della forma sopra descritta. Un semplice calcolo combinatorio mostra che ciò può essere ottenuto moltiplicando δ per il seguente fattore moltiplicativo

$$y = \sum_{k=0}^{l} \frac{(l-k+|P'|)! \cdot (|Z|+k)!}{|N|!} \cdot \binom{l}{k}, \tag{4.2}$$

dove $l = |N \setminus (Neigh(i) \cup \{i\})|$.

Per quanto riguarda il limite superiore del valore di Shapley di i si procede in modo analogo, ma considerando, in luogo dei contributi marginali alle varie coalizioni $C' \subseteq N$ tali che $C' \cap Neigh(i) = P'$, il solo contributo marginale di i al profilo P' contenente solo agenti suoi vicini. In questo caso abbiamo quindi $P' \subseteq C'$ e quindi $marg(\{i\}, C') \leq marg(\{i\}, P')$. Anche in questo caso occorre moltiplicare tale valore per un fattore che tenga conto di tutti i possibili modi di completare P' ad una qualche coalizione C' con lo stesso profilo di vicini di i. È semplice vedere che può essere usato allo scopo lo

stesso fattore y sopra descritto, sfruttando il fatto che, per i coefficienti binomiali, vale la seguente relazione:

$$\binom{l}{k} = \binom{l}{l-k} \tag{4.3}$$

Per quanto riguarda la complessità, è immediato osservare che per ogni elemento P' del power set di $Neigh(i)$ occorre risolvere un numero costante di problemi di allocazione ottima. Ognuno di tali problemi equivale alla soluzione di un classico problema di matching pesato di valore massimo, che può essere risolto in $O(N^3)$. □

L'analisi di complessità mostra che il calcolo dei bound sopra descritti è esponenziale nel degree massimo del grafo degli autori. Come vedremo nei capitoli con gli esperimenti, tali valori sono comunque calcolabili quasi sempre in modo efficiente, soprattutto grazie alle fasi di preprocessamento delle istanze, basate sulle tecniche di semplificazione descritte in questo stesso capitolo.

Capitolo 5

Approssimazione del valore di Shapley nei giochi di allocazione

Nei precedenti capitoli è stato osservato più volte che in applicazioni reali, come il caso di studio considerato, si ha a che fare con un numero di giocatori dell'ordine delle migliaia. In questi casi il problema del calcolo del valore di Shapley è intrattabile e dunque un approccio "brute-force" non è praticamente utilizzabile. Motivati da queste brutte notizie, in questo capitolo presenteremo una tecnica di approssimazione (che sfrutta alcune delle proprietà discusse nel precedente capitolo) per il calcolo del valore di Shapley. La tecnica proposta è una lieve variazione dell'algoritmo proposto in [25], in cui si usano alcune peculiarità dei problemi di allocazione, che abbiamo provato nel precedente capitolo. Un ulteriore contributo, importante per le applicazioni concrete, è una stima più accurata di valori attesi e conseguentemente di iterazioni da effettuare, sulla base dei risultati descritti in [60].

5.1 Schema di approssimazione randomizzato

In [25] è stato descritto una schema di approssimazione randomizzato: per ogni $\epsilon > 0$ e $\delta > 0$, è possibile calcolare in tempo polinomiale una $\epsilon-$approssimazione del valore di Shapley con probabilità di fallimento al più δ. La tecnica proposta è applicabile solo su giochi di coalizione supermodulari e monotoni. Si può dimostrare che i nostri giochi di allocazione rispettano queste proprietà [54]. Il metodo è basato sulla generazione di un certo numero di permutazioni (di tutti gli agenti) e sul calcolo dei contributi marginali di ogni agente rispetto alla coalizione di agenti che occorrono prima di lui nella permutazione considerata. Quindi il valore di Shapley di ogni giocatore è calcolato come la media di tutti questi contributi marginali. Chiaramente, maggiore sarà il numero permutazioni considerate più vicino sarà il risultato al valore di Shapley. Di seguito riportiamo una

versione lievemente modificata della procedura di base di questo algoritmo, in cui in particolare eviteremo il calcolo di alcuni contributi marginali. Si noti che tale risultato deriva dall'applicazione della proprietà *Agenti disconnessi* discussa nel precedente capitolo.

Algorithm 2 Calcolo approssimato del valore di Shapley

Input: Un gioco di allocazione $\mathsf{G}_{\mathscr{A}} = \langle N, v_{\mathscr{A}} \rangle$;
Parametri: Numeri reali $1 > \epsilon > 0$ e $1 > \delta > 0$;
Output: Un vettore $\tilde{\phi}$ che rappresenta una ϵ-approssimazione del valore di Shapley di $\mathsf{G}_{\mathscr{A}}$, con probabilità $1 - \delta$;

1: $m = \frac{|N| \cdot (|N|-1)}{\delta \cdot \epsilon^2}$;
2: $i = 0$;
3: **while** $i < m$ **do**
4: $\quad$ *shuffle*(N);
5: $\quad C := \{\emptyset\}$;
6: $\quad$ **for all** $j \in N$ **do**
7: $\quad\quad$ **if** $Neigh(j) \cap C \neq \emptyset$ **then**
8: $\quad\quad\quad \tilde{\phi}_j \mathrel{+}= v_{\mathscr{A}}(C \cup \{j\}) - v_{\mathscr{A}}(C)$;
9: $\quad\quad$ **else**
10: $\quad\quad\quad \tilde{\phi}_j \mathrel{+}= v_{\mathscr{A}}(\{j\})$;
11: $\quad\quad$ **end if**
12: $\quad\quad C := C \cup \{j\}$;
13: $\quad\quad i = i + 1$;
14: $\quad$ **end for**
15: **end while**
16: **for all** $j \in N$ **do**
17: $\quad \tilde{\phi}_j = \frac{\tilde{\phi}_j}{m}$;
18: **end for**
19: **return** $\tilde{\phi}$;

L'algoritmo procede come segue. In un passo preliminare viene determinato il numero di permutazioni m che garantisce l'errore atteso richiesto. In ognuna di queste m iterazioni, l'algoritmo genera una permutazione casuale a partire dall'insieme degli agenti N. Successivamente per ciascun agente della permutazione è calcolato il contributo marginale di ogni angente j all'insieme degli agenti C che occorrono prima di j nella permutazione considerata. Se nel grafo degli agenti esiste un vicino di j che occorre in C, allora l'algoritmo procede come al solito risolvendo il problema di ottimizzazione per calcolare $v_{\mathscr{A}}(C \cup \{j\})$ (i.e., per ottenere il valore $\texttt{opt}(C \cup \{j\})$ di un'allocazione ottima per $C \cup \{j\}$). Si noti che per ottenere il contributo marginale desiderato abbiamo bisogno di risolvere un solo problema di allocazione, in quanto il valore $\texttt{opt}(C)$ per tutti gli agenti "predecessori" è noto dal passo precedente. Inoltre, dalla proprietà di *Agenti disconnessi*,

sappiamo che per quelle permutazioni in cui tutti i giocatori in $Neigh(j)$ seguono j, il contributo marginale di j è proprio $\texttt{opt}(\{j\})$ (si veda il passo 10). In conclusione, nei passi 16-18 per ogni agente l'algoritmo divide la somma dei suoi contributi per il numero di iterazioni compiute m. La correttezza dell'intero algoritmo segue dal Teorema 4 in [25].

Analisi di complessità. Sia $n = |N|$ il numero di agenti e m sia il numero di iterazioni. Il costo dell'algoritmo è di $O(m \times n \times margBlock)$, dove $margBlock$ denota il costo necessario per calcolare ogni contributo marginale. Il calcolo dell'allocazione ottima su un grafo bipartito, è un problema noto in letteratura ed è risolvibile con $O(n^3)$ operazioni, mediante l'utilizzo del classico algoritmo ungherese. Tuttavia, se l'agente corrente risulta disconnesso dal resto della coalizione, il costo dell'allocazione è semplicemente dato da una ricerca nella cache in cui è memorizzata la migliore allocazione per ogni singolo agente.

5.2 Un'approssimazione più accurata

Maleki et al. [60] propongono un limite più accurato sul numero di campioni richiesti per stimare il valore di Shapley di un agente, in tutti quei casi in cui è noto l'intervallo (cioè il limite superiore e il limite inferiore) dei suoi contributi marginali. Il loro limite è basato sulla disuguaglianza di Hoeffding [59], e afferma che, per approssimare il valore di Shapley di un agente i entro il valore assoluto ϵ, con probabilità di fallimento al più δ_i, cioè a dire, per poter ottenere

$$Prob\{|\tilde{\phi}_i - \phi_i| \geq \epsilon\} \leq \delta_i \tag{5.1}$$

sono richiesti almeno m_i campioni, dove:

$$m_i = \left\lceil \frac{\ln\left(\frac{2}{\delta_i}\right) \cdot r_i^2}{2 \cdot \epsilon^2} \right\rceil \tag{5.2}$$

Nell'espressione di cui sopra, r_i denota l'intervallo dei contributi marginali di i (i.e., $r_i = \texttt{opt}(\{i\}) - marg(\{i\}, N)$), dove N è un insieme di agenti che partecipa nel gioco di allocazione).

Questo limite ci permette di determinare il numero di campioni richiesti per ogni agente i, una volta che sono stati fissati ϵ e δ. Assumendo di volere una probabilità di fallimento δ, ad ogni agente $i \in N$ potrebbe essere assegnata una probabilità di fallimento $\delta_i = \delta/|N|$, oppure una più alta probabilità di fallimento potrebbe essere tollerata per gli agenti aventi un intervallo di valori (dei propri contributi marginali) più ampio, a scapito di una bassa probabilità di fallimento per gli agenti con un intervallo più piccolo (nei nostri test abbiamo verificato ciò, ottenendolo solo guadagni marginali). Una volta determinato

il numero di campioni richiesti per ogni agente, il valore di Shapley approssimato, entro la garanzia teorica desiderata, può essere calcolato facilmente.

Osserviamo inoltre che, rimpiazzando ϵ con $\epsilon \cdot \phi_i$ in (5.1), otteniamo un errore limite espresso in termini percentuali del valore di Shapley corretto (analogamente al limite fornito dall'Algoritmo 2 descritto nella sezione precedente). Chiaramente, ϕ_i è precisamente quello che stiamo provando a stimare, e tale valore è sconosciuto. Di conseguenza il valore non può essere inserito in (5.2) per determinare il numero corrispondente di campioni richiesti (si noti che anche qui è necessaria l'ulteriore assunzione che $\phi_i \neq 0$). Tuttavia, il valore di ϕ_i che apparirebbe in (5.2) potrebbe essere rimpiazzato da un qualsiasi limite inferiore noto, a scapito di dover prendere più campioni rispetto quelli strettamente necessari per i. Da ciò segue che possiamo facilmente affrontare istanze in cui occorrono agenti il cui valore di Shapley è 0, rimuovendo tali agenti in una fase di preprocessing. Per i restanti agenti, $marg(\{i\}, N)$ è un limite inferiore ovvio, che comunque può essere 0 anche se $\phi_i \neq 0$. Un'alternativa a tale approccio, è data dal nostro algoritmo per il calcolo del limite inferiore, presentato nel Capitolo 4. Per l'istanza più grande considerata nei nostri test, queste tecniche portano ad ottenere dei limiti inferiori che sono maggiori di 0 per tutti gli agenti, permettendoci in definitiva di approssimare il valore di Shapley entro il 5% dei valori corretti in poche ore.

In effetti, a causa della relazione esponenziale tra m_i e δ_i, il teorema di Maleki et al. ci permette di approssimare il valore di Shapley in modo efficiente, almeno per le nostre istanze di test dove l'intervallo tra i valori dei contributi marginali è piccolo. Per confronto, l'approccio FPRAS descritto nella sezione precedente, fissata una certa garanzia per l'errore massimo ammissibile, impiegherebbe qualche anno (invece di qualche ora) nelle nostre istanze di test più grandi.

Capitolo 6

Implementazione e analisi sperimentale di algoritmi per il calcolo dei limiti e delle approssimazioni del valore di Shapley nei giochi di allocazione

6.1 Implementazione

Questa sezione, illustra i risultati delle attività di implementazione degli algoritmi presentatati nel capitolo 5. Nello specifico si analizzano le implementazioni parallele dell'algoritmo di approssimazione basato sull'approccio di Liben-Nowell et al. [25], e dell'algoritmo basato su Maleki et al. [60], viene inoltre riportata una descrizione dell'implementazione dell'algoritmo brute-force utilizzato negli esperimenti.

Approssimazione basata su Liben-Nowell et al. L'implementazione di questo algoritmo richiede in input oltre ad un gioco di allocazione anche due parametri δ e ϵ, a questi se ne aggiunge un terzo che definisce il massimo numero di thread che l'algoritmo può utilizzare nella sua esecuzione. L'algoritmo inizializza preliminarmente tutti i thread e le diversi componenti necessarie per la simulazione. Durante l'esecuzione dell'algoritmo ciascun thread genera un certo numero di permutazioni, in accordo con il fattore di approssimazione. Per tutte le permutazioni, l'algoritmo calcola il contributo marginale di ogni autore. In particolare per ogni autore i il metodo calcola il contributo marginale alla coalizione composta dai soli autori che precedono i nella permutazione in esame e ne salva il valore in una cache locale. Ogni thread, calcolati i contributi marginali di ciascun autore, li invia ad un componente *acceptor* che in modo sincronizzato ne incrementa il punteggio. La procedura viene ripetuta in modo indipendente un certo numero di volte in accordo con il parametro δ. il componente *output acceptor* calcola, per ogni autore, il valore di Shapley approssimato effettuando la media della somma di tutti i contributi

marginali sul numero di campioni richiesti per quell'autore. Il risultato finale è quindi scalato in modo da garantire la proprietà budget-balance e un errore di approssimazione inferiore a ϵ con una probabilità di fallimento δ.

Approssimazione basata su Maleki et al. Come primo passo l'algoritmo calcola con una procedura sequenziale (tale calcolo può essere effettuato efficientemente) il numero di permutazioni da analizzare per ogni singolo autore i, sulla base dei parametri δ e ϵ e sui valori $\texttt{opt}(\{i\})$, $marg(\{i\}, N)$ fornito in input, dove N è l'insieme di tutti gli autori, l'algoritmo richiede inoltre due ulteriori parametri $threadPoolSize$ e $batchSize$. Successivamente, ogni thread (si noti che il numero totale di threads è determinato dal $threadPoolSize$) richiede al componente *producer* sincronizzato, l'assegnazione di un task, quest'ultimo o ne ritorna uno (ovvero una coppia $(author, numSamples)$) se disponibile o ritorna *null* se non vi sono più task da assegnare. Dopo aver ricevuto un task, il thread genera $numSamples$ sottoinsiemi $S \in \{N \setminus author\}$ uniformemente distribuiti e, per ciascun sottoinsieme S, l'algoritmo calcola il contributo marginale dell'autore $author$ alla coalizione S. La somma di questi contributi è inviata ad un componente sincronizzato *output acceptor* che memorizza, per ogni autore, la somma di tutti i contributi marginali calcolati dai diversi threads. Al fine di garantire una corretta redistribuzione del carico di lavoro sui diversi thread, il componente *producer* assegna a ciascun thread un task in cui è richiesta la generazione di un numero di campioni sia minore di $batchSize$ $numSamples \leq batchSize$. Un thread termina la sua esecuzione nel momento in cui, effettuata una richiesta di un nuovo task gli viene ritornato il valore *null*, in quanto non vi sono più task da assegnare. Quando tutti i thread terminano, il componente *output acceptor* calcola, per ogni autore, il valore di Shapley approssimato effettuando la media della somma di tutti i contributi marginali sul numero di campioni richiesti per quell'autore.

Algoritmo Esatto. Anche l'implementazione dell'algoritmo esatto, presenta come input il numero massimo di thread $threadPoolSize$ che l'algoritmo può utilizzare. Istanziati i diversi therad ciascun di loro richiede al componente *producer* l'assegnazione di sottoinsieme di autori da elaborare. Il *producer* fornisce ad ogni thread un numero intero composto da n-bit (dove n è il numero di autori) o il *null* se tutti i 2^n sottoinsiemi sono già stati assegnati. Il thread una volta ricevuto l'intero composto da n-bit lo trasforma in una coalizione, nello specifico se un bit del numero, è impostato a 1, allora il corrispondente autore è incluso nel sottoinsieme. Effettuata questa trasformazione l'algoritmo calcola i punteggi parziali per tutti gli autori del sottoinsieme e memorizza i valori ottenuti in una cache locale. Terminato il calcolo il thread richiede nuovamente al *producer* un nuovo sottoinsieme, se il *producer* ritorna *null* allora il thread si termina inviando i punteggi calcolati ad un componente *acceptor* che in modo sincronizzato incrementa il punteggio complessivo di ogni autore. Quando tutti i thread terminano, il componente *acceptor* calcola il valore esatto Shapley per tutti gli autori.

6.2 Esperimenti e risultati

Configurazione hardware e software. Gli esperimenti sono stati eseguiti su due macchine dedicate. In particolare, l'implentazione sequenziale degli algoritmi sono state eseguite su una macchina equipaggiata con un processore Intel Core i7-3770K 3.5 GHz, 12 GB (DDR3 a 1600 MHz) di RAM e sistema operativo Linux Debian Jessie. L'implementazione parallela degli algoritmi sono state testate su una macchina dotata di due processori Intel Xeon E5-4610 v2 2.30GHz con 8 core e 16 processori logici ciascuno, per un totale di 32 processori logici, 128 GB di RAM e sistema operativo Linux Debian Wheezy. Gli algoritmi sono stati implementati in Java, ed il codice è stato eseguito con JDK 1.8.0 05-b13 per la macchina Intel Core i7, e OpenJDK Runtime Environment (IcedTea 2.6.7) (7u111-2.6.7-1 deb7u1), per la macchina Intel Xeon.

Descrizione del dataset. Abbiamo applicato i nostri algoritmi per il calcolo di una divisione fair dei punteggi ottenuti dai ricercatori dell'Università della Sapienza di Roma i quali hanno partecipato all'esercizio di valutazione della qualità della ricerca negli anni 2011-2014. Il numero di ricercatori della Sapienza che hanno partecipato a tale esercizio è di 3562 ed è stato richiesto quasi a tutti di presentare almeno 2 pubblicazioni per il processo di revisione. Il punteggio di ogni pubblicazione è stato calcolato utilizzando le tabelle di valutazione fornite dall'ANVUR.

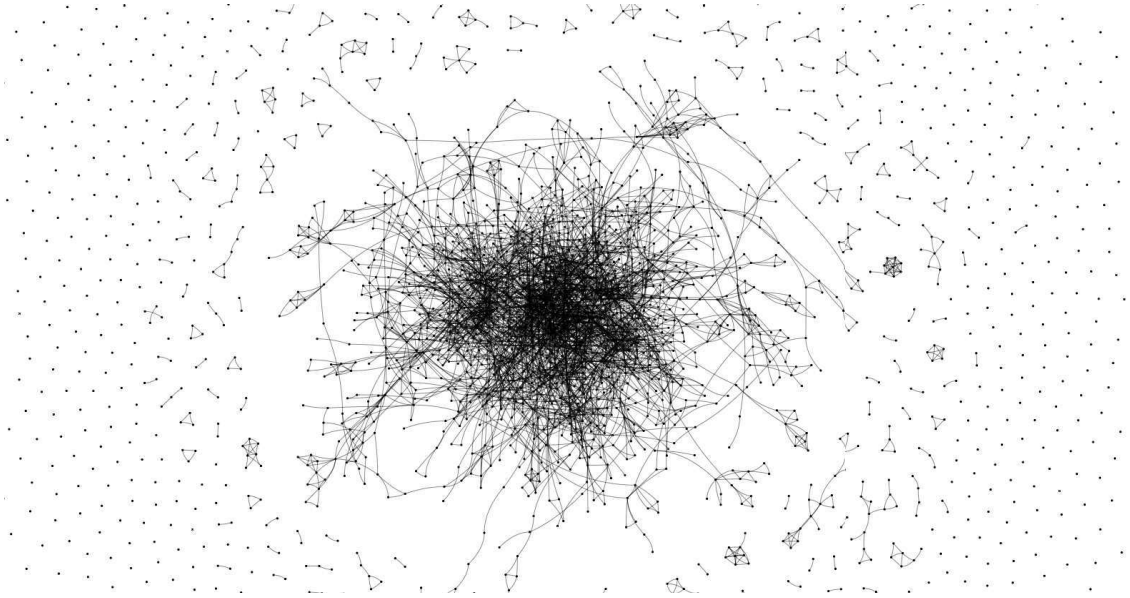


Figura 6.1: Grafo Autori Complessivo

Preprocessing. Il primo passo nelle nostre analisi consiste nella semplificazione dell'input applicando le proprietà discusse nel Capitolo 4, come spiegato di seguito.

Tabella 6.1: Componenti Connesse Grafo 1135

Dimensione della componente	Numero di componenti
2	94
3	36
4	7
5	8
6	4
7	1
8	2
9	2
15	1
691	1

Partendo da una situazione che comprende 3562 ricercatori e 5909 pubblicazioni, inizialmente si rimuovono tutti i ricercatori che non hanno pubblicazioni. Dopo questa fase, sono stati rimossi un totale di 370 ricercatori. Successivamente, sfruttando la semplificazione descritta nella sezione 4.1.2, sono state rimosse 2323 pubblicazioni. Dopo questa prima fase il grafo degli agenti assume la forma mostrata nella figura 6.1, dove è possibile osservare la presenza di diversi autori indipendenti e componenti connesse di diverse dimensioni.

Utilizzando il Teorema 4.1.1, il grafo è stato ulteriormente semplificato rimuovendo tutti gli autori aventi un contributo marginale alla grande coalizione uguale al valore dell'allocazione ottima ristretta all'autore stesso. Dopo questo passo, sono stati rimossi 2427 ricercatori su 3562.

Successivamente sono state calcolate le componenti connesse del grafo, il cui numero è risultato pari a 156. Di queste solo due presentano un numero di agenti maggiore di dieci e nello specifico le dimensioni delle componenti più gradi sono 691 e 15 come illustrato nella tabella 6.1. Infine, le componenti sono state ulteriormente scomposte come segue: Per ogni agente i, e per ogni bene g, l'agente i è rimosso dall'insieme degli agenti se la seguente formula è soddisfatta: $\texttt{val}(g) + \max_{g' \in \Omega(i) \setminus \{g\}} \texttt{val}(g') < v(N) - v(N \setminus \{i\})$. Dopo la fase di pre-elaborazione, si è ottenuto un totale di 159 componenti connesse, in cui la componente da 691 nodi è stata ridotta a 685. La dimensione della seconda più grande è solo di 15, mentre le restanti hanno subito un ulteriore semplificazione.

Nel resto della sezione si illustrano i risultati dell'attività sperimentale condotta con i diversi algoritmi proposti. A tal fine, è stato fissato il valore $\delta = 0{,}01$, in modo euristico sulla base di una serie di test condotti su varie aree CNR dell'università La Sapienza. Le aree CNR fanno riferimento alle discipline scientifiche di matematica e informatica (Area 01) e fisica (Area 02).

Esperimenti su componenti di varie dimensioni. Come già sottolineato nella sezione

precedente, dopo una fase di pre-elaborazione le componenti connesse ottenute, presentano delle dimensioni molto ridotte meno di 10 nodi, ad eccezione delle due più gradi il cui numero di nodi è rispettivamente di 685 e 15. Per tutte le componenti con meno di 10 nodi, è stato calcolato il valore di Shapley con l'algoritmo brute-force, che ha impiegato pochi millisecondi e quindi se ne omette l'analisi. Al fine di testare tutti gli algoritmi proposti oltre alle due componenti principali, sono stati selezionati in modo casuale alcuni nodi del grafo originale, con l'obiettivo di generare diversi sottografi connessi di dimensioni $|N| \in \{23, 26, 30, 40\}$ le cui rappresentazioni sono riportate nella figura 6.2. Per i casi considerati non sono state trovate differenze significative tra l'algoritmo approssimato basato su Liben-Nowell et al. e l'algoritmo brute-force. Tale risultato è riportato nelle Figure 6.3, 6.4 e 6.5 . In particolare, ad eccezione di un piccolo numero di casi, i limiti calcolati con l'algoritmo proposto sono sempre molto vicini (soprattutto i limiti inferiori) al valore esatto. Ad esempio, per $n = 26$ si è riusciti ad ottenere immediatamente il valore di Shapley per tutti gli agenti, in quanto limiti superiori e inferiori sono risultati coincidenti, inoltre dagli esperimenti, emerge che l'errore di approssimazione dell'algoritmo FPRAS è molto di più piccolo della garanzia teorica.

Sono state inoltre calcolate il numero di allocazioni ottime risparmiate (cioè i casi per cui non è stato necessario risolvere il problema di ottimizzazione). Fissando l'errore di approssimazione a $\epsilon = 0.3$, per ogni $n \in \{15, 23, 26, 30, 40\}$ si ottengono rispettivamente i seguenti risparmi: $9.65 \cdot 10^5$ contro $3.5 \cdot 10^6$ (i.e., 28%), $2.34 \cdot 10^6$ contro $1.29 \cdot 10^7$ (18%), $5.36 \cdot 10^6$ contro $1.87 \cdot 10^7$ (29%), $8.78 \cdot 10^6$ contro $2.9 \cdot 10^7$ (30%), e $1.46 \cdot 10^7$ contro $6.93 \cdot 10^7$ (21%).

Infine, si riportano gli errori di approssimazione medi e massimi (denotati rispettivamente con X e Y) della nostra implementazione confrontati con l'algoritmo esatto per ogni $n \in \{15, 23, 26\}$. In particolare, per $n = 15$ si ottiene X = 0.01 e Y = $3 \cdot 10^{-3}$, per $n = 23$ si ottiene X = 1,5 $\cdot 10^{-3}$ e Y = 1,7 $\cdot 10^{-4}$ e, per $n = 26$ si ottiene X = 1,06 $\cdot 10^{-4}$ e Y = 1,59 $\cdot 10^{-5}$. In tutti gli altri casi, l'errore di approssimazione massimo è di circa 1% (o meno) e quindi, abbondantemente al di sotto della garanzia teorica (30 %)

Analisi dei Tempi. Le Figure 6.6 e 6.7 riportano i tempi di esecuzione dei diversi algoritmi. In particolare, la Figura 6.6 illustra i tempi di esecuzione dell'implentazione brute-force per il calcolo del valore di Shapley, e i tempi delle implementazioni degli algoritmi per il calcolo dei limiti superiori e inferiori. Nella Figura 6.7 sono invece riportati i tempi di esecuzione dell'implementazione del metodo FPRAS per le diverse componenti al variare di ϵ.

Con riferimento alla Figura 6.6, è stato deciso di calcolare separatamente il limite inferiore e il limite superiore, al fine di evidenziare che in generale l'algoritmo per il calcolo del limite inferiore richiede più tempo, perché considera coalizioni più grandi rispetto quelle considerate nel calcolo del limite superiore. Inoltre, come già discusso nel Capitolo 4, i tempi di esecuzione per il calcolo di entrambi i limiti dipendono fortemente dalla cardinalità dell'insieme dei vicini degli agenti. Questo spiega il motivo per cui i

tempi di esecuzione per la componente $n = 52$ sono più bassi di quelli della componente $n = 40$. Nella tabella 6.2 è possibile osservare che nel caso $n = 40$ vi sono diversi autori che presentano un numero di vicini maggiore rispetto rispetto al caso $n = 52$.

Tabella 6.2: Dimensioni delle coalizioni

$\|C\|$	15	25	26	30	40	52
1	3	2	4	7	3	7
2	2	0	6	7	6	7
3	3	0	8	7	6	18
4	4	5	4	4	4	6
5	0	6	2	2	5	5
6	3	1	2	1	5	3
7	0	5	0	1	2	1
8	0	1	0	0	1	0
9	0	1	0	1	2	1
10	0	2	0	0	1	1
11	0	0	0	0	3	0
12	0	0	0	0	1	1
13	0	0	0	0	1	0
14	0	0	0	0	0	0
15	0	0	0	0	0	0
16	0	0	0	0	0	0

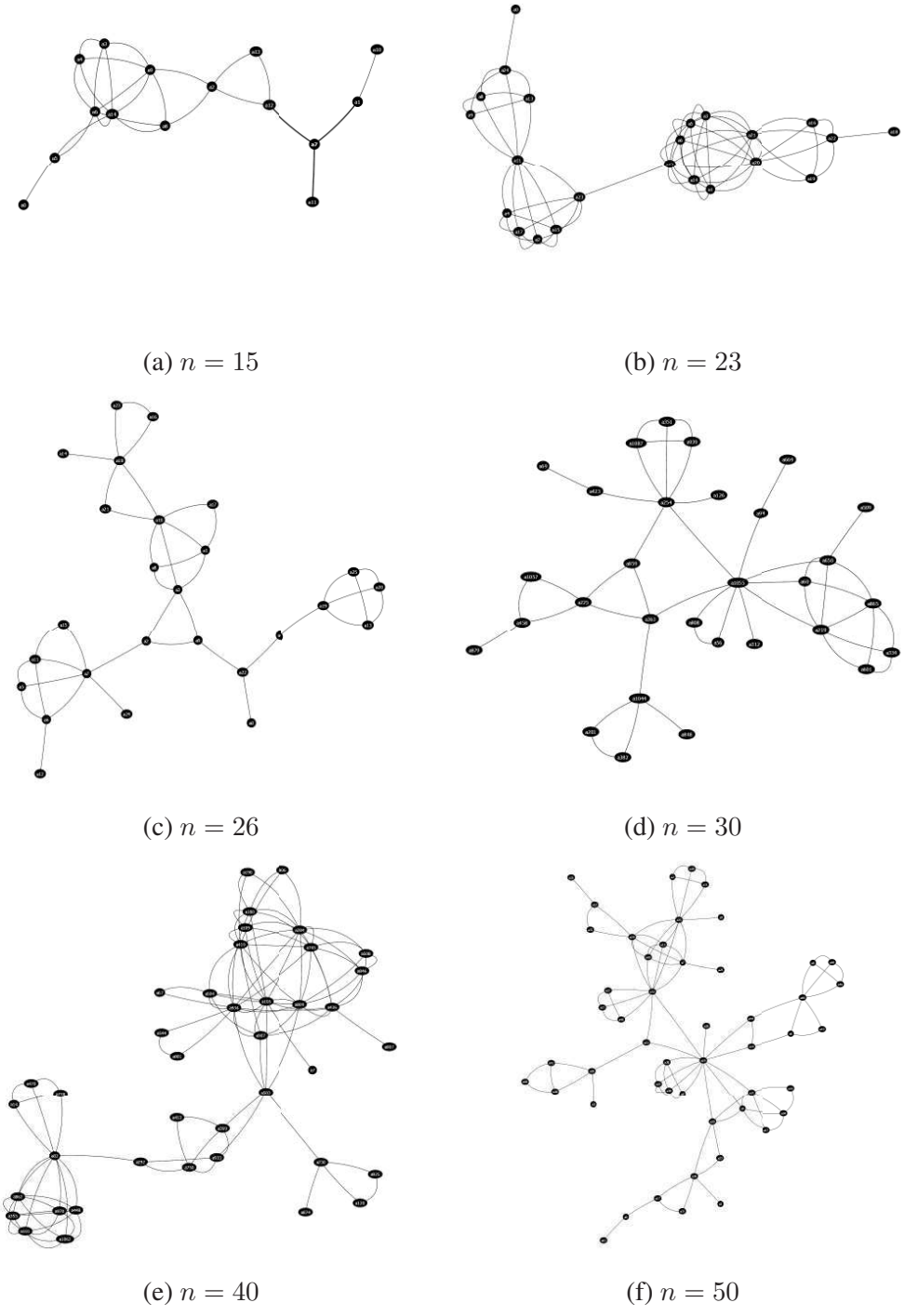
La Figura 6.7 mostra i tempi di esecuzione dell'implementazione del metodo FPRAS, che utilizza 24 threads, per i diversi valori di ϵ. In particolare, sono state effettuate cinque prove indipendenti per le diverse componenti di cui sopra. Quello che è possibile osservare e che per i giochi di dimensioni ragionevoli è possibile ottenere un garanzia teorica alta sull'errore di approssimazione.

Ad esempio, per la componente più grande di 52 agenti è stato possibile calcolare il valore di Shapley approssimato con un errore $epsilon = 0.1$ in meno di 90 minuti. Si osserva che esiste un forte divario in termini di prestazioni per l'algoritmo FPRAS quando si considerano i valori di ϵ che sono agli estremi dell'intervallo considerato. Tuttavia, come già sottolineato, anche in presenza di una garanzia teorica molto bassa sull'errore di approssimazione, abbiamo comunque ottenuto una buona un'accuratezza.

E' stato stimato che la nostra implementazione parallela dell'algoritmo FPRAS impiegherebbe, nel caso di $\epsilon = 0.05$ e 24 thread, circa 3.33 anni per analizzare completamente la componente più grande del caso di studio considerato che comprende 685 autori. Contrariamente, l'implementazione dell'algoritmo basata sui limiti proposti da Maleki et al., con le stesse impostazioni, richiede solo 11,75 ore. Tale limite sul numero di campioni richiede tuttavia di conoscere l'intervallo di valori di tutti i contributi marginali. Inoltre,

per poter gantire che i risultati siano entro una certa percentuale dei valori esatti, la nostra implementazione ha bisogno di conoscere preliminarmente i limiti inferiori del valore di Shapley.

Infine, si osservi che l'algoritmo basato sulla tecnica proposta da Maleki et al. può in principio essere applicato direttamente alla coalizione che comprende 1176 autori senza applicare ulteriori semplificazioni. In questo caso, il calcolo del valore approssimato per ogni autore richiede, sul nostro server, un errore assoluto pari a $\epsilon_{abs} = 5$, circa 20.5 ore. Se si incrementa la precisione dell'algoritmo portando l'errore assoluto ad un valore pari a $\epsilon_{abs} = 1$, il tempo di calcolo aumenta fino ad arrivare a 31 giorni.

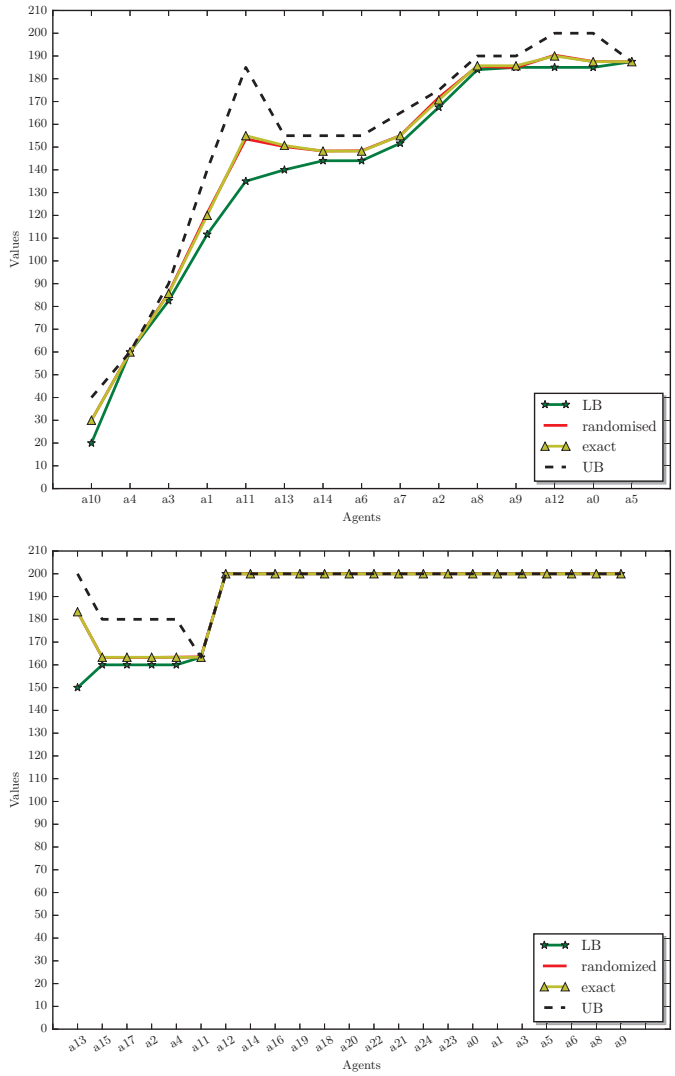


(a) $n = 15$ (b) $n = 23$

(c) $n = 26$ (d) $n = 30$

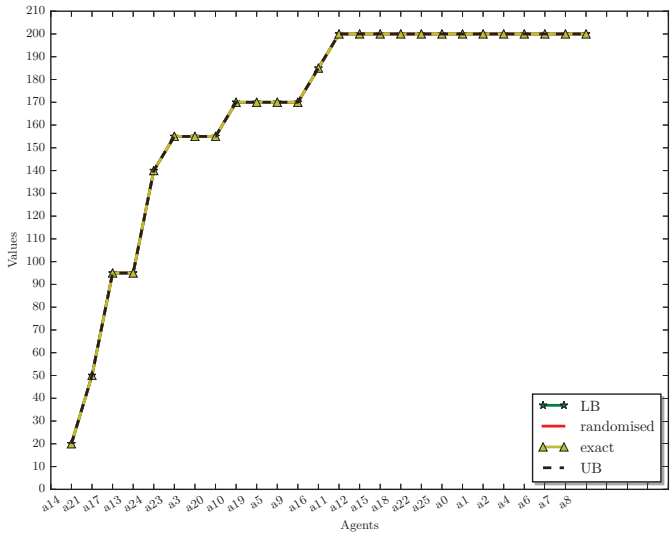
(e) $n = 40$ (f) $n = 50$

Figura 6.2: Componenti

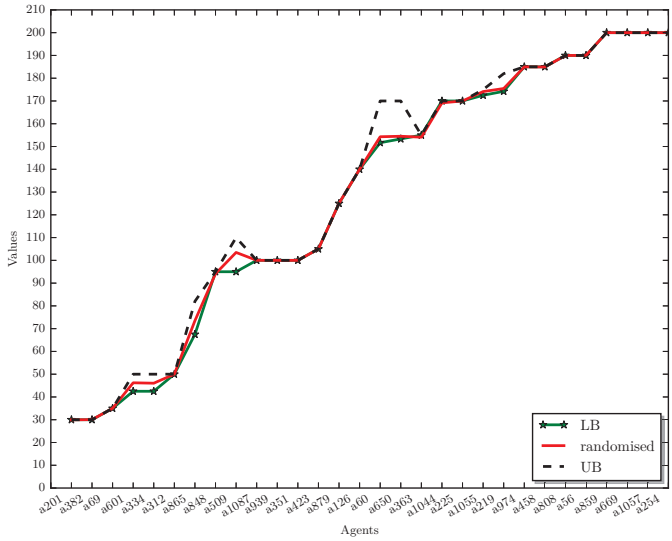


(a) $n = 23$

Figura 6.3: Confronto tra i diversi metodi (Parte 1)

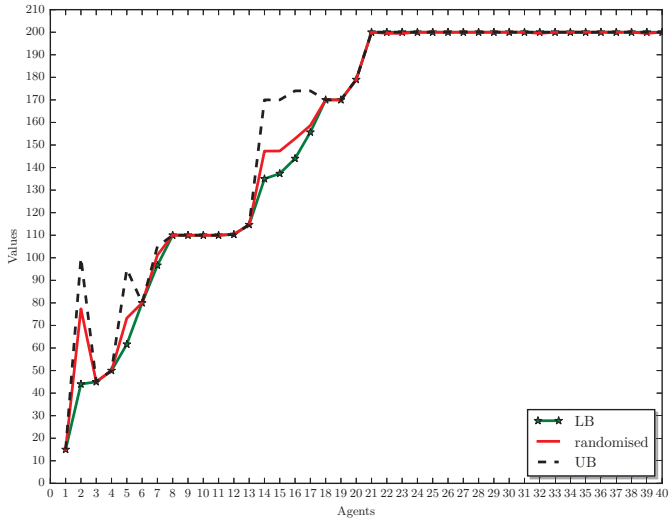


(a) $n = 26$

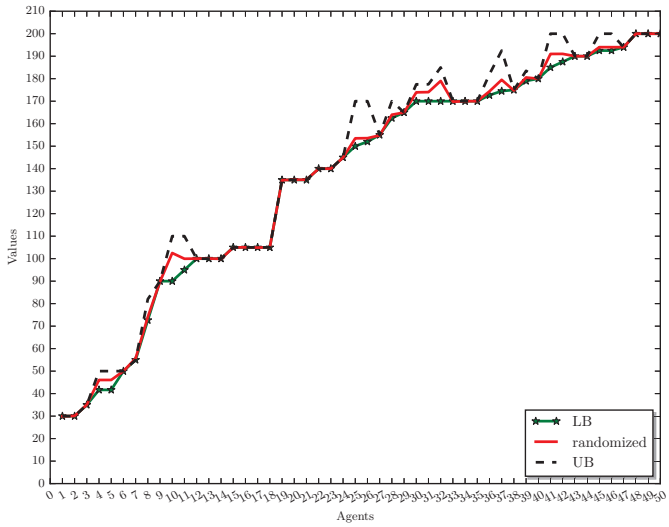


(b) $n = 30$

Figura 6.4: Confronto tra i diversi metodi (Parte 2)



(a) $n = 40$



(b) $n = 50$

Figura 6.5: Confronto tra i diversi metodi (Parte 3)

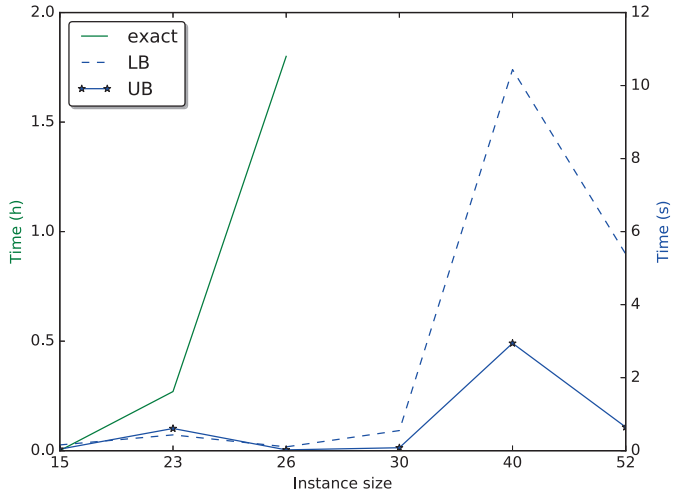


Figura 6.6: Tempi di esecuzione dell'algoritmo esatto (verde), upper bound e lower bound (blu) delle diverse istanze

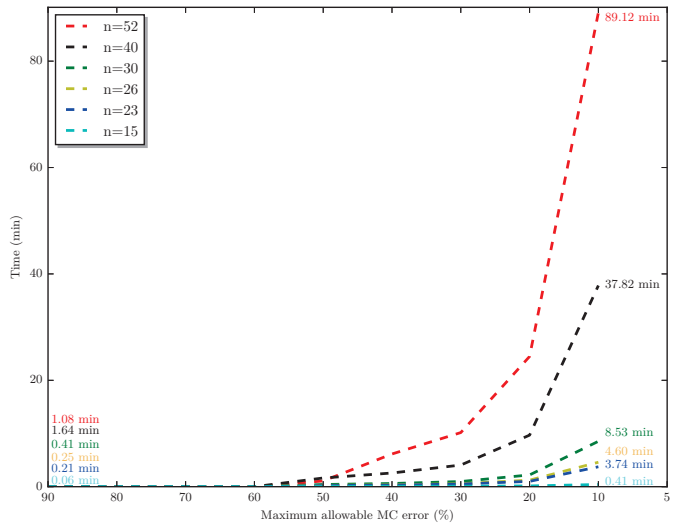


Figura 6.7: Implementazione Parallela: Tempi di esecuzione vs ϵ

Capitolo 7

Trattabilità del Calcolo del Valore di Shapley in Giochi di Treewidth Limitata

In questo capitolo si studia una nuova caratterizzazione del valore di Shapley in giochi di allocazione, finalizzata al calcolo del valore di Shapley in tempo polinomiale per i giochi aventi strutture a treewidth limitata. Le tecniche che saranno illustrate da qui in avanti si riferiscono a giochi di allocazione in cui ad ogni agente può essere allocato al massimo un bene, mentre nei capitoli precedenti è stata considerata una generalizzazione del problema in cui ad ogni agente poteva essere allocato un numero di beni k anche maggiore di 1 (nel caso di studio della VQR, $k = 2$).

La caratterizzazione descritta in questo capitolo è ispirata dal recente risultato di trattabilità in [14] che ha mostrato la trattabilità dei giochi di allocazione aventi un grafo degli agenti sia quasi-aciclico, riducendo il problema del calcolo del valore di Shapley al problema di contare in modo efficiente le soluzioni di una serie polinomiale di problemi di soddisfacimento di vincoli. A differenza di tale lavoro, in questa tesi sarà presentato un risultato più generale basato sulle proprietà strutturali del grafo bipartito agenti-beni e sul conteggio efficiente delle soluzioni di una serie di formule logiche monadiche del secondo ordine (MSO). La generalizzazione del risultato è significativa nella pratica, poiché la treewidth del grafo agenti-beni può essere arbitrariamente più piccola della treewidth del grafo degli agenti. Ad esempio uno scenario di allocazione in cui vi è un solo bene di interesse per 10 agenti induce una clique di cardinalità 10 nel grafo degli agenti, mentre il grafo bipartito agenti-beni è chiaramente aciclico.

7.1 Una Caratterizzazione per Livelli del valore di Shapley

Lo scenario di riferimento sono i giochi di allocazione, ristretti al caso in cui è ammessa l'allocazione di un solo prodotto per agente.

Assumiamo nel seguito che sia dato lo scenario $A = \langle N, G, \Omega, val \rangle$, dove $\{w_1, ..., w_m\} = \{val(g) | g \in G\}$ è l'insieme dei possibili valori (distinti) dei beni in G. Assumiamo senza perdita di generalità che $w_1 = 0 < w_2 < \cdots < w_m$. Ogni allocazione è una funzione $\pi : N \mapsto G$ tale che, per ogni $i, j \in N$, $\pi(i) \neq \pi(j)$.

Preliminarmente si osserva che sia il valore di Shapley sia il valore di Banzhaf di un giocatore i possono essere scritti considerando i contributi marginali di i alle varie possibili coalizioni raggruppate in base alla loro cardinalità.

$$\begin{cases} \phi_i(G_A) & = \sum_{h=0}^{n-1} \frac{h!(n-h-1)!}{n!} \beta_i(G_A, h) \\ \beta_i(G_A) & = \frac{1}{2^{n-1}} \sum_{h=0}^{n-1} \beta_i(G_A, h) \end{cases} \tag{7.1}$$

dove, per ogni $h \in \{0, ..., n-1\}$,

$$\beta_i(G_A, h) = \sum_{C \subseteq N \setminus \{i\}, |C|=h} (V(C \cup \{i\}) - v(C)). \tag{7.2}$$

Alla base dell'algoritmo che si intende proporre vi è il calcolo del numero delle coalizioni C tale che $|C| = h$ e $v_A(C \cup \{i\}) - v_A(C) \geq w_l$, cioè il numero di coalizioni di una data cardinalità alle quali i dà almeno un certo livello di contributo marginale w_l. Denotiamo tale numero con $\#c_l^i(G_A, h)$. In [14] è stato osservato che, mediante semplici manipolazioni algebriche e sfruttando la proprietà di monotonicità dei giochi di allocazione, i coefficienti $\beta_i(G_A, h)$ possono essere riscritti nella seguente forma:

$$\beta_i(G_A, h) = w_m \times c_m^i(G_A, h) + \sum_{l=1}^{m-1} w_l \times (\#c_l^i(G_A, h) - \#c_{l+1}^i(G_A, h) \tag{7.3}$$

Per calcolare efficientemente tali numeri è fondamentale per questo lavoro la seguente proprietà, che ci permette di individuare se un contributo marginale è maggiore di un certo livello senza risolvere problemi di ottimizzazione (come quelli usati per il calcolo dell'allocazione ottima nella definizione standard). Prima di riportare il risultato principale di questo capitolo, per una data coalizione C, definiamo l'insieme $A_\pi^i(C)$ degli agenti dipendenti da un giocatore $i \in N$ rispetto ad una allocazione π (per $C \cup \{i\}$) come il più piccolo insieme di agenti che soddisfa le seguenti proprietà:

1. $i \in A_\pi^i(C)$;
2. se $j \in A_\pi^i(C)$, $z \in C$ e $\pi(j) \in \Omega(z)$, allora $z \in A_\pi^i(C)$.

Questa definizione ci dice quindi che se un agente $z \in C$ è interessato ad un bene allocato ad un agente dipendente da i, allora è anch'egli dipendente da i. Il seguente risultato ci dice che trovare un'allocazione per tali agenti i cui beni abbiano tutti almeno valore w è sufficiente per inferire che il contributo marginale di i alla coalizione C è almeno w.

Teorema 7.1.1. *Siano $C \subseteq N$ una coalizione, $i \in N \setminus C$ e w il valore di un qualche bene. Allora le seguenti affermazioni sono equivalenti:*

1. $v_A(C \cup \{i\}) - V_A(C) \geq w$;
2. *esiste un'allocazione $\bar{\pi}$ per $C \cup \{i\}$ tale che per ogni $j \in A^i_{\bar{\pi}}(C)$ si ha $val(\bar{\pi}(j)) \geq w$.*

Dimostrazione. Preliminarmente osserviamo che se $\Omega(i)$ non contiene alcun bene di valore almeno w oppure contiene qualche bene g di valore maggiore o uguale a w che non interessa a nessun altro agente in C, allora lo statement vale banalmente. In particolare, (1) vale banalmente poiché il valore di g può essere sempre garantito aggiungendo i alla coalizione C; inoltre, è immediato verificare che anche (2) vale: è sufficiente considerare l'allocazione $\pi'(i) = g$, in corrispondenza della quale $A^i_{\pi'}(C) = \{i\}$.

Assumiamo quindi nel seguito che i sia interessato a qualche bene di valore almeno w e che tutti questi beni siano condivisi con altri agenti di C.

(1)⇒(2). Supponiamo per assurdo che (2) non valga. Allora per ogni allocazione π per $C \cup \{i\}$ esiste un $j \in A^i_{\pi}(C)$ per il quale $val(\pi(j)) < w$. Sia $\bar{\pi}$ un'allocazione ottima per $C \cup \{i\}$, per la quale vale ovviamente quanto sopra e consideriamo due possibili casi. In primo luogo assumiamo che $j = i$, cioè assumiamo che $\texttt{val}(\bar{\pi}(i)) < w$. Poiché la restrizione di $\bar{\pi}$ agli agenti di C è un'allocazione ammissibile per C, si ha $v_{\mathscr{A}}(C) \geq \texttt{val}(\bar{\pi}) - \texttt{val}(\bar{\pi}(i)) > \texttt{val}(\bar{\pi}) - w = v_{\mathscr{A}}(C \cup \{i\}) - w$, and hence $v_{\mathscr{A}}(C \cup \{i\}) - v_{\mathscr{A}}(C) < w$, che contraddice (1).

In secondo luogo assumiamo che $\texttt{val}(\bar{\pi}(i)) \geq w$. In questo caso deve esistere qualche altro agente in $A^i_{\bar{\pi}}(C)$ a cui l'allocazione assegna un bene di valore minore di w. Per costruzione di $A^i_{\bar{\pi}}(C)$, deve esistere un percorso $i = j'_1, j'_2, ..., j'_h$ such that $\bar{\pi}(j'_x) \cap \Omega(j'_{x+1}) \neq \emptyset$, for each $x \in \{1, ..., h\text{-}1\}$, e $\texttt{val}(\pi(j'_h)) < w$. Consideriamo allora la funzione $\bar{\pi}_{\text{-}i} : R \to \mathbb{G} \cup \{\emptyset\}$ con $\bar{\pi}_{\text{-}i}(j'_{x+1}) = \bar{\pi}(j'_x)$, per ogni $x \in \{1, ..., h\text{-}1\}$; e $\bar{\pi}_{\text{-}i}(j'') = \bar{\pi}(j'')$, per ogni $j'' \in C \setminus \{j'_2, ..., j'_h\}$. Notiamo che la funzione $\bar{\pi}_{\text{-}i}$ è un'allocazione per C ed il suo valore è $\texttt{val}(\bar{\pi}_{\text{-}i}) = \texttt{val}(\bar{\pi}) - \texttt{val}(\bar{\pi}(j'_h))$. Poiché questa variante di $\bar{\pi}$ non influisce sugli agenti in $C \setminus A^i_{\bar{\pi}}(C)$, abbiamo quindi $v_{\mathscr{A}}(C) \geq \texttt{val}(\bar{\pi}) - \texttt{val}(\bar{\pi}(j'_h)) > \texttt{val}(\bar{\pi}) - w$. Anche in questo caso otteniamo perciò $v_{\mathscr{A}}(C \cup \{i\}) - v_{\mathscr{A}}(C) < w$, che contraddice (1).

(2)⇒(1) Sia $\bar{\pi}'$ una allocazione per $C \cup \{i\}$ tale che $\texttt{val}(\bar{\pi}'(j)) \geq w$ per ogni $j \in A^i_{\pi'}(C)$. Consideriamo ora una allocazione ottima $\bar{\pi}$ per $C \cup \{i\}$. Notiamo preliminarmente che tale allocazione ottima non può assegnare ad alcun agente $z \in C \setminus A^i_{\pi'}(C)$ un qualche bene g allocato da $\bar{\pi}'$ ad agenti in $A^i_{\pi'}(C)$. Infatti ciò implicherebbe $g \in \Omega(z)$ e quindi anche z dovrebbe appartenere a $A^i_{\pi'}(C)$, per definizione di questo insieme.

Mostriamo ora che anche per l'allocazione ottima $\bar{\pi}$ vale $\texttt{val}(\bar{\pi}(j)) \geq w$ per ogni $j \in A^i_{\pi'}(C)$. Supponiamo per assurdo che non sia così e sia $j' \in A^i_{\pi'}(C)$ un agente tale che $\texttt{val}(\bar{\pi}(j')) < w$. Se il bene $\pi'(j')$, diciamo g (che ha valore almeno w), non è usato in $\bar{\pi}$ per nessun agente, allora potremmo utilizzare g invece del bene $\bar{\pi}(j')$ e quindi migliorare $\bar{\pi}$, contraddicendone immediatamente l'ottimalità. Se ciò non accade vuol dire

che g è allocato a qualche vicino di j' in $A^i_{\pi'}(C)$ ed anche in questo caso è possibile mostrare che si contraddice l'ottimalità di $\bar{\pi}$. Per ipotesi, infatti, vi sono beni di valore almeno w allocabili a tutti gli agenti in $A^i_{\pi'}(C)$ e che, come appena osservato, non possono essere stati allocati ad agenti in $C \setminus A^i_{\pi'}(C)$. Deve quindi esistere un bene di valore almeno w appartenente all'immagine di π' che non appartiene all'immagine di $\bar{\pi}$. Per costruzione dell'insieme $A^i_{\pi'}(C)$ è allora possibile individuare un percorso (simile a quello esibito al punto precedente) che ci permette di definire una nuova allocazione per $A^i_{\pi'}(C)$ in cui si fa uscire il bene di j' di valore minore di w e si fa entrare uno di tali beni esclusi, producendo un miglioramento rispetto all'allocazione $\bar{\pi}$ e quindi contraddicendone l'ottimalità.

Consideriamo adesso una allocazione ottima $\bar{\pi}_{\text{-}i}$ per la sola coalizione C (quindi senza l'agente i). Per la proprietà di motonicità delle allocazioni si ha $\texttt{img}(\bar{\pi}_{\text{-}i}) \subseteq \texttt{img}(\bar{\pi})$ e, in particolare, i beni usati per $A^i_{\pi'}(C) \setminus \{i\}$ sono tutti quelli già usati in $\texttt{img}(\bar{\pi})$ tranne uno, diciamo g'. Poiché $v_{\mathscr{A}}(C \cup \{i\}) = \texttt{val}(\texttt{img}(\bar{\pi}))$ and $v_{\mathscr{A}}(C) = \texttt{val}(\texttt{img}(\bar{\pi}_{\text{-}i}))$, il contributo marginale di i coincide con il valore di g' che, per quanto detto, è almeno w. □

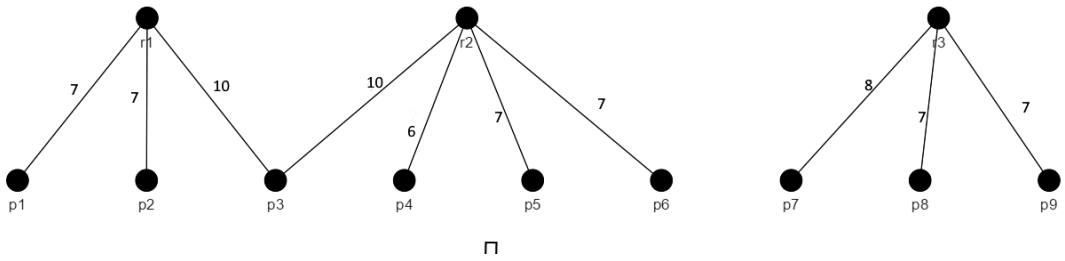


Figura 7.1: Grafo autori prodotti

Esempio 7.1. Si consideri lo scenario di allocazione A_0 della figura 7.1, dove è raffigurato un grafo bipartito agenti/beni in cui ogni arco descrive le relazioni di appartenenza di un bene ad un autore. Nello scenario sono presenti un insieme di agenti $N = \{a_1, a_2, a_3\}$ ed un insieme di beni $P = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8\}$, dove il valore dei beni è codificato sugli archi della figura 7.1. Per questo scenario un'allocazione ottima è data dall'insieme dei prodotti $\{p_2, p_7, p_8\}$ il cui valore è: $val(p_2) + val(p_7) + val(p_8) = 10 + 7 + 8 = 25$. Per ogni $C \subseteq \{a_1, a_2, a_3\}$, con $C \neq \emptyset$, le rispettive allocazioni ottime sono:

- $V_{A_0}(\{a_1, a_2, a_3\}) = 23$
- $V_{A_0}(\{a_1, a_2\}) = 17$
- $V_{A_0}(\{a_1, a_3\}) = 18$
- $V_{A_0}(\{a_2, a_3\}) = 15$

- $V_{A_0}(\{a_1\}) = 10$
- $V_{A_0}(\{a_3\}) = 8$
- $V_{A_0}(\{a_3\}) = 7$

Poiché le proprietà da verificare per ogni livello corrispondente ad un determinato valore w dei beni riguardano solo i beni aventi almeno quel valore, è conveniente considerare nel calcolo un diverso grafo G_{w_l} per ciascun livello l, nel quale sono contenuti solo i beni che presentano un valore maggiore o uguale a w_l. Nell'esempio, per $w = \{w_1, ..., w_m\} = \{6,7,8,10\}$, riportiamo di seguito l'insieme dei nodi di ciascun grafo (distinti per autori e prodotti):

- $G_6 = (\{a_1, a_2, a_3\}, \{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8\})$
- $G_7 = (\{a_1, a_2, a_3\}, \{p_1, p_2, p_4, p_6, p_7, p_8\})$
- $G_8 = (\{a_1, a_2, a_3\}, \{p_2, p_6\})$
- $G_{10} = (\{a_1, a_2, a_3\}, \{p_2\})$

Per ciascuno dei grafi di livello è necessario identificare gli agenti indipendenti, ovvero quegli agenti che presentano almeno un bene di valore almeno w_l non condiviso con altri, nell'esempio sono:

- G_6 $\{a_1, a_2, a_3\}$
- G_7 $\{a_1, a_2, a_3\}$
- G_8 $\{a_1, a_2, a_3\}$
- G_7 $\{a_3\}$
- G_{10} $\emptyset$

Per tali agenti indipendenti il parametro $\#c_l^i(G_A, h)$ è $\binom{n-1}{k}$ dove n è il numero degli autori (e h è la dimensione della coalizione). Questo perché ogni agente indipendente al livello l dà un contributo marginale almeno w_l a qualsiasi coalizione. Per gli agenti che non hanno alcun bene di interesse di valore almeno w_l tale numero di coalizioni è banalmente 0. Negli altri casi il calcolo non è immediato ed una implementazione efficiente richiede particolari tecniche discusse nel seguito. Nel caso dello scenario in questione, per ogni agente sono state calcolate il numero di coalizioni per ogni livello

e riportate nelle seguenti matrici, dove l'indice di colonna indica la dimensione delle coalizioni e le righe corrispondono ai diversi livelli di valore.

$$r_1 = \begin{bmatrix} 1 & 2 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \quad r_2 = \begin{bmatrix} 1 & 2 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \quad r_3 = \begin{bmatrix} 1 & 2 & 1 \\ 1 & 2 & 1 \\ 1 & 2 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

Applicando la formula

$$\beta_i(G_A, h) = w_m \times c^i_m(G_A, h) + \sum_{l=1}^{m-1} w_l \times (\#c^i_l(G_A, h) - \#c^i_{l+1}(G_A, h)$$

otteniamo per i diversi livelli

$$B_1(G,0) = 10,\ B_1(G,1) = 15,\ B_1(G,2) = 7$$
$$B_2(G,0) = 10,\ B_2(G,1) = 15,\ B_2(G,2) = 7$$
$$B_3(G,0) = 10,\ B_3(G,1) = 16,\ B_3(G,2) = 7$$

Infine utilizzando la formula

$$\phi_i(G_A) = \sum_{h=0}^{n-1} \frac{h!(n-h-1)!}{n!} \beta_i(G_A, h)$$

è possibile ottenere il valore Shapley

$$sh_{a_1}(G) = 3{,}33 + 2{,}5 + 2{,}33 = 8{,}16$$
$$sh_{a_2}(G) = 3{,}33 + 2{,}5 + 2{,}33 = 8{,}16$$
$$sh_{a_3}(G) = 3{,}33 + 2{,}66 + 2{,}33 = 8{,}33$$

Nel prosieguo di questo capitolo sfruttiamo le caratterizzazioni appena dimostrate per ottenere un algoritmo polinomiale per il problema del calcolo del valore di Shapley in giochi i cui grafi agenti-beni abbiano basso grado di ciclicità. Come misura di ciclicità usiamo la potente e ben nota nozione di *treewidth* [32]. Per completezza, riportiamo in appendice la definizione formale di treewidth e dell'associata nozione di tree decomposition di un grafo ed i concetti fondamentali relativi alla logica monadica del secondo ordine (MSO), insieme ad alcuni risultati rilevanti per la tesi.

7.2 Una formula MSO per individuare le coalizioni con un determinato livello di contributo marginale

Il Teorema 7.1.1 ci permette di identificare le coalizioni $C \subseteq N \setminus \{i\}$ a cui i dà un contributo marginale almeno w_l come quelle coalizioni per le quali esiste un'allocazione $\bar{\pi}$ tale che, per ogni $j \in A^i_{\bar{\pi}}(C)$, si ha $val(\bar{\pi}(j)) \geq w_l$.

In questa sezione presentiamo una formula MSO la cui variabile libera C del secondo ordine rappresenta un insieme di agenti, che interpretiamo come una coalizione con la proprietà desiderata da restituire in output. L'input alla formula sarà un grafo agenti-beni G_l ottenuto dal grafo del problema da risolvere mantenendo solo i beni di valore almeno w_l. Permettiamo quindi che $\bar{\pi}$ sia un'allocazione parziale, ma è sufficiente verificare che tutti gli agenti dipendenti in $A^i_{\bar{\pi}}(C)$ abbiano un bene (non occorre verificare nulla riguardo il valore). Poiché ognuno di loro deve quindi avere esattamente un bene, si tratta quindi di verificare l'esistenza di un matching nel grafo bipartito G_l in cui tutti gli agenti in $A^i_{\bar{\pi}}(C)$ siano *matched*.

Riepilogando, occorre quindi fare un *guess* di un insieme M che rappresenti i rami di un matching e verificare che tutti gli agenti in $A^i_{\bar{\pi}}(C)$ abbiano esattamente un bene. Un sovrainsieme degli agenti dipendenti $A^i_{\bar{\pi}}(C)$ può a sua volta essere individuato con un *guess*, usando un'altra variabile esistenziale del secondo ordine P sulla quale occorre verificare le proprietà

1. $i \in A^i_{\bar{\pi}}(C)$;
2. se $j \in A^i_{\bar{\pi}}(C)$, $z \in C$ e $\pi(j) \in \Omega(z)$, allora $z \in A^i_{\bar{\pi}}(C)$.

È importante osservare che non siamo in grado di verificare che P corrisponda esattamente a $A^i_{\bar{\pi}}(C)$, cioè che sia il più piccolo insieme di agenti con tali proprietà. Tuttavia essendo il tutto in una quantificazione esistenziale l'unico modo di fallire è che nessun insieme con le proprietà richieste (nemmeno il più piccolo) ci porti ad ottenere l'allocazione desiderata.

Poiché il sistema di risoluzione delle formule MSO che abbiamo usato non supporta la quantificazione del secondo ordine su insiemi di archi, il grafo G_l usa in realtà dei nodi aggiuntivi per rappresentare i rami del grafo originale. Quindi un ramo $\{p, b\}$ è rappresentato con un nuovo nodo $e_{p,b}$ e G_l contiene i rami $\{p, e_{p,b}\}$ e $\{e_{p,b}, b\}$. Il sistema usato, che si chiama Sequoia e sarà descritto in maggiore dettaglio nel capitolo seguente, supporta tuttavia la presenza di predicati unari nel grafo di input, chiamati *colori*, che semplificano la scrittura della formula. Si noti che tali colori non corrispondono a predicati unari arbitrari, poiché ogni nodo del grafo ha il vincolo di poter comparire in uno solo di essi, cioè può avere un solo colore. Nella descrizione che segue useremo direttamente le convenzioni di Sequoia, con le sue limitazioni e con l'uso dei colori.

L'input è quindi costituito dal grafo G_l (con la rappresentazione sopra descritta in cui i rami del grafo originario sono nodi aggiuntivi) e dai seguenti predicati unari (colori), trattati nella formula come insiemi di nodi:

- G, che rappresenta i nodi di tipo *bene* in G_l;
- E, che rappresenta i nodi di tipo *ramo* in G_l;
- I, che rappresenta l'insieme singoletto $\{i\}$ che individua l'agente i di cui intendiamo calcolare i contributi marginali.

Nelle sottoformule di seguito riportate, al fine di rendere più semplice la comprensione, è stata usata la seguente convezione:

- g è un bene, quindi un elemento in G;
- e è un ramo, quindi un elemento in E;
- x e y rappresentano nodi di tipo *agente*.

Siamo finalmente pronti a riportare la formula MSO che ci occorre, in cui la variabile libera C rappresenta una coalizione a cui i dà un contributo marginale almeno w_l.

$$\exists\, M\, \exists\, P\, \exists\, i\, (i \in I\ \wedge\ i \in P\ \wedge\ i \notin C\ \wedge \\ \forall x(x \notin C \vee agent(x, G, E)) \wedge \\ conn(i, C, M, P, G, E)\ \wedge\ matching(i, C, P, M, G, E))$$

Si noti che la formula è soddisfatta se esiste un matching M ed un insieme di agenti dipendenti P (rispetto all'allocazione codificata da M) con le proprietà richieste. In particolare la formula richiede che la variabile i sia istanziata con il particolare agente $i \in I$ su cui ci stiamo concentrando; che $i \notin C$; che C contenga solo nodi di tipo agente; che P soddisfi le proprietà richieste per l'insieme di agenti dipendenti (rispetto all'allocazione codificata da M e rispetto all'agente di partenza i); che ogni agente in P abbia esattamente un bene allocato.

Riportiamo di seguito la definizione di tutte le sottoformule usate:

- **Agente**: per definire che nodo è un agente, è necessario che questo non appartenga all'insieme dei beni o all'insieme dei rami

$$agent(x, G, E) := x \notin G \wedge x \notin E$$

- **Adiacenza tra un Bene e un Agente**: per poter identificare se un agente x è proprietario di un bene g, è necessario definire che il nodo x sia effettivamente un agente, e che inoltre esiste un nodo arco e tale che x è collegato con e ed e è collegato con g

$$adjG(x, g, G, E) := agent(x, G, E)\ \wedge\ \exists\, e\, (e \in E \wedge\ adj(x, e) \wedge\ adj(e, g))$$

- **Agenti dipendenti**: la formula riportata verifica che P contenga gli agenti dipendenti rispetto ad M (più precisamente l'allocazione codificata dall'insieme M) ed

all'agente i

$$\begin{gathered} conn(i, C, M, P, G, E) := \forall x((\neg(x \in C \ \wedge \ \exists\, y\, \exists\, e\, \exists\, g \\ (y \in P \ \wedge \ e \in M \ \wedge g \in G \ \wedge \ adj(y, e) \\ \wedge adj(e, g) \wedge \ adjG(x, g, G, E))) \\ \vee(x \in P \ \exists\, e_1(e_1 \in M \wedge adj(x, e_1))))) \wedge \\ \forall x(x \notin P \ \vee \ (x \in C \ \vee \ x = i)) \end{gathered}$$

- **Matching**: verifichiamo che gli agenti dipendenti rispetto ad M ed i abbiano un collegamento in M con esattamente un bene dell'insieme G. Poiché in realtà il *guess* di P potrebbe contenere elementi non desiderati, la proprietà viene in realtà verificata sul sottoinsieme a cui siamo interessati, segnatamente $P \cap (C \cup \{i\})$

$$\begin{gathered} matching(i, C, P, M, G, E) := \\ \forall x(\neg(x \in P \ \wedge \ (x \in C \ \vee \ x = i)) \ \vee \ \exists e(e \in M \ \wedge \ adj(x, e))) \wedge \\ \forall e(e \notin M \ \vee \ (\forall a \, \forall g(\neg(adj(a, e) \ \wedge \ adj(g, e) \ \wedge \ \neg g = a) \vee \\ (\forall e_1(\neg(adj(a, e_1) \ \wedge \ e_1 \in M) \ \vee \ e_1 = e) \wedge \\ \forall e_2(\neg(adj(g, e_2) \ \wedge \ e_2 \in M) \ \vee \ e_2 = e))))) \wedge \\ \forall e(e \notin M \ \vee \ e \in E) \end{gathered}$$

7.3 Problemi di conteggio e calcolo efficiente del valore di Shapley

Come descritto in Sezione 7.1, il problema di calcolare il valore di Shapley si riduce al problema di calcolare i coefficienti riportati in Equazione 7.3.

Ricordiamo che, per un dato livello di valore dei beni w_l e per una data cardinalità delle coalizioni $0 < h < |N|$, per calcolare il coefficiente $\#c_l^i(G_A, h)$ dobbiamo contare il numero di coalizioni $C \subseteq N \setminus \{i\}$ di cardinalità h alle quali l'agente i dà contributo marginale almeno w_l. Per il Teorema 7.1.1, ciò equivale a contare le coalizioni $C \subseteq N \setminus \{i\}$ della cardinalità desiderata per le quali esiste un'allocazione $\bar{\pi}$ tale che, per ogni $j \in A_{\bar{\pi}}^i(C)$, si ha $val(\bar{\pi}(j)) \geq w_l$.

Nella precedente sezione abbiamo illustrato una formula MSO in grado di individuare le coalizioni desiderate. Sfruttiamo a questo punto un importante risultato, noto in letteratura e riportato di seguito in una forma leggermente adattata e semplificata, che stabilisce la trattabilità del problema del conteggio di soluzioni di formule MSO su istanze in input aventi treewidth limitata.

Sia $\phi(C)$ una formula MSO in cui C è una variabile insieme (sui nodi) libera e, per ogni $\hat{C} \subseteq N$, denotiamo con $\phi(\hat{C})$ la formula ottenuta sostituendo la variabile C con lo specifico insieme di nodi $\hat{C}$. Per ogni grafo $G = (N, E)$ ed ogni numero naturale $d \in \{1, ..., |N|\}$, definiamo inoltre $\texttt{histogram}(\phi(C), G, d)$ come il numero dei sottoinsiemi $\hat{C} \subseteq N$ aventi cardinalità $|\hat{C}| = d$ e che permettono di soddisfare la formula, cioè per i quali si ha $G \models \phi(\hat{C})$.

Teorema 7.3.1. *[46] Sia $\phi(C)$ una formula MSO con variabile libera C e $\mathscr{G}$ una classe di grafi la cui treewidth è limitata da un numero naturale (finito). Allora per ogni $G = (N, E) \in \mathscr{G}$ e per ogni numero naturale $d \in \{1, ..., |N|\}$, il numero $\texttt{histogram}(\phi(C), G, d)$ è calcolabile in spazio logaritmico (deterministico) e quindi è anche calcolabile in tempo polinomiale.*

Sfruttando questo teorema ed i risultati presentati nelle precedenti sezioni, otteniamo il risultato principale di questo capitolo.

Teorema 7.3.2. *Sia $\mathscr{A}$ una classe di giochi di allocazione tali che, per ogni $A \in \mathscr{A}$, la treewidth del grafo agenti-beni di A abbia treewidth al più k, per qualche k finito. Allora per ognuno di tali giochi il valore di Shapley di ciascun agente può essere calcolato in tempo polinomiale. Lo stesso vale per il valore di Banzhaf.*

Dimostrazione. Dato un gioco $A \in \mathscr{A}$, è possibile utilizzare la caratterizzazione del valore di Shapley descritto in Sezione 7.1. Per ottenere il valore di Shapley sv_i per un agente i del gioco A dobbiamo calcolare, per ogni livello di valore dei beni w_l e per ogni cardinalità delle coalizioni $0 \leq h < |N|$, il coefficiente $\#c_l^i(G_A, h)$. Si noti che il numero di questi coefficienti è polinomiale rispetto alla taglia dell'input.

Per ogni livello di valore w_l possiamo costruire a partire dal gioco A il sottografo G_l agenti-beni in cui sono presenti solo i beni di valore almeno pari a w_l (ed in cui i rami del grafo originale sono rappresentati mediante nodi aggiuntivi). Lanciata con tale grafo G_l in input, la formula MSO descritta nella precedente sezione individua, attraverso la variabile libera C, le coalizioni alle quali i dà contributo marginale almeno w_l. Il Teorema 7.3.1 stabilisce inoltre che il numero di soluzioni di tale formula di una determinata cardinalità h è calcolabile in tempo polinomiale se la treewidth dei grafi in input alla formula è limitata superiormente da una costante. Ciò vale per ipotesi per la classe di giochi $\mathscr{A}$.

Mettendo insieme le varie fasi sopra descritte abbiamo quindi un algoritmo polinomiale per calcolare il valore di Shapley. La stessa tecnica si applica immediatamente anche per il calcolo del valore di Banzhaf. □

Capitolo 8

Implementazione del metodo polinomiale basato su MSO

La tecnica descritta nel precedente capitolo è stata implementata e testata usando il programma Sequoia [51] come risolutore delle formule MSO. In verità la versione di Sequoia che era disponibile quando abbiamo iniziato ad usarlo non conteneva l'implementazione del calcolo degli istogrammi per i problemi di conteggio. Abbiamo quindi collaborato con l'autore di Sequoia che, con grande disponibilità, ha esteso il software e messo a disposizione la versione corrente che supporta gli istogrammi.

8.1 Preparazione dell'input e calcolo del valore di Shapley

Dato un gioco di allocazione, si costruisce preliminarmente il grafo bipartito agenti-beni, si analizza il valore dei beni e si determina i diversi livelli w_l. Per ciascuno dei livelli, vengono generati i rispettivi grafi di livello G_{w_l} che contengono solo i prodotti che presentano un valore maggiore o uguale di w_l. Per ciascun grafo di livello possono essere effettuate delle semplificazioni, individuando gli agenti indipendenti per i quali il numero delle coalizioni con un determinato contributo marginale può essere immediatamente calcolato.

Diciamo che un agente i è indipendente rispetto ad un livello di valore w_l se esiste $g \in \Omega(i)$ tale che $\texttt{val}(g) \geq w_l$ e g è di esclusivo interesse per i. Un tale agente indipendente può essere temporaneamente isolato insieme al suo bene g non condiviso con gli altri. Nel momento in cui viene isolato, l'agente rilascia tutti gli altri beni in $\Omega(i) \setminus \{g\}$, tecnicamente vengono eliminati gli archi in G_{w_l} tra questi beni ed i. Questo processo di semplificazione continua iterativamente, poiché il processo di isolamento può generare in cascata l'identificazione di altri agenti indipendenti tra i suoi vicini.

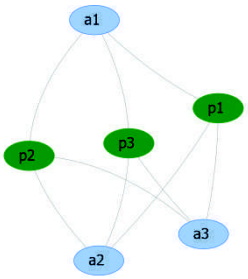


Figura 8.1: Un grafo agenti-beni

Per i restanti agenti occorre preparare il grafo perché possa essere passato in input al tool Sequoia. Come già descritto nel precedente capitolo, per realizzare la quantificazione esistenziale su insiemi di rami, questi vengono rappresentati come nuovi nodi del grafo, come illustrato in Figura 8.2. È importante osservare che tale modifica non incrementa la treewidth del grafo.

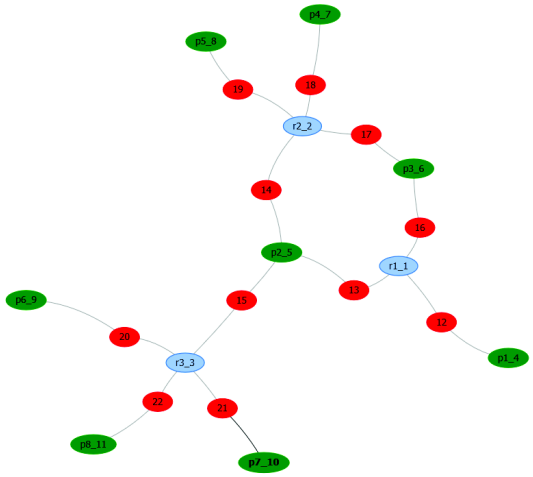


Figura 8.2: Trasformazione del grafo della figura 7.1

La Figura 8.2 mostra anche i colori usati come predicati unari aggiuntivi dell'istanza in input alla formula MSO. In particolare i nodi in verde rappresentano il predicato G corrispondente ai beni; quelli in rosso rappresentano il predicato E corrispondente ai

nuovi nodi che codificano i rami del grafo originario. Per una migliore visualizzazione la figura rappresenta anche col colore blu gli agenti.

Per i restanti agenti il sistema istanzia un problema di conteggio basato sulla formula MSO sopra descritta e lancia il tool Sequoia con il parametro *CardCount* che permette di calcolare, con un'unica esecuzione, un vettore con i numeri di coalizioni richieste per ciascun valore di cardinalità. Terminati i problemi di conteggio sui diversi livelli, è finalmente possibile calcolare il valore di Shapley per ciascun agente. In questa fase finale occorre porre attenzione al calcolo dei termini $\frac{h!(n-h-1)!}{n!}$ nella Formula 7.1, per i quali occorre procedere con tutte le semplificazioni possibili tra numeratore e denominatore, senza calcolare realmente i vari fattoriali che vi appaiono.

8.2 Esperimenti

8.2.1 Configurazione hardware e software.

Gli esperimenti sono stati effettuati su una macchina dedicata con processore Intel Core i7-3770K 3.5 GHz, 12 GB (DDR3 a 1600 MHz) di RAM e sistema operativo Linux Debian Jessie. Il sistema è costituito da una parte Java e dal tool Sequoia per la risoluzione delle formule MSO.

8.2.2 Descrizione dataset e tempi di esecuzione

Gli esperimenti qui presentati sono stati condotti sulla variante del gioco di allocazione con un solo bene del problema VQR per l'università "La Sapienza" di Roma. Come vedremo, la tecnica proposta è stata in grado di risolvere completamente e in modo esatto questa istanza reale con migliaia di nodi.

Un primo esperimento ha riguardato le componenti utilizzate negli esperimenti precedenti, con $|N| \in \{15, 30, 40, 52\}$, dove l'algoritmo esatto risulta addirittura più veloce di quello approssimato (parametri: errore massimo 0.9% e probabilità di errore $\delta = 0.01\%$). I risultati sono riportati nella tabella, 8.1.

Tabella 8.1: Tempi di esecuzione

$\|C\|$	#nodi	#archi	treewidth	Randomizzato (sec)	MSO - esatto (sec)
15	104	116	3	701	0,522
30	226	250	4	33,124	4,9
40	383	468	6	1257,622	0,81
52	348	394	3	300,158	233,158

Si noti che le ottime prestazioni dell'algoritmo basato su MSO sono anche dovute alle semplificazioni effettuate nella fase di pre-processamento dell'input. Le seguenti tabelle mostrano in particolare il numero di autori privi di prodotti e di autori indipendenti per ciascun livello di valore.

Tabella 8.2: Autori privi di prodotti per livello

$\|C\|$	0	10	40	70	100
15	0	-	0	2	4
30	0	3	0	0	8
40	0	0	1	4	7
50	0	0	4	7	16

Tabella 8.3: Autori indipendenti per livello

$\|C\|$	0	10	40	70	100
15	15	0	15	13	9
30	30	28	26	23	15
30	40	40	39	36	31
30	50	50	44	34	27

Le semplificazioni riducono notevolmente la cardinalità complessiva dell'input ed il numero di sotto problemi da risolvere: per la componente da 15 sono solo 2, per la componente da 30 sono 10, per la componente da 40 sono 2 e per la componente da 50 sono 18. Ciò giustifica i tempi di esecuzione delle diverse componenti riportate nella tabella 8.1, dove si osserva che i tempi di calcolo delle componenti da 15 e 40 sono di gran lunga più bassi delle componenti da 30 e 52. Inoltre, è da osservare che il numero di nodi e di archi della tabella 8.1 fa riferimento alle dimensioni dei grafi del livello più basso, sui quali non è stato effettuato alcun tipo di pre-processing.

Inoltre, sono stati condotti altri esperimenti su vari sottografi del problema volti all'analisi della scalabilità della soluzione. Come era attendibile, la tabella 8.4, mostra che i tempi aumentano molto quando la treewidth aumenta.

È stato poi risolto il problema relativo l'intera istanza (ma considerando il problema di allocazione ad un bene) del gioco relativo al problema VQR per l'Università di Roma "La Sapienza". Nello specifico:

1. E' stato costruito un gioco di allocazione che vede coinvolti tutti gli autori e solo i prodotti a loro assegnati nella migliore possibile allocazione di un prodotto a ciascun autore.

2. Il grafo associato a tale gioco presenta presenta una sola grossa componente di 570 autori.

Tabella 8.4: Tempi di esecuzione su sottografi del problema VQR

#Prodotti + #Agenti	# Nodi	# Archi	# Agenti	Treewidth	Tempi in sec
10	23	26	3	2	23
10	23	26	4	3	28
10	26	32	5	4	89
20	43	46	7	2	87
20	45	50	9	3	186
20	50	60	7	4	818
30	66	72	12	2	178
30	78	96	11	3	853
30	69	78	12	4	4209
40	81	82	15	2	266
40	95	110	17	3	3032
40	99	118	17	4	4212
50	103	106	16	2	283
50	110	120	20	3	540

3. Con le tecniche di semplificazione descritte nei capitoli precedenti, che hanno richiesto 19 minuti, per molti autori è stato immediatamente possibile calcolare il valore di Shapley ed il grafo si è ulteriormente frammentato. La componente più grande del grafo finale ha cardinalità 23.

4. Per ciascuna delle altre componenti, tutte di cardinalità al più 15, è stato calcolato il valore esatto di Shapley in pochi secondi.

5. Per la componente da 23 è stato calcolato il valore esatto con l'algoritmo *brute-force* in circa 7 minuti, ed una approssimazione con $\epsilon = 0.9$ e $\delta = 0.01$, che ha richiesto circa un minuto.

6. Sempre per la componente da 23 è stato utilizzato l'algoritmo esatto basato su MSO che ha richiesto 20 secondi. In particolare il grafo di questa componente (corrispondente al livello 0) presenta una treewidth pari a 3, tuttavia i grafi di livello effettivamente generati hanno treewidth massima 2. Sono stati eseguiti in tutto 30 problemi di conteggio con Sequoia.

Capitolo 9

Conclusioni e sviluppi futuri

Il presente lavoro di tesi si è concentrato sul calcolo del valore di Shapley in giochi di allocazione, alla ricerca di schemi di approssimazione polinomiale e di casi trattabili, basati sulle proprietà strutturali del grafo che lega agenti e beni da allocare. Gli algoritmi polinomiali descritti per tali istanze si basano sulla relazione tra il valore di Shapley ed i problemi di conteggio di soluzioni in formule monadiche del secondo ordine.

Il lavoro presenta anche un caso di studio reale, in cui i giochi di allocazione sono stati applicati al problema della valutazione della ricerca in Italia. A differenza di molti altri casi di applicazione della teoria dei giochi, questa applicazione coinvolge giochi con migliaia di agenti (e di beni da allocare). Per questo motivo il lavoro è stato anche volto ad individuare proprietà dei giochi di allocazione che permettessero di semplificare le istanze senza alterare il valore di Shapley. Sono stati inoltre descritti degli algoritmi per il calcolo di un lower bound ed un upper bound del valore di Shapley. Tali valori permettono spesso di ottenere direttamente il valore corretto del valore di Shapley per un agente (laddove i limiti sono coincidenti) e, in ogni caso, permettono di avere valutazioni più accurate degli errori di approssimazione e del numero di campioni che gli algoritmi approssimati devono considerare per ottenere una determinata qualità della soluzione.

Vi sono ancora molti problemi da affrontare in quest'ambito e che possono essere oggetto di lavori futuri. Ad esempio, è probabile che le tecniche che ci hanno permesso di individuare i casi trattabili con al massimo un bene allocabile a ciascun autore, possano essere estese alla generalizzazione dei giochi di allocazione in cui ad ogni agente possa essere allocato un numero fisso k di beni (cioè la generalizzazione considerata nella prima parte della tesi). Inoltre può essere interessante studiare diversi concetti di soluzione per i giochi di allocazione, ad esempio il *kernel* o il *nucleolo*, per individuarne ambiti di applicazione e casi trattabili.

Bibliografia

[1] J.von Neumann and O. Morgenstern. Theory of Games and Economic Behavior. Princeton, NJ, USA: Princeton University Press, Sep. 1944.

[2] R. Aumann. Accetable points in general cooperative n-person games. Contribution to the Theory of Games, IV, 1959.

[3] C.J. Colbourn, J.S. Provan, D. Vertigan. The complexity of computing the tutte polynomial on transversal matroids. Combinatorica, 15(1):1– 10, 1995.

[4] R. Myerson. Game Theory, Analysis of Conflict. Cambridge, MA, USA: Harvard University Press, Sep. 1991.

[5] R. Thrall, W. Lucas. N-person games in partition function form. Naval Research Logistics Quarterly, vol. 10, pp. 281–298, 1963.

[6] R. Myerson. Graphs and cooperation in Games. Mathematics of Operations Research vol. 2, pp. 225–229, Jun. 1977.

[7] M. Jackson, A. Wolinsky. A strategic model of social and economic networks. Journal of Economic Theory, vol. 71, pp. 44–74, 1996.

[8] G. Owen. Game Theory, 3rd edition. London, UK: Academic Press, Oct. 1995.

[9] T. Alpcan, T. Basar. A globally stable adaptive congestion control scheme for Internet-style networks with delay. IEEE/ACM Trans. on Networking, vol. 13, pp. 1261–1274, Dec. 2005.

[10] X. Deng , C. Papadimitriou. On the complexity of cooperative solution concepts. Mathematics of Operations Research, 19(2):257–266, 199.

[11] Y. Matsui, T. Matsui. NP-completeness for calculating power indices of weighted majority games. Theoretical Computer Science, 263(1-2):305–310, 2001.

[12] Y. Matsui, T. Matsui. A survey of algorithms for calculating power indices of weightedmajority games. *Journal of the Operations Research Society of Japan*, 43(1):71–86, 2000

[13] Francesco Scarcello, Gianluigi Greco. *Journal of Artificial Intelligence Research*, 49 (2014). Mechanisms for Fair Allocation Problems: No-Punishment Payment Rules in Verifiable Settings.

[14] G. Greco, F. Lupia, F. Scarcello. Structural tractability of shapley and banzhaf values in allocation games. In *Proc. of IJCAI'15*, pages 547–553, 2015.

[15] Gottlob, Leone, Scarcello. Robbers, marshals, and guards: game theoretic and logical characterization of hypertree width. *Journal of Computer and System Sciences*, 66, 2003.

[16] G. Gottlob, G. Greco, Z. Miklòs, F. Scarcello, T. Schwentick. Tree Projections: Game Characterization and Computational Aspects. Graph Theory, Computational Intelligence and Thought, pp. 217–226, 2009.

[17] G. Gottlob, M. Grohe, N. Musliu, M. Samer, F. Scarcello. Hypertree Decompositions: Structure, Algorithms, and Applications. In Proc. of WG'05, 2005.

[18] Georg Gottlob, Nicola Leone, Francesco Scarcello. Hypertree Decompositions: A Survey. Mathematical Foundations of Computer Science 2001, Volume 2136 of the series Lecture Notes in Computer Science pp 37-57.

[19] Georg Gottlob, Gianluigi Greco, Nicola Leone, Francesco Scarcello. Hypertree Decompositions: Questions and Answers. In *Proceedings of PODS 2016*, 57–74, 2016

[20] G. Greco, F. Scarcello. Fair division rules for funds distribution: The case of the italian research assessment program (vqr 2004-2010). *Intelligenza Artificiale*, 7(1):45–56, 2013.

[21] G. Greco, F. Scarcello. Mechanisms for fair allocation problems: No-punishment payment rules in verifiable settings. *J. Artif. Intell. Res. (JAIR)*, 49:403–449, 2014.

[22] H. Aziz, B. de Keijzer. Shapley meets shapley. In *Proc. of STACS'14*, pp. 99–111.

[23] Y. Bachrach, Y.S. Rosenschein. Power in threshold network flow games, pp. 106–132, 2009.

[24] X. Deng, C.H. Papadimitriou. On the complexity of cooperative solution concepts. *Mathematics of Operations Research*, 19:257–266, May 1994.

[25] D. Liben-Nowell, A. Sharp, T. Wexler, K. Woods. Computing Shapley value in supermodular coalitional games. In *Proc. of COCOON'12*, pages 568–579. 2012.

[26] F. Maniquet. A characterization of the Shapley value in queueing problems. *Journal of Economic Theory*, 109(1):90–103, 2003.

[27] D. Mishra, B. Rangarajan. Cost sharing in a job scheduling problem. *Social Choice and Welfare*, 29(3):369–382, October 2007.

[28] H. Moulin. An application of the Shapley value to fair division with money. *Econometrica*, 60(6):1331–49, November 1992.

[29] H. Nagamochi, Dao-Zhi Zeng, Naohisa Kabutoya, Toshihide Ibaraki. Complexity of the Minimum Base Game on Matroids. *Math. Oper. Res.*, 22(1):146–164, 1997.

[30] A. Iera, L. Militano, L. Romeo, F. Scarcello. Fair Cost Allocation in Cellular-Bluetooth Cooperation Scenarios. *EEE Transactions on Wireless Communications.*, 10(8):2566–2576, 2011.

[31] R. Pichler, S. Skritek. Tractable counting of the answers to conjunctive queries. *Journal of Computer and System Sciences*, 79(6):984–1001, 2013.

[32] N. Robertson, P. Seymour. Graph minors iii: Planar tree-width. *Journal of Combinatorial Theory, Series B*, 36(1):49–64, February 1984.

[33] L. S. Shapley. A value for n-person games. *Contributions to the theory of games*, 2, 1953.

[34] Bruno Courcelle. Graph rewriting: An algebraic and logic approach. In Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B), pages 193–242. Elsevier, 1990.

[35] B. Courcelle. The Monadic Second-Order Theory of Graphs. Recognizable Sets of Finite graphs. *Information and Computation*, 85:12–75, 1990.

[36] B. Courcelle. On the model-checking of monadic second-order formulas with edge set quantifications. *Discrete Applied Mathematics*, 160(6):866–887, 2012.

[37] B. Courcelle, I. A. Durand. Tractable constructions of finite automata from monadic second-order formulas, 2010. Presented at Logical Approaches to Barriers in Computing and Complexity, Greifswald, Germany.

[38] B. Courcelle, I. A. Durand. Verifying monadic second-order graph properties with tree automata. In 3rd European Lisp Symposium, pages 7–21, 2010. Informal proceedings edited by C. Rhodes.

[39] B. Courcelle, J. Engelfriet. Graph Structure and Monadic Second Order Logic: A Language Theoretic Approach. Number 138 in Encyclopedia of Mathematics and its Applications. Cambridge University Press, June 2012.

[40] B. Courcelle and M. Mosbah. Monadic second-order evaluations on tree-decomposable graphs. Theor. Comput. Sci., 109(1-2):49–82, 1993.

[41] Joachim Kneis, Alexander Langer, Peter Rossmanith. Courcelle's Theorem - A Game-Theoretic Approach. CoRR abs/1104.3905 (2011).

[42] Alexander Langer, Felix Reidl, Peter Rossmanith, Somnath Sikdar. Practical algorithms for MSO model-checking on tree-decomposable graphs. Computer Science Review 13-14: 39-74 (2014).

[43] Alexander Langer, Felix Reidl, Peter Rossmanith, Somnath Sikdar. Evaluation of an MSO-Solver. ALENEX 2012: 55-63

[44] Robert Ganian, Petr Hlinený, Alexander Langer, Jan Obdrzálek, Peter Rossmanith, Somnath Sikdar. Lower Bounds on the Complexity of MSO1 Model-Checking. CoRR abs/1109.5804 (2011)

[45] G. Gottlob, C. Koch. Monadic datalog and the expressive power of languages for web information extraction. J. ACM, 51(1):74–113, 2004.

[46] M. Elberfeld, A. Jakoby, T. Tantau. Logspace versions of the theorems of Bodlaender and Courcelle. *In Proc. of FOCS'10, pp. 143-152, 2010*

[47] H.L. Bodlaender, F.V. Fomin. A Linear-Time Algorithm for Finding Tree Decompositions of Small Treewidth. *SIAM Journal on Computing*, 25(6), pp. 1305–1317,1996.

[48] R.G. Downey, M.R. Fellows. Parametrized complexity Springer, 1999

[49] Bi Lim, Fatima Zahra Moataz,Nicolas Nisse Minimum Size Tree-Decompositions. Electronic Notes in Discrete Mathematics 50: 21-27 (2015).

[50] S. Arnborg, J. Lagergren, and D. Seese Easy problems for tree-decomposable graphs. J. Algorithms 12(2):308–340, 1991.

[51] Alexander Langer Fast algorithms for decomposable graphs. RWTH Aachen University 2013.

[52] Martin J. Osborne and Ariel Rubinstein. *A Course in Game Theory*. The MIT Press, Cambridge, MA, USA, July 1994.

[53] John F. Banzhaf. Weighted Voting Doesn't Work: A Mathematical Analysis. *Rutgers Law Rev.*, 19:317–343, 1965.

[54] Gianluigi Greco and Francesco Scarcello. Counting solutions to conjunctive queries: structural and hybrid tractability. In *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS'14, Snowbird, UT, USA, June 22-27, 2014*, pages 132–143, 2014.

[55] S. Shaheen Fatima and Michael Wooldridge and Nicholas R. Jennings. A linear approximation method for the Shapley value. Artif. Intell. 172(14): 1673-1699 (2008).

[56] S. Shaheen Fatima, Michael Wooldridge, Nicholas R. Jennings. A randomized method for the shapley value for the voting game. AAMAS 2007: 157

[57] Youngsub Chun. Consistency and monotonicity in sequencing problems. Int. J. Game Theory 40(1): 29-41 (2011).

[58] Javier Castro, Daniel Gómez, Juan Tejada. Polynomial calculation of the Shapley value based on sampling. Journal Computers and Operations Research archive Volume 36 Issue 5, May, 2009.

[59] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963

[60] Sasan Maleki, Long Tran-Thanh, Greg Hines, Talal Rahwan, and Alex Rogers. Bounding the estimation error of sampling-based shapley value approximation with/without stratifying. *CoRR*, abs/1306.4265, 2013.

[61] Herve Moulin. An application of the Shapley value to fair division with money. *Econometrica*, 60(6):1331–49, November 1992.

[62] François Maniquet. A characterization of the Shapley value in queueing problems. *Journal of Economic Theory*, 109(1):90–103, 2003.

[63] Debasis Mishra and Bharath Rangarajan. Cost sharing in a job scheduling problem. *Social Choice and Welfare*, 29(3):369–382, October 2007.

[64] Roger Myerson and Mark A Satterthwaite. Efficient mechanisms for bilateral trading. *Journal of Economic Theory*, 28:265-281, 1983.

[65] Bachrach, Y., Markakis, V., Procaccia, A. D., Rosenschein, J. S., & Saberi, A. Approximating power indices. In The seventh international joint conference on autonomous agents and multiagent systems (AAMAS 2008), 12–16 May, Estoril, Portugal (pp. 943–950).

Appendice A

Grafi quasi-aciclici e Formule Monadiche del Secondo Ordine

In questa appendice richiamiamo i concetti di base relativi alla nozioni di *treewidth* e *tree decomposition* ed alle *formule monadiche del secondo ordine (MSO)*.

A.1 Treewidth: una misura di ciclicità dei grafi

Vi sono diverse metodologie per misurare la distanza di un grafo da un albero. La tecnica della treewitdh è una della più importanti e studiate ed è stata usata in diversi domini applicativi, come descritto nei lavori [19], [17], [16], [18].

A.1.1 Definizione formale

Per treewidth s'intende una misura della possibilità di scomporre un grafo, in modo ricorsivo, in sotto grafi più piccoli garantendo una sovrapposizione minima (piccoli separatori). Molti problemi $NP - hard$ possono essere risolti in tempo polinomiale, spesso lineare, se la treewidth del grafo è limitata (dal punto di vista pratico, se è sufficientemente piccola) [19].

Una *tree decomposition* (T, X) di un grafo $G = (V, E)$ è una coppia $\langle T, X \rangle$ dove $T = (I, F)$ è un albero e X è una funzione $X : I \to 2^V$ che assegna ai nodi $i \in I$ un insieme di vertici $X(i) \subseteq V$ in modo da soddisfare tre vincoli. Per evitare confusione nel linguaggio, con il termine $vertice$ si farà sempre riferimento ai vertici di G, e con il termine *nodi* ai nodi di T.

- Ogni vertice v di G occorre in almeno un nodo di T, cioè esiste $i \in I$, tale che $v \in X(i)$. Si ha inoltre si ha $\bigcup_{i \in I} X(i) = V$.
- Per ogni ramo $\{u, v\} \in E$ esiste in T un nodo $i \in I$ tale che $\{u, v\} \subseteq X(I)$.

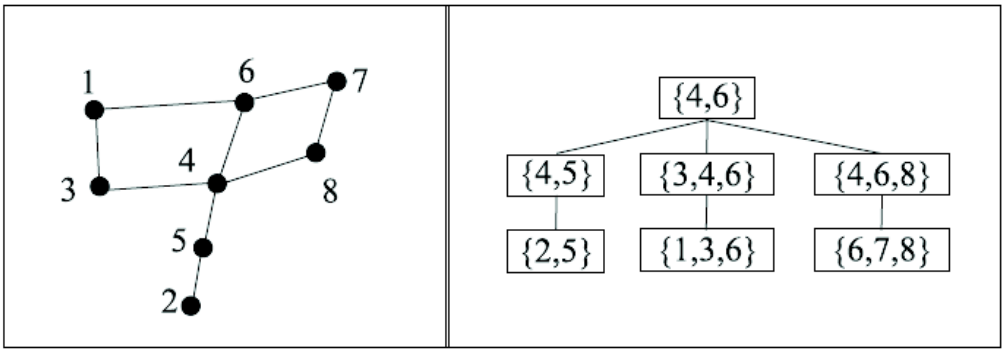


Figura A.1: Esempio 1

- Per ogni nodo v di G l'insieme $\{i \in I | v \in x(i)\}$ definisce un sottoalbero (connesso) di T. Equivalentemente, vale che $X(i) \cap X(k) \subseteq X(j)$ per ogni nodo j che si trova sul cammino che va dal nodo i al nodo k in T.

Detto questo è possibile definire diversi concetti:

- **Width**: L'ampiezza di una *tree decomposition* è pari a $max_{i \in I}|X(i)| - 1$
- **Treewidth**: La treewidth di un grafo G si indica con $tw(G)$ ed è la più piccola width tra tutte le sue possibili decomposizioni. Si noti che il concetto di treewidth è la generalizzazione del concetto di aciclicità definito su un grafo. Infatti, un grafo G è aciclico se e solo se $tw(G) = 1$.

Esempio A.1. *La treewidth del grafo G riportato nella figura A.1 è 2. Esso è infatti ciclico, per cui $tw(G) > 1$, ma ha una decomposizione di width 2, riportata nella figura di fianco, che testimonia $tw(G) \leq 2$.*

A.1.2 Il gioco delle guardie e del ladro

Una interessante caratterizzazione del concetto di treewidth, dovuta a Robertson e Seymour, consiste nella definizione del concetto attraverso il gioco delle guardie e del ladro. Nel gioco in questione sono due le parti in causa: un ladro che può muoversi solo sui vertici di un grafo $G = (V, E)$ sfruttando gli archi e i poliziotti che possono muoversi su tutti i nodi del grafo G dall'alto con un elicottero. Sia i poliziotti che il ladro possono controllarsi l'un l'altro come in un caso reale.

L'obiettivo dei poliziotti è quello di catturare il ladro; l'obiettivo del ladro è quello di sfuggire alla cattura. Al ladro è permesso di muoversi velocemente da un nodo ad un altro del grafo utilizzando un arco che li collega, i poliziotti utilizzando ciascuno un elicottero non sono vincolati a muoversi lungo gli archi del grafo, ma a loro discapito vi è il fatto che i loro movimenti sono molti più lenti di quelli del ladro. Inoltre, l'unico modo in

cui i poliziotti possono catturare il ladro è quello di far atterrare un elicottero sul vertice dove si trova il ladro. Se il ladro e il poliziotto si trovano sullo stesso nodo il ladro non può sfuggire alla cattura. Il ladro può vedere i poliziotti mentre atterrano su di un nodo e spostarsi velocemente su quello adiacente e sfuggire alla cattura prima che i poliziotti siano effettivamente atterrati. L'unico modo per i poliziotti di catturare il ladro è quello di bloccargli tutte le vie di fuga. In questo scenario il concetto di treewidth del grafo G è rappresentato dal numero minimo di poliziotti richiesto per catturare il ladro, meno uno. Si assume che i grafi trattati abbiano almeno due vertici, altrimenti solo un poliziotto è sufficiente a catturare il ladro. Se il grafo è connesso e ha almeno due vertici e vi è solo un poliziotto, questo non potrà mai catturare il ladro, perché quest'ultimo può sempre spostarsi su di un vertice dove non c'è il poliziotto.

Nel caso in cui la struttura dei nodi sia un albero due poliziotti sono necessari e sufficienti per eseguire la cattura: uno atterra su un vertice u arbitrario dell'albero T costringendo il ladro a spostarsi da una sotto struttura T_i a una T_u. Il secondo poliziotto si sposta sull'unico nodo adiacente a u in T_i e il gioco continua con i poliziotti che scambiano il loro ruolo spingendo in modo efficace il ladro fino ad un nodo foglia dove vi rimane intrappolato.

Si può osservare che per un grafo che ha un ciclo C_n con un numero di vertici maggiore di tre, due poliziotti non sono più sufficienti. In tale scenario il ladro può usare la seguente strategia: per ogni coppia di vertici u, v scelti dai due poliziotti, al ladro conviene spostarsi in un vertice della componente di $C_n \setminus \{u, v\}$ più grande, prima che il secondo poliziotto atterri. Inoltre per $n \geq 3$, esiste sempre un vertice libero in cui il ladro può sempre sfuggire alla cattura. Con tre poliziotti lo scenario cambia e i poliziotti possono procedere come segue: due poliziotti possono occupare i vertici diametralmente opposti della componente C_n che contiene il ciclo, cosi il ladro è costretto a muoversi in un percorso $P \in C_n \setminus \{u, v\}$ e il terzo poliziotto può posizionarsi su un vertice al centro del commino P. Questa strategia può essere ripetuta dai poliziotti dimezzando il cammino del ladro ad ogni iterazione finché il ladro rimane intrappolato. Se il grafo presenta una struttura tipo *clique* con n vertici, per catturare il ladro sono necessari n poliziotti.

Nel caso che la struttura G è un albero, la strategia dei poliziotti si basa sul fatto che ogni vertice interno all'albero è un vertice di taglio, infatti un albero può essere scomposto nei suoi vertici di taglio. La definizione di treewidth replica essenzialmente questa idea dove il concetto di vertice di taglio è esteso al concetto di insieme di vertici di taglio. Rimuovere un insieme di vertici di taglio in un grafo equivale a decomporre il grafo in piccoli pezzi, similmente a quanto accade quando viene rimosso un vertice all'interno di un albero. Questi sotto grafi risultanti possono essere a loro volta scomposti ricorsivamente.

A.1.3 Complessità del problema di determinare la treewidth

Il concetto di Tree-decomposition è alla base di molti algoritmi di programmazione dinamica per risolvere i problemi su grafo. Poiché la complessità di diversi algoritmi per

risolvere problemi su grafo è generalmente esponenziale nella dimensione della decomposizione, molti lavori di ricerca sono stati dedicati a calcolare la tree decompositions con taglia piccola, quindi è di interesse definire algoritmi in grado di minimizzare la taglia della decomposizione.

Il cuore della questione consiste nel calcolare una decomposizione di taglia minima, che è dimostrato essere un problema NP-completo. Tuttavia, per un fissato numero naturale k, verificare che un dato grafo G abbia una decomposizione di width al più k è fattibile in tempo lineare, in $O(k^{k^3}n)$, come descritto nel lavoro [47]. Inoltre Elberfeld ha dimostrato che tale problema è risolvibile in spazio logaritmico [46].

Si noti che la costante moltiplicativa dell'algoritmo lineare di Bodlaender è legata in modo esponenziale a k. In effetti questi algoritmi richiedono nella pratica molto tempo anche per $k = 3$. Ad oggi la progettazione di un algoritmo efficiente che calcola una tree decomposition ottima è un problema aperto, ma nel lavoro di Bodlaender e Koster sono elencati una serie di algoritmi implementati e testati. Purtroppo, tutti questi algoritmi risultano in pratica inutilizzabili per istanze reali. L'attenzione si è concentrata quindi sul definire tecniche euristiche in grado di calcolare dei buon upper bound per il problema in questione. L'utilizzo di un albero di decomposizione non ottimale aumenta notevolmente il tempo di esecuzione dell'algoritmo, ma non impatta sulla correttezza della soluzione restituita.

Esempio A.2. *Nella figura A.2 sono riportati i grafi bipartiti delle componenti con 15 e 23 autori, descritti nel Capitolo 6, e una loro decomposizione ad albero. Inoltre è stata calcolata la tree decomposition per le diverse componenti usate negli esperimenti precedenti, ed i risultati sono riportati nella tabella A.1*

Tabella A.1: Tabella riepilogo Treewidth

Taglia componenti	Treewidth
15	3
23	8
26	2
30	4
40	6
50	3
685	51

A.1.4 Tree decomposition e problemi di decisione

Le Tree decomposition e gli algoritmi polinomiali basati su di esse sono strumenti efficaci per identificare isole di trattabilità e risolvere in modo efficiente problemi che sono in

generale NP-hard. In particolare, la nozione di treewidth è alla base di importanti metateoremi, come quello di Courcelle, in cui si afferma che qualsiasi problema esprimibile in MSO (logica monadica del secondo ordine) su strutture con treewidth limitata può essere risolto in tempo lineare. Molti problemi sono esprimibili facilmente in logica MSO, quindi, il teorema di Courcelle si rivela essere uno strumento utile per ottenere risultati sulla trattabilità di problemi difficili per istanze aventi una struttura a grafo con bassi livelli di ciclicità.

A.2 Logica monadica del secondo ordine

La logica monadica del secondo ordine (MSO) è un sottoinsieme della logica del secondo ordine in cui, oltre a variabili del primo ordine $x_1, x_2, \ldots$, possono occorrere variabili unarie del secondo ordine $X_1, X_2, \ldots$, che rappresentano insiemi. Le formule MSO definite su un vocabolario τ possono essere composte da:

- Formule atomiche
- Connettivi $(\neg, \vee, \wedge, \rightarrow, \leftrightarrow)$,
- Quantificatori su variabili del primo ordine della forma $(\exists x, \forall x)$
- Quantificatori su variabili del secondo ordine della forma $(\exists X, \forall X)$
- variabili libere o vincolate.

A.2.1 Logica MSO e problemi su grafo

Molti problemi su grafi possono essere codificati in logica MSO e risolti in tempo polinomiale se il grafo presenta una treewidth limitata. Per i grafi, consideriamo le due seguenti tipologie di formule MSO:

MSO_1: dato un vocabolario τ, le formule atomiche espresse in MSO_1 sul vocabolario τ sono del tipo $x = y$, $R(x_1, ..., x_k)$, $x \in X$, dove $x, y, x_1, ..., x_k$ sono variabili relative ai vertici, X è una variabile che rappresenta un insieme di vertici, il simbolo " $=$ " indica uguaglianza, $R \in \tau$ è una relazione di arità k. Le formule di MSO_1 hanno la medesima struttura delle formule FO, ma in più è possibile esprimere quantificatori del secondo ordine sui vertici di un grafo in input, del tipo $\exists X$. Nelle formule MSO_1 il grafo è rappresentato mediante un modello di adiacenza. L'esempio A.3 riporta una formulazione del problema della 3-colorabilità su grafi in logica MSO_1

Esempio A.3. *Il problema della 3-colorabilità consiste nel determinare una colorazione dei vertici di un grafo con soli 3 colori, tale che nessuna coppia di vertici adiacenti abbia lo stesso colore. La formula in MSO_1 è la seguente:*

$$3 - colorability \equiv \exists X_1 \exists X_3 \exists X_3 \forall x \forall y : part(X_1, X_2, X_3) \wedge$$
$$x \in X_1 \wedge y \in X_1 \vee x \in X_2 \wedge y \in X_2 \vee x \in X_3 \wedge y \in X_3 \Rightarrow \neg edge(x, y)$$

Il predicato $part$ è vero quando X_1, X_2, X_3 è un partizionamento dell'insieme dei vertici, che può essere espresso come segue:

$$\begin{aligned} part(X_1, X_2, X_3) \equiv \forall z : (z \in X_1 \vee z \in X_2 \vee z \in X_2) \\ \wedge[(z \in X_1 \Rightarrow z \notin X_2 \wedge z \notin X_3) \\ \wedge(z \in X_2 \Rightarrow z \notin X_1 \wedge z \notin X3) \wedge \\ (z \in X_3 \Rightarrow z \notin X_1 \wedge z \notin X_2)] \end{aligned}$$

Dato un grafo $G(V, E)$ dove $n = |V|$ è il numero di nodi, il metodo basato su brute force per determinare tutte le soluzioni del problema della 3-colorabilità va a considerare ognuno dei possibili 3^n insiemi di nodi e controlla per ciascuno di essi se questo è una colorazione ammissibile per il grafo G. Come vedremo, tale problema può essere risolto in tempo lineare utilizzando tecniche basate su logica MSO, se il grafo G presenta una treewidth limitata.

MSO_2: La logica MSO_2 estende logica la MSO_1 assumendo la relazione binaria $inc(x, e)$ che esprime la relazione di incidenza tra x, che è una variabile riferita ad vertice, ed e, che è una variabile riferita ad un arco. I problemi formalizzati in MSO_2 richiedono quindi che il grafo G in input sia rappresentato con un modello di incidenza.

In MSO_2 la quantificazione può essere espressa non solo sui vertici ma anche sugli archi del grafo in input, in particolare la quantificazione del secondo ordine riguarda qui anche insiemi di archi. Questo linguaggio è strettamente più potente di MSO_1 in quanto esistono dei problemi che è possibile esprimere in MSO_2 ma che non possono essere espressi in MSO_1, ad esempio il problema di verificare se un grafo presenta un ciclo hamiltoniano.

Esempio A.4. *Un grafo G è hamiltoniano se e solo se gli archi G possono essere suddivisi in due gruppi, rosso e blu, tali che ogni vertice v ha esattamente due archi incidenti rossi, e il sotto grafo indotto dagli archi rossi ricopre G. La formula riportata in MSO_2 permette di verificare se in un grafo esiste un ciclo hamiltoniano. Nella formula vi sono due variabili R e B che rappresentano rispettivamente l'insieme degli archi rossi e l'insieme degli archi blu.*

- $Hamiltonian \equiv \exists R \exists B \forall u \forall v : part(R, B) \wedge deg(u, R) = 2 \wedge span(u, v, R)$
- $part(R, B) \equiv [\forall e : (e \in R \vee e \in B) \wedge \neg(e \in R \wedge e \in B)$

- $deg(u, R) \equiv [\exists e_1, e_2 : (e_1 \neq e_2 \wedge inc(e_1, u) \wedge inc(e_2, u) \wedge e_1 \in R \wedge e_2 \in R)] \wedge \neg[\exists e_1, e_2, e_3 : (e_1 \neq e_2 \neq e3 \wedge (inc(e_i, u) \wedge e_i \in R \ \forall i \in \{1, 2, 3\}))]$
- $span(u, v, R) \equiv \forall V, W : (part(V, W) \wedge u \in V \wedge v \in W) \Rightarrow (\exists e, x, y : (inc(e, x) \wedge inc(e, y) \wedge x \in V \wedge y \in W \wedge e \in R))$

A.2.2 Teorema di Courcelle ed estensioni

Teorema A.2.1 ((Courcelle [34])). *Ogni proprietà su grafi esprimibile in MSO_2 è risolvibile in tempo lineare sulla classe dei grafi che hanno treewidth limitata.*

Il teorema di Courcelle è stato esteso da Arnborg, Lagergren, e Seese [50], che hanno dimostrato che la trattabilità delle proprietà su grafi di treewidth limitata può essere estesa ad una più ampia classe di problemi di decisione, nonché a problemi di ottimizzazione e di conteggio.

A tale scopo questi autori propongono un'estensione della logia MSO, chiamata Extended MSO $(EMSO)$. Tale estensione offre delle funzioni per mappare gli insiemi nel dominio dei numeri interi o reali al fine di effettuare dei confronti con qualsiasi numero fornito in input, oppure semplicemente contare il numero di soluzioni. Con questo tipo di estensione è possibile esprimere problemi di decisione che presentano numeri interi come parte del loro input, problemi di ottimizzazione e problemi di conteggio. Una funzione di valutazione l definita su un insieme di variabili $H = \{h_1, ..., h_k\}$ è una funzione $l : R^k \rightarrow R$ basata sull'utilizzo di operatori aritmetici $-, +, \times$.

Teorema A.2.2. *(Arnborg et al [50]) Sia P un problema definibile in logica $EMSO$ o un problema di ottimizzazione definibile $EMSO$ con funzioni peso delimitate da una costante, e un intero $k \in N$, allora si può risolvere un problema P su grafo $G = (V, E)$ di ordine $n := |V|$ con treewidth al massimo k in tempo $O(f_p(k) \cdot n)$.*

Teorema A.2.3. *(Arnborg et al. [50]) Sia P un problema di conteggio definibile su MSO o un problema di ottimizzazione anch'esso definibile in $EMSO$ con funzioni peso intere e un intero $k \in N$, allora si può risolvere P su un grafo $G = (V, E)$ di ordine $n := |V|$ e treewidth al massimo k in un tempo $O(f_p(k)poly(n))$.*

Si noti che la complessità di trattare una formula in logica $EMSO$ è maggiore rispetto alle formule in logica MSO, che per grafi di treewidth limitata garantivano la trattabilità lineare.

A.3 Sequoia: un risolutore per formule MSO

Sequoia è un tool per la risoluzione di problemi su grafo che presentano una treewidth limitata, definiti in logica EMSO. Tale tool è descritto nel lavoro [51] ed è stato rilasciato

come software open source. Sequoia è scritto in $C++$, ed offre oltre ad un risolutore di formule MSO anche una serie di piccoli programmi di utilità per il calcolo della treewidth e una collezione di esempi riguardanti diversi problemi noti su grafo. Allo stato dell'arte supporta solo grafi non orientati ed etichettati, non supporta vocabolari arbitrari.

Il tool per la risoluzione delle formule logiche utilizza la programmazione dinamica implementata dalla classe DynProgSolver che è il cuore dell'intero progetto. Tale classe si occupa del caricamento del grafo di input e dell'albero della decomposizione e, qualora quest'ultimo non fosse presente, ne provvede alla generazione. La classe si occupa di navigare l'albero di decomposizione in modo bottom-up, richiamando i metodi virtuali do-leaf(), do-introduce(), do-forget(), do-join(), e do-root() che gli permettono di recuperare ed elaborare le informazioni relative al nodo in questione. Per la navigazione dei nodi della decomposizione, la classe Dyn- ProgSolver (classe astratta) necessita dell'implementazione di questi metodi. Attualmente una sua implementazione concreta è offerta dalla classe SequoiaSolver.

A.3.1 MSO Solver

SequoiaSolver è la classe principale che estende DynProgSolver e ne implementa i metodi astratti. Dato un problema P definito sulla struttura $struct(\tau)$, una formula $\varphi \in MSO(\tau')$ ed un semianello $E = (U_E, \oplus, \otimes, \hat{0}, \hat{0})$, dove gli operatori $\oplus$ e $\otimes$ possono essere calcolati in tempo costante e avendo inoltre i dati: $s \in \{1,0\}$, una costante $t \in N$, e un insieme Θ di operatori etichettati sulla struttura τ, l'algoritmo genera la decomposizione $\mathfrak{U}$ della struttura τ, e il suo albero di sintattico T. L'algoritmo inoltre genera la struttura $\hat{\mathfrak{U}}$ in cui $universe(\mathfrak{U}) = universe(\hat{\mathfrak{U}})$, dove gli operatori nulli sono i simboli terminali ed i restanti operatori sono usati per generare le regole $G \to G \oplus G$. Più formalmente, un albero sintattico è un albero ordinato dove tutti i nodi u di T sono marcati con un operatore, di arietà $r = |children(u)|$. L'algoritmo prosegue con una operazione di riduzione, associando ad ogni nodo $i \in V(T)$ dell'albero sintattico una tabella S_i contenente le soluzioni parziali e i valori val_i corrispondenti, ovvero gli elementi del semianello. Terminata questa fase, l'algoritmo controlla che la formula φ sia verificata sulla struttura, calcolandone quindi il risultato.

Nel progetto Sequoia vi sono classi che rappresentano i simboli, i vocabolari e le formule. Un vocabolario è un contenitore di istanze di classe Simbolo di arietà arbitraria di cui ne esistono diverse estensioni per specifiche costanti (ConstantSymbol per simboli di arietà zero oppure UnarySymbol per quelli di unaria). Ad ogni simbolo è associato un concetto di profondità che rappresenta la posizione del simbolo nel vocabolario.

Le formule MSO sono rappresentate dalla classe Formula di cui esistono tre estensioni che rappresentano i quantificati QFormula, le formule negate NegatedFormula e le formule booleane BoolCombFormula. Le istanze di queste classi sono combinate tra

loro al fine di rappresentare la formula fornita in input. Le classi QFormula NegatedFormula rappresentano formule che operano con una sola sottoformula, mentre la classe BoolCombFormula opera un numero arbitrario di sottoformule.

Il risolutore MSO definisce tre vocabolari di cui uno è determinato direttamente dalla struttura di input G e i restanti svolgono un ruolo di appoggio nell'algoritmo di programmazione dinamica. I tre vocabolari sono:

- **vocabolario di base**: Il vocabolario di base è un vocabolario che non cambia durante la valutazione ed è composto da $\tau'_{Graph} = \tau_{Graph} \cup \{=\} \cup \{S_1, ..., S_n\}$, dove $\tau_{Graph} = (adj)$, e contiene le relazioni di adiacenza tra i vari nodi del grafo, dove $S_1, ..., S_n$ sono i vertici etichettati del grafo G.

- **vocabolario della tree decomposition**: Il vocabolario della tree decomposition τ''_{Graph} è un vocabolario: $\tau''_{Graph} = \tau'_{Graph} \cup \{t_0, ..., t_k\}$ dove k è la taglia della tree decomposition e $t_0, ..., t_k$ sono i nodi terminali della decomposizine.

- **vocabolario della formula**: Il vocabolario della formula τ^{φ}_{Graph} è il set di variabili della formula φ interpretate sul vocabolario τ''_{Graph}

Ad ogni assegnazione $(U_1; ....; U_l)$ delle variabili libere $(R_1; ...; R_l)$ =$\tau^{\varphi}_{Graph} \setminus \tau^{\text{“}}_{Graph}$ la formula φ viene verificata e l'assegnazione viene salvata in un elemento nel semianello. Costruito il semianello, la classe SequoiaSolver, in relazione alla tipologia di valutazione richiesta, effettua la navigazione dell'albero procedendo a seconda del nodo incontrato come segue:

- **Nodo foglia**: Per i nodi foglia della decomposizione, l'algoritmo trasforma la formula φ in una struttura vuota, con un solo assegnamento $(\emptyset, \ldots, \emptyset)$ per tutte le variabili libere della formula

- **Nodo interni**: Per tutti i nodi i interni, l'algoritmo itera su tutti i figli del nodo i e valuta la formula. Terminata la valutazione, l'algoritmo combina i risultati.

- **Nodo Radice**: Quando l'algoritmo arriva sul nodo radice converte i risultati dei livelli precedenti e li restituisce.

A.3.2 Sistema di Valutazione delle formule

L'interfaccia Evaluation fornisce una rappresentazione dei concetti di semianello e omomorfismo e maschera la logica di questi all'algoritmo di programmazione dinamica. Il SequoiaSolver utilizza un factory che, in base ai parametri di input, crea un tipo di valutazione. Per garantire la piena flessibilità ed estensibilità, la classe SequoiaSolver mantiene solo dei puntatori generici di tipo const void* verso le soluzioni dei semianelli.

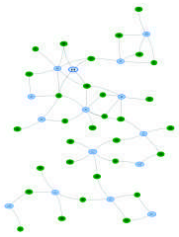
Le classi che implementano la strategia di valutazione possono accedere a queste informazioni tramite delle operazioni di cast.

L'interfaccia Evaluation specifica tre metodi:

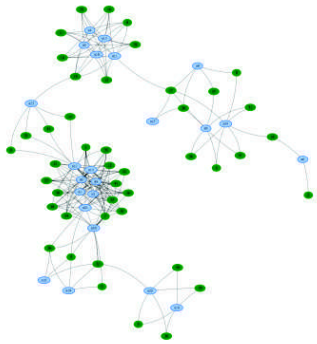
- **elem**: ritorna un assegnamento $(U_1, ..., U_l)$ per le variabili libere della formula φ.
- **add**: implementa l'operatore somma del semianello e presenta due parametri in ingresso e_1,e_2, dove e_1 e e_2 sono due assegnamenti.
- **mult**: implementa l'operatore prodotto del semianello e presenta tre parametri in ingresso e_1,e_2, dove e_1 e e_2 sono due assegnamenti mentre e_3 è l'intersezione dei due semianelli.
- **output solution**: implementa il metodo per mappare in stringa un assegnamento.

All'attuale stato dell'arte, il tool presenta le seguenti valutazioni:

- **BoolEvaluation**: è in grado di controllare se la struttura verifica la formula e ritorna un booleano;
- **WitnessEvaluation**: calcola una soluzione random per le variabili libere della formula;
- **MinCardEvaluation** e **MaxCardEvaluation**: calcolano rispettivamente la cardinalità minima e quella massima dei possibili assegnamenti;
- **MinCardSetEvaluation** e **MaxCardSetEvaluation**: restituiscono rispettivamente un assegnamento di cardinalità minima e di cardinalità massima;
- **Counting**: restituisce il numero di assegnamenti che soddisfano la formula. E' stato proposto e implementato in collaborazione con l'autore del tool Sequoia un'ulteriore valutazione;
- **CardCounting**: permette di valutare il numero di assegnazioni ammissibili che presentano una data cardinalità.



(a) $n = 15$



(b) $n = 23$

(c) $n = 23$

(d) $n = 23$

Figura A.2: Tree decomposition Componenti

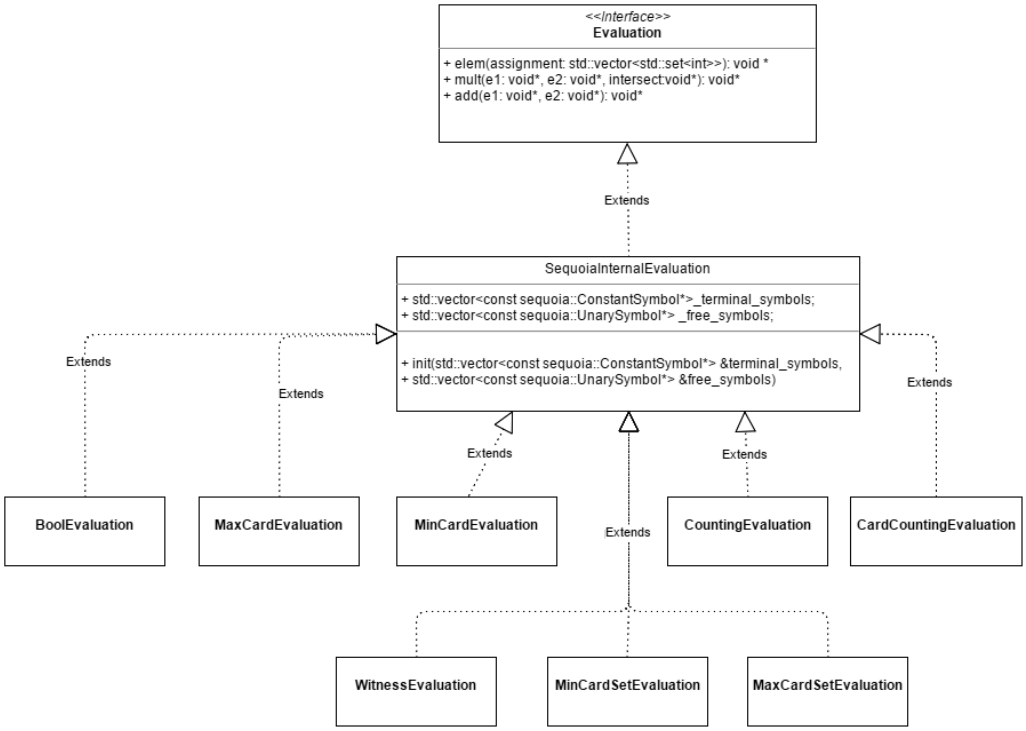


Figura A.3: Gerarchia delle Valutazioni Class

Ringraziamenti

Dopo anni di studio è arrivato il momento di chiudere un percorso. Il giorno è arrivato: scrivere queste frasi di ringraziamento è il tocco finale della mia tesi. È stato un periodo di profondo apprendimento, non solo a livello scientifico, ma anche personale. A tal punto colgo l'occasione per ringraziare le persone vicine, che sono state dei punti fermi durante questo percorso di dottorato.

Innanzitutto desidero ringraziare il prof. Francesco Scarcello per la fiducia, la disponibilità e l'umanità con cui mi ha accompagnato in questo percorso, lo ringrazio per i preziosi insegnamenti e per avermi incoraggiato ad andare avanti nei momenti di difficoltà. Desidero ringraziare il dott. Francesco Lupia per il suo sostegno e per la sua disponibilità a dirimere i miei dubbi durante questi anni di dottorato e per i consigli nella redazione di questo lavoro di tesi. Lavorare con entrambi è stata per me un'opportunità unica di crescita a livello culturale, professionale e personale. I ringraziamenti più grandi vanno ai miei genitori per i loro saggi consigli e la loro capacità di ascoltarmi e per essere sempre stati al mio fianco.

Un ringraziamento speciale va a mio fratello Emanuele che mi ha fortemente incoraggiato nello scegliere questo percorso e di raggiungere questo traguardo. Un grazie va a tutto lo staff di Artemat per l'affetto e per l'atmosfera serena e piacevole che accompagnano le mie giornate. Un grazie va anche agli amici di sempre Vincenzo, Vanessa, Gianni, Maria Rosaria, Andrea perché so che potrò sempre contare su di voi, come ho fatto finora.

Dulcis in fundo, grazie mio nonno, Angelo, persona coraggiosa, che attraverso la sua storia mi ha insegnato il valore del lavoro e della costanza e l'importanza dello studio e della cultura. Un sentito grazie a tutti!