Università degli Studi di Salerno
Facoltà di Scienze Matematiche Fisiche e Naturali

Tesi di Laurea Magistrale in
Informatica

A more efficient implementation of the subgraphs-world for the Glauber dynamics in the Ising model

Relatori
Prof. Vincenzo Auletta
Dott. Diodato Ferraioli

Candidato
Francesco Farina
Matricola 0522500282

Anno Accademico 2015-2016

Dediche e ringraziamenti

Ai miei genitori, a mia sorella Giovanna ed alla mia famiglia.

Ai miei fratelli e colleghi Marco e Daniele.

A tutti gli amici.

Al mio relatore Prof. Vincenzo Auletta ed al Dott. Diodato Ferraioli.

A te.

Indice

Capitolo 1

Introduzione

Un *sistema complesso* può essere visto come un sistema il cui comportamento complessivo risulta dall'interazione delle singole parti, ognuna con i propri obiettivi, spesso soggette ad influenze esterne. Esempi di tali sistemi si possono trovare in vari ambiti di ricerca: a partire dall'*economia*, con lo studio dei mercati, dalla *fisica*, con i sistemi di spin, passando per la *biologia* e lo studio dell'evoluzione, fino ad arrivare all'*informatica*, con lo studio di Internet e delle reti sociali. Questi sistemi sono stati studiati sempre in maniera indipendente, ma negli ultimi anni è stata messa in evidenza l'analogia tra i fenomeni che accadono al loro interno: pertanto è nato il bisogno di trovare uno strumento che fornisca un linguaggio universale comprensibile sia per le scienze naturali che sociali e che permetta di analizzare, quindi, sia la natura che la società.
Nel corso degli anni, la scienza ha sviluppato ben due linguaggi comuni, che si fondono per descrivere tali fenomeni: la *teoria dei grafi* e la *teoria dei giochi*. La teoria dei grafi [1] è una branca della matematica, nata nel 1700, che consente di descrivere le relazioni che intercorrono tra un insieme di oggetti. Il grafo è lo strumento attraverso il quale tali relazioni possono essere espresse ed organizzate. Esso consiste di oggetti chiamati *nodi* e relazioni tra coppie di questi oggetti detti *archi*; tale struttura gode di notevole versatilità: questa caratteristica, in unione alla completa pervasività della tecnologia in ogni ambito operativo umano, fa sì che sia possibile rilevare il grafo nei contesti più variegati, così da essere in grado di analizzare i comportamenti dei suoi nodi. Per modellare il comportamento dei singoli componenti viene utilizzata la *teoria dei giochi*, un campo che riesce ad abbracciare i sistemi complessi nei vari ambiti: economico, biologico, fisico, informatico, sociologico e così via.

La *teoria dei giochi* [1], tratta con un insieme di *giocatori*, ognuno dei quali possiede un insieme di possibili azioni che può intraprendere, dette anche *strategie*. Ciascun agente sceglie la propria strategia in accordo ad una funzione, detta *payoff*, che non dipende solo dalla strategia del giocatore stesso, ma anche dalle strategie scelte da tutti gli altri giocatori. Il modo in cui tali giocatori modificano le loro strategie in accordo alle scelte effettuate dagli altri, definisce la dinamica del gioco e descrive, quindi, come tale gioco evolve. Se la dinamica, dopo un certo periodo di tempo, raggiunge un punto fisso (cioè uno stato di stabilità), si dice che il gioco è in *equilibrio*.
La *teoria dei giochi* classica assume che ogni giocatore abbia una *conoscenza completa* del gioco (conosce tutti gli altri giocatori, i loro insiemi di strategie e i loro *payoff* e, in più sa che anche gli altri giocatori hanno le stesse informazioni); assume, inoltre, che ogni agente abbia una potere computazionale illimitato e, pertanto, riesca sempre a scegliere la strategia che massimizza la sua utilità. In questo modello *razionale*, l'evoluzione di un sistema viene modellata attraverso la *best response dynamic*, ed è possibile fare predizioni sul gioco utilizzando il ben noto *equilibrio Nash*. Tuttavia, in molti casi concreti l'assunzione fatta dalla *game theory* classica, può essere irrealistica: i sistemi complessi sono spesso influenzati da fattori ambientali che possono, a loro volta, influire sul modo con cui un giocatore sceglie la propria strategia; inoltre, i giocatori possono avere capacità computazionali limitate, e ciò rappresenta il principale limite soprattutto in sistemi informatici e sociali, e conoscenze limitate circa i fattori esterni che possono influenzare il gioco.
Pertanto, c'è bisogno di definire dinamiche che siano capaci di modellare la *bounded rationality* (conoscenza limitata), in cui i giocatori possano prendere decisioni sbagliate, e che portino auspicabilmente il sistema a raggiungere un *equilibrio* che esista sempre, sia unico e velocemente raggiungibile. La *Logit dynamics*, introdotta da Blume [2], modella le dinamiche di questo tipo. In una *Logit dynamics*, descritta in dettaglio nel Capitolo 3, ad ogni passo si sceglie a caso un giocatore per permettergli di modificare la propria strategia in accordo ad un parametro β che rappresenta il livello di razionalità (o conoscenza) e allo stato del sistema (le strategie attualmente scelte dagli altri giocatori). Tale modo di operare avviene nel rispetto di una regola detta *logit update rule*, che può essere vista come la versione "rumorosa" della classica *best response rule*, dove β rappresenta la tendenza verso le scelte che sono buone. Intuitivamente, valori piccoli di β rappresentano situazioni in cui i giocatori sono soggetti a forte rumore, oppure hanno una conoscenza molto limitata del gioco e, di conseguenza, scelgono le loro strategie in maniera quasi casuale; grandi valori di β, invece, rappresentano situazioni in cui i giocatori sono quasi sicuri di giocare la loro *best response*. Questo modello è

molto simile ad un altro modello utilizzato dai fisici per descrivere i *sistemi di particelle*, dove il comportamento di ogni singola particella è influenzato dalla temperatura: qui, alta temperatura significa bassa razionalità e bassa temperatura, invece, indica alta razionalità.
Come affermato in [2], è ben noto che tale dinamica definisce una *catena di Markov* sull'insieme di profili di strategie del gioco e, inoltre, è noto che esiste sempre un'unica *distribuzione stazionaria* verso cui la catena converge, indipendentemente dal profilo di partenza. È facile vedere che la catena di Markov in questione è *ergodica*, pertanto esiste un'unica distribuzione stazionaria π e, per ogni stato iniziale x, la distribuzione della catena al tempo t approccia a π man mano che t tende ad infinito (maggiori dettagli in 2.4.2 e in 3.3).
Consideriamo ora un *potential game*. Un gioco $G = ([n], S, \mathcal{U})$ è detto *potential game* se esiste una funzione $\Phi : S_n \times \cdots \times S_n \to \mathcal{R}$ tale che per ogni giocatore i e per ogni coppia di profili x ed y che differiscono solo per la posizione i, si ha che $u_i(x) - u_i(y) = \Phi(x) - \Phi(y)$.
In [2] è possibile vedere che se $\mathcal{G}$ è un *potential game* con funzione potenziale Φ, allora la catena di Markov descritta in precedenza è reversibile e la sua distribuzione stazionaria è data dalla *Gibbs measure*. In più, quando $\mathcal{G}$ è un potential game, la *Logit dynamics* è equivalente alla ben nota Glauber dynamics [3]; ciò significa che la *Logit dynamics* per $\mathcal{G}$ e la Glauber dynamics per la Gibbs distribution definiscono la stessa catena di Markov.
Grazie a tale analogia, ci si può riferire alla terminologia utilizzata dai fisici per indicare le quantità coinvolte; in particolare il parametro β è detto *inverse temperature* ed il fattore di normalizzazione presente nella distribuzione stazionaria (che sarà indicato con Z) è detto ***partition function***. Maggiori dettagli in 3.2.2.

Obiettivo del lavoro. L'obiettivo di questo lavoro di tesi è quello di proporre un algoritmo, applicabile in contesti del mondo reale, per la computazione della distribuzione stazionaria della catena di Markov relativa alla *Glauber dynamics* nel modello di Ising.
La pubblicazione che ha dato il via a questa linea di ricerca è stata quella di Jerrum e Sinclair in [4] nel 1993, che ha spostato il problema nell'ambito del *subgraphs-world*, fornendo un algoritmo polinomiale, ma non abbastanza scalabile da poter essere utilizzato in pratica. Per questo è stata necessaria un'attenta analisi del lavoro di Jerrum e Sinclair, così da poterne migliorare i bound temporali, rendendo l'esecuzione molto più efficiente, anche grazie al contributo apportato da Auletta, Ferraioli, Pasquale, Penna, Persiano in [5], integrato in questo lavoro. Quindi sono stati affinati i teoremi, i lemmi e le

assunzioni, i concetti assimilati, ottimizzati ed inglobati all'interno dell'algoritmo sviluppato.
Il lavoro svolto ha portato a raggiungere notevoli risultati. Infatti, l'algoritmo proposto, oltre al portare a termine l'esecuzione di esperimenti su reti di piccole-medie dimensioni in secondi, consente di computare la *partition function* anche su reti di medie-grandi dimensioni in tempi ragionevoli, cosa prima assolutamente fuori portata, basti considerare che il tempo d'esecuzione per uno stesso esperimento è stato abbattuto da ore a secondi.
Inoltre è stata verificata la bontà dell'algoritmo sviluppato con un'applicazione pratica nota come "*The Effectiveness of Advertising*" su grafi di medie dimensioni, in cui si va calcolare la magnetizzazione della rete a partire dalla sua struttura e dalla polarizzazione dei singoli nodi. In termini economici, questo si traduce nel poter prevedere il risultato della pubblicizzazione di due prodotti, ottenendo la preferenza finale della popolazione.

Related works. I primi lavori riguardanti la *Logit dynamics* hanno focalizzato la loro attenzione principalmente sul comportamento a lungo termine delle dinamiche considerate: Blume [2] ha mostrato che, per *coordination games* 2×2 e per *potential games* il comportamento a lungo termine del sistema è concentrato in uno specifico equilibrio Nash; Alós-Ferrer e Netzer [7] danno una caratterizzazione generale del comportamento a lungo termine di tali dinamiche per un'ampia gamma di classi di giochi.
Un numero considerevole di lavori si è dedicato ad analizzare il tempo che la dinamica impiega per raggiungere un equilibrio Nash, chiamato *hitting time*: tra questi è rilevante considerare il lavoro di Peyton Young [8] che ha esteso tali considerazioni per famiglie più generali di grafi.
Ci sono stati, inoltre, lavori che si sono focalizzati sulla descrizione di un ulteriore equilibrio nella *game theory* relativamente alle *Logit dynamics*: Ferraioli in [9] descrive un nuovo equilibrio chiamato *Logit equilibrium* e introduce un nuovo concetto chiamato *metastability*; Auletta, Ferraioli, Pasquale, Penna, Persiano in [5] descrivono la convergenza verso tale equilibrio della *Logit dynamics* per un *potential game*, mentre in [10], Auletta, Ferraioli, Pasquale, Persiano descrivono la convergenza verso la *metastability* di *coordination games*.
Il lavoro più recente [6] raggiunge l'obiettivo di computare la funzione di partizione Z in tempo polinomiale: l'intuizione è quella di far girare un'altra dinamica che goda di due importanti proprietà, ovvero la convergenza "rapida" alla distribuzione stazionaria ed il poter utilizzare tale stazionaria per calcolare quella di nostro interesse. La nuova dinamica in questione è quella proposta in [4] da Mark Jerrum ed Alistair Sinclair nel lavoro "Polynomial

time approximation algorithms for the Ising model", in cui presentano un algoritmo di approssimazione per il calcolo della partition function Z per il modello di Ising, un classico problema combinatoriale della fisica statistica; l'algoritmo proposto è un *fully polinomial randomised approimation scheme* (fpras) ed ha tempo d'esecuzione polinomiale. Tale articolo rappresenta il punto di partenza su cui si basa [6], ma nonostante l'approfondita analisi e le ottimizzazioni che introduce, non si raggiunge una concretezza pratica che renda in grado di utilizzare tale algoritmo in situazioni reali.

Organizzazione della tesi. Nel Capitolo 2 saranno introdotti alcuni concetti base utili a comprendere il lavoro proposto in questa tesi. In particolare si descrivono alcuni concetti di teoria dei grafi, passando poi alla definizione del modello di Ising, focus di tutto il lavoro; si prosegue con cenni sulle probabilità e sulle catene di Markov, terminando con la definizione di algoritmo di approssimazione. Nel Capitolo 3 si descrive la *Logit dynamics*, focalizzando l'attenzione su alcune proprietà e sulle motivazioni che hanno portato all'adozione di tale dinamica per modellare i sistemi complessi. Nel Capitolo 4, si descrive l'algoritmo proposto in [4], che ha gettato le basi per questo lavoro di tesi. Successivamente, nel Capitolo 5 si descrive il contributo fornito al calcolo della ***partition function*** mentre nel Capitolo 6 si propone un nuovo metodo per calcolare il *mean magnetic moment* del sistema, analizzando i miglioramenti e le ottimizzazioni proposte. Tale analisi troverà poi giustificazione nel Capitolo 7, in cui verranno mostrati principalmente i risultati ottenuti effettuando dei test sugli algoritmi descritti nei capitoli precedenti, mostrandone i risultati e confrontando questi ultimi con quelli ottenuti in [6]. Nel capitolo 7, inoltre, è mostrato un esempio di applicazione dell'algoritmo proposto per il calcolo del numero atteso di nodi di un grafo che utilizzano un dato prodotto, simulando una campagna pubblicitaria su tale network. Si conclude infine, nel Capitolo 8, con alcune considerazioni e proposte di sviluppi futuri.

Capitolo 2

Alcuni concetti base

2.1 Teoria dei Grafi

La teoria dei grafi è una branca della matematica, nata nel 1700 con Eulero, che consente di descrivere le relazioni che intercorrono tra un insieme di oggetti. Il grafo è lo strumento attraverso il quale tali relazioni possono essere espresse ed organizzate. Infatti, il grafo, consiste di oggetti chiamati *nodi* e relazioni tra coppie di questi oggetti detti *archi*; nodi connessi tra loro da un arco sono detti *vicini* o *adiacenti*.

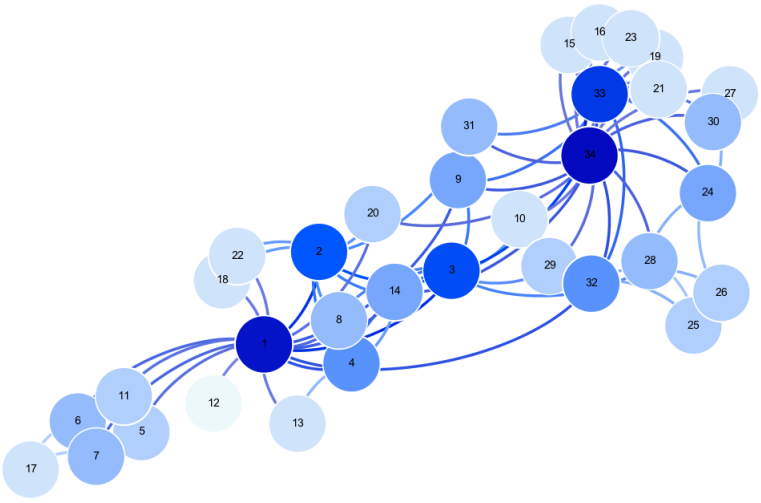


Figura 2.1: Grafo dello Zachary Karate Club.

La relazione tra una coppia di nodi può essere di due tipi:

- Simmetrica: l'arco connette i nodi con un collegamento bidirezionale ed è detto *indiretto*. Un grafo costituito di soli archi indiretti è anch'esso detto indiretto.
- Asimmetrica: l'arco connette i nodi con un collegamento unidirezionale ed è detto *diretto*. Un grafo costituito di soli archi diretti è anch'esso detto diretto.

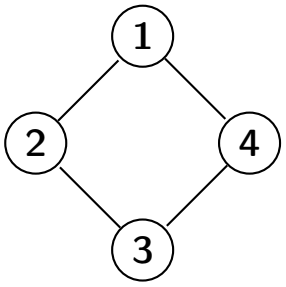


Figura 2.2: Grafo indiretto

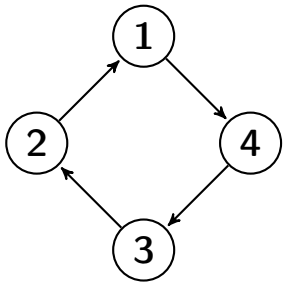


Figura 2.3: Grafo diretto

Un grafo può essere formalmente descritto come una coppia di insiemi **G = (V, E)**, dove V è l'insieme dei nodi ed E è l'insieme degli archi. Un arco e ∈ E è rappresentato come un sottoinsieme di due elementi di V, $e = \{u, v\}$ per $u, v \in V$.
Le rappresentazioni atte a descrivere un grafo sono molteplici:

- *Rappresentazione grafica*: ad ogni nodo corrisponde una figura circolare sul piano e ad ogni arco (i, j) corrisponde una linea che che collega il nodo i al nodo j.
- *Matrice di adiacenza*: matrice di dimensione $n \times n$, dove n è il numero di nodi, il cui elemento (i, j) assume valore 1 se esiste l'arco tra il nodo i ed il nodo j, 0 altrimenti.
- *Lista di adiacenza*: ad ogni vertice v è associata la lista dei nodi ad esso vicini.

Negli anni, gli studi sulla teoria dei grafi hanno prodotto una quantità enorme di definizioni e teoremi, per cui, di seguito vengono descritti solamente i concetti necessari alla comprensione di questo lavoro di tesi.

Sottografo. Un grafo H si dice sottografo di un grafo G se i vertici di H sono un sottoinsieme dei vertici di G e gli archi di H sono un sottoinsieme degli archi di G. Siano $G = (V, E)$ ed $H = (V_1, E_1)$ due grafi. H è un sottografo di G se e solo se $V_1 \subseteq V$ ed $E_1 \subseteq E$. Un concetto particolarmente utile alla comprensione di questo lavoro è lo *spanning subgraph*: uno spanning subgraph H di un grafo G è un sottografo che contiene tutti i vertici di G, cioé $V_1 = V$.

Grado di un nodo. Il grado di un nodo v è il numero di nodi ad esso adiacenti ed è indicato con *deg(v)*.
In un grafo diretto, si distinguono due tipi di grado:

- *in-deg(v)*, il grado in ingresso del nodo v, dato dal numero di archi in cui v compare come nodo destinazione;
- *out-deg(v)*, il grado in uscita del nodo v, dato dal numero di archi in cui v compare come nodo sorgente.

Cammino. Un cammino è una sequenza di nodi, in cui ogni coppia consecutiva della sequenza sia connessa da un arco. Formalmente, un cammino è una sequenza di vertici $v_0, v_1, \cdots, v_n \in V$ tale che $\{v_{i-1}, v_i\} \in E, \forall 1 \leq i \leq n$. Un cammino con almeno tre vertici distinti, i cui vertici di inizio e fine coincidono, è detto *ciclo*.

Grafo connesso. Un grafo è connesso se, per ogni coppia distinta di vertici (i, j), esiste un cammino da i a j.

2.1.1 Reti Sociali

Una rete sociale è un costrutto teorico proveniente dalle scienze sociali, utilizzata per studiare le relazioni fra individui, gruppi, organizzazioni ed intere società. Il termine rete sociale viene utilizzato per descrivere una struttura sociale determinata dalle interazione tra gli attori che la compongono [11]. Tali reti sono quindi strutture relazionali tra attori ed in quanto tali costituiscono una forma sociale rilevante che definisce il contesto in cui si muovono quegli stessi attori. Una rete sociale risulta essere allora la struttura di relazioni, le cui caratteristiche potessero essere usate per spiegare, in tutto o in parte, il comportamento delle persone che la costituiscono. Elementi costitutivi della rete sociale sono dunque:

- I soggetti, che rappresentano le unità, i nodi che compongono la rete (individui, gruppi, posizioni, luoghi, etc.);
- Le relazioni che legano i soggetti, che compongono la rete e che possiedono determinate caratteristiche.

Con riferimento al contenuto della relazione è possibile cogliere ed individuare alcune particolari reti che, per il tipo di legami che le costituiscono, si caratterizzano per essere reti di sostegno (supporto sociale):

- *Formali*, costituite dalle istituzioni sociali;
- *Informali*, che non presentano una veste istituzionalmente definita;
- *Primarie*, costituite da quelle relazioni, che in virtù dei legami naturali, accomunano gli individui, come rapporti familiari, parentali, di vicinato;
- *Secondarie*, formate da relazioni di conoscenza indiretta;
- *Complesse*, composte da un elevato numero di relazioni con caratteristiche specifiche.

Nell'ambito delle scienze sociali, il concetto di rete sociale è stato utilizzato a lungo come "metafora" per tradurre quelle che sono le idee di società e d'azione sociale, come esito di vincoli ed opportunità emergenti dalle relazioni tra i soggetti. L'uso metaforico del termine ha posto il concetto di rete sociale ad un livello di astrazione piuttosto elevato, creando confusione terminologica e mancanza di chiarezza. Successivamente, mediante l'impiego scientifico del termine, tale livello di astrazione è diminuito, determinando il passaggio del concetto di rete dall'immagine intuitiva di un fenomeno complesso alla sua rappresentazione sul piano formale ed analitico. Ciò ha portato le reti sociali, come rappresentazione organizzativa dei rapporti sociali ed il suo metodo di studio, **l'analisi delle reti sociali**, ad essere adottate come strumenti teorici e metodologici per lo studio di numerosi fenomeni e processi. In ambito sociologico, tali studi hanno mostrato che nelle reti si depositano valori materiali, ma soprattutto non materiali che contribuiscono a determinare la "ricchezza" individuale e collettiva (diversa da individuo ad individuo, non solo grazie alle capacità relazionali, ma anche per effetto di specifici processi strutturali) [12]. Quindi, *l'analisi delle reti sociali* comprende principalmente tutta la parte teorica ed i modelli utilizzati per lo studio delle reti sociali in generale. Un esempio reale di una fitta rete sociale è quello mostrato in Figura 2.4. Di seguito, dopo alcuni cenni storici si passa alla descrizione

di concetti e strumenti derivanti dalla *teoria dei grafi* che è alla base della *Social Network Analysis*.

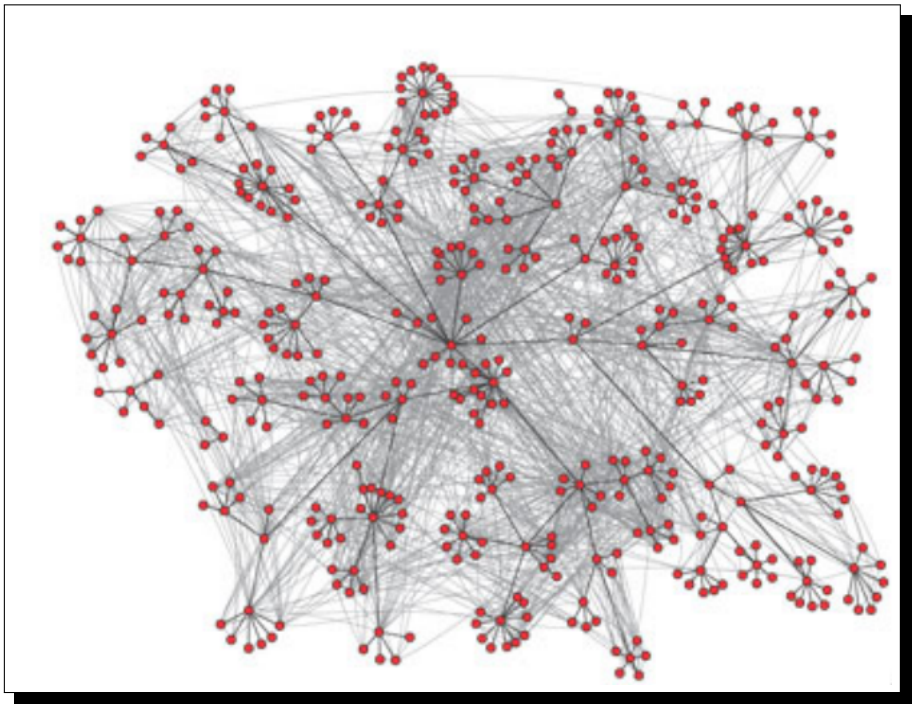


Figura 2.4: Scambio di e-mail tra i 436 impiegati di HP Research Lab

2.1.2 Cenni storici

L'idea delle reti sociali nacque verso la fine del 1900, grazie alle teorie e le ricerche su gruppi sociali di *Èmile Durkheim* e *Ferdinand Tönnies*. Quest'ultimo sosteneva che i gruppi sociali possono esistere come legami personali e diretti, i quali collegano gli individui che condividono valori e credenze (*community*), oppure impersonali e formali (*società*) [13]. Durkheim fornì una spiegazione non individualistica dei fenomeni sociali, affermando che essi sorgono quando gli individui interagiscono costituendo una realtà che non può più essere rappresentata, in termini di proprietà dei singoli [14].
Inoltre intorno al ventesimo secolo, *Georg Simmel* si concentrò sulla natura delle reti e su quanto le dimensioni delle reti influenzassero le interazioni tra le componenti della rete, esaminandone le probabilità di interazione in reti debolmente collegate, piuttosto che in gruppi [15].

Importanti sviluppi nel campo possono essere riscontrati intorno al 1930, grazie a differenti ed indipendenti gruppi di psicologi, antropologi e matematici. In psicologia, *Jacob L. Moreno* iniziò una registrazione ed un'analisi sistematica delle interazioni sociali all'interno di piccoli gruppi. Per quanto riguarda l'antropologia, la fondazione della teoria delle reti sociali è dovuta ai lavori teorici ed entrografici di *Bronislaw Malinowski* [16], *Alfred Radcliffe-Brown* [17, 18], e *Claude Lévi-Strauss* [19], gruppo di antropologi sociali a cui vengono attribuiti i primi lavori di analisi di rete, investigando le community nel sud Africa, India e Regno Unito. In concomitanza a tale gruppo, l'antropologo inglese *S.F. Nadel* codificò una teoria riguardante le strutture sociali che influenzò l'analisi delle reti [20]. Nel campo della sociologia, nei primi anni del 1930 *Talcott Parsons* preparò il terreno per l'adozione di un approccio relazionale per comprendere le struttura sociali [21, 22]. Successivamente, attingendo dalle teorie di Parsons, il lavoro del sociologo *Peter Blau* sulla teoria dello scambio sociale fornì un forte impatto per l'analisi dei legami relazionali [23–25]. A partire dal 1970, un numero crescente di studiosi lavorò per combinare le differenti tracce e tradizioni, fra cui un gruppo composto dal sociologo *Harrison White* ed i suoi studenti del dipartimento di scienze sociale dell'università di Harvard. Ed in maniera indipendente, *Charles Tilly* si concentrò sulle relazioni fra le dinamiche politiche e sociali, e *Stanley Milgram* sviluppò la tesi sui sei gradi di separazione [26]. In seguito, *Mark Granovetter* [27] e *Barry Wellman* [28] furono fra i primi studenti di White ad elaborare e sostenere l'analisi delle reti sociali [29–31].

2.1.3 Grafo come modello della realtà

I grafi hanno una grande utilità, in quanto consentono di astrarre le relazioni che intercorrono tra più oggetti, e di rappresentare tali relazioni in strutture su cui è possibile applicare modelli matematici. In [1] viene proposto un esempio reale: la Figura 2.5 rappresenta la struttura della rete Internet nel Dicembre del 1970, noto come ARPANET allora, composto solo da 13 macchine. I nodi rappresentano gli host, e vi è un arco tra due host se esiste una comunicazione diretta tra di essi. Come è possibile intuire, la posizione geografica dei nodi non ha molta importanza, ma quel che conta è il come ogni nodo sia connesso agli altri. Infatti la figura 2.6 mostra lo stesso grafo di ARPANET, attraverso una rappresentazione logica. Il grafo di ARPANET mostrato in precedenza è un esempio di ***communication network***, i cui nodi sono computer o altri dispositivi capaci di inviare messaggi mentre gli archi rappresentano i collegamenti diretti lungo i quali tali messaggi possono viaggiare. Ma questo è solamente uno dei tipi di rete che possiamo avere.

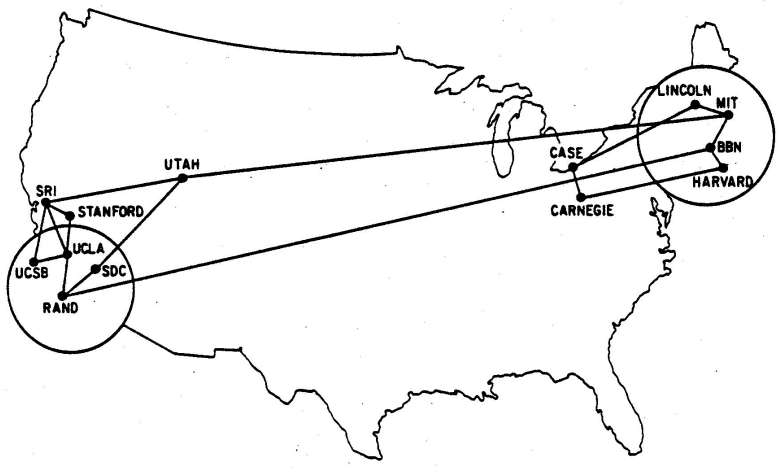


Figura 2.5: ARPANET nel Dicembre 1970

Le ***social network*** i cui nodi sono persone o gruppi di persone, ed i cui archi rappresentano un tipo di interazione (amicizia, inimicizia, ecc.), sono reti massive che al giorno d'oggi comprendono gran parte della popolazione mondiale, come ad esempio le reti di Facebook e Twitter.
Le ***information network*** sono reti che rappresentano il mondo dell'informazione, i cui nodi sono le fonti di informazione e gli archi rappresentano collegamenti logici come riferimenti, citazioni identificati da hyperlink. A tale categoria appartiene il grafo del Web, così come la rete di documenti di Wikipedia.

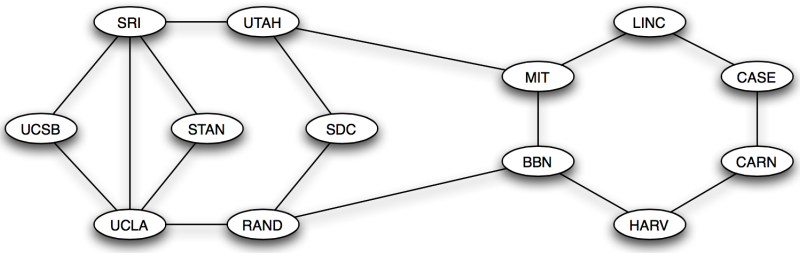


Figura 2.6: Grafo di ARPANET nel Dicembre 1970

Le ***dependency network***, che descrivono le dipendenze esistenti in una collezione di oggetti. Ne sono esempi reti di dipendenze tra task, per cui sono stati sviluppati molti lavori, applicati anche in altri campi come lo studio del sistema immunitario e delle reti semantiche.
I grafi mostrano la loro grande utilità anche nelle ***transportation network***, reti i cui nodi sono luoghi geografici ed i cui archi sono le linee stradali, ferroviarie o aeree che li collegano. Questo tipo di rete è stata di grande importanza nello sviluppo di concetti ed algoritmi su grafi, infatti molta della terminologia deriva dal mondo dei trasporti.
Quindi, come si evince dai vari tipi di network descritte finora, i grafi sono uno strumento potente che trova applicazione nelle realtà più disparate, partendo dalla matematica ed informatica, fino ad arrivare agli studi della sociologia, biologia, linguistica, chimica e fisica. Ed è proprio l'ultimo ambito ad interessare questo lavoro di tesi: la *fisica statistica* è una branca della fisica che utilizza metodi della teoria delle probabilità e statistici, ed in particolare gli strumenti matematici usati per gestire grandi popolazioni ed approssimazioni, per la risoluzione di problemi fisici. Può descrivere una grande varietà di campi dotati di una natura stocastica intrinseca. Le sue applicazioni coinvolgono problemi nel campo della fisica, biologia, chimica, neurologia e scienze sociali.
Lo scopo finale per cui nasce la fisica statistica, in particolare la meccanica statistica, è quello di fornire strutture e strumenti per mettere in relazione proprietà microscopiche di atomi e molecole individuali rispetto a proprietà del materiale che possono essere osservate ad occhio nudo. Spiegando ad esempio la termodinamica come il risultato naturale di statistica, meccanica classica e meccanica quantistica al livello microscopico: il grafo riesce a modellare le particelle del sistema su cui agisce la dinamica del processo fisico.

2.2 Modello di Ising

Il modello di Ising è un classico problema combinatoriale della fisica statistica, studiato per la prima volta da Ernest Ising nel 1920. Tale modello è interessato alla fisica delle transizioni di fase, che occorrono quando un piccolo cambiamento nei parametri, come temperatura o pressione, causa un grande cambiamento qualitativo nello stato del sistema. Le transizioni di fase sono comuni nella fisica, come ad esempio nel fenomeno del ferromagnetismo.
Uno degli scopi principali del modello di Ising è spiegare come interazioni

di breve raggio tra le componenti del sistema siano in grado di dare luogo a comportamenti correlati di lungo raggio e, quindi, di predire in un certo senso il potenziale per una transizione di fase. Tale modello trova applicazione in molti ambiti, in generale in tutti quegli ambiti dove si studia il comportamento cooperativo di grandi sistemi: tali applicazioni sono possibili perché il modello di Ising può essere formulato come un problema matematico.
Una semplice definizione matematica del modello di Ising è data in [32] e in [4]: consideriamo una collezione di siti $[n] = \{0, 1, \cdots, n-1\}$, in cui ogni coppia i, j ha associata un'energia di interazione V_{ij}. Nella maggior parte dei casi di interesse fisico, l'insieme E di coppie con energia di interazione non nulla forma un *regular lattice graph* $([n], E)$: detto anche *mesh graph* o *grid graph*, è un grafo la cui rappresentazione grafica sul piano forma delle tassellature regolari.
Il primo passo del modello è quello di assegnare ad ogni sito i una variabile σ_i, detta **spin**, con $i = 1, \ldots, n$. Le variabili σ_i possono assumere solo due valori, $\sigma_i = \pm 1$, che si possono definire come gli *stati* dei siti.
Una **configurazione** σ del sistema è un assegnamento di spin positivi ($\sigma_i = 1$) o negativi ($\sigma_i = 1$) ad ogni sito $i \in [n]$.
L'*energia* di una configurazione è data dall'Hamiltoniana del sistema: nella fisica matematica, tale quantità governa le dinamiche del sistema. Per il modello di Ising, l'Hamiltoniana è definita sotto un'assunzione ben precisa: si assume che solo le interazioni di breve raggio e con i siti più vicini e le interazioni dei siti con un "campo esterno" contribuiscono al livello di energia del sistema. Pertanto, per ogni configurazione $\sigma = (\sigma_1, \ldots, \sigma_n)$ si ha:

$$H(\sigma) = -\sum_{\{i,j\} \in E} V_{ij} \sigma_i \sigma_j - B \sum_{k \in [n]} \sigma_k \tag{2.1}$$

in cui la prima sommatoria varia su tutte le coppie di vicini nel lattice graph mentre la seconda varia su tutti i siti del sistema. V_{ij} e B sono parametri associati rispettivamente alle interazioni dei vicini ed alle interazioni con il campo esterno.
Nel caso in cui tutte le energie di interazione siano non negative, tale sistema modella il comportamento di un *ferromagnete*: una configurazione "'magnetizzata' (con la maggior parte di coppie di siti vicini aventi $\sigma_i = \sigma_j$), ha un livello di energia più basso rispetto ad una configurazione non magnetizzata. Il parametro B corrisponde alla presenza di un "campo magnetico esterno".

2.2.1 Partition Function

La funzione di partizione è senza alcun dubbio l'oggetto di maggior rilievo introdotto dalla meccanica statistica. La si ottiene dall'esponenziazione dell'Hamiltoniana e sommando su tutte le possibili configurazioni σ, cioè su 2^n possibili assegnamenti di valori ± 1 alle n variabili $(\sigma_1, \ldots, \sigma_n)$:

$$Z = Z(V_{ij}, B, \beta) = \sum_{\sigma} exp(-\beta H(\sigma)). \tag{2.2}$$

Nell'ambito della meccanica statistica, solitamente $\beta = 1/kT$, dove k è la costante di Boltzmann e T è la temperatura ($\beta > 0$).
La partition function è utilizzata come "denominatore" nel calcolo delle probabilità. In particolare, la probabilità che il sistema si trovi in una specifica configurazione σ è data dalla formula

$$P(\sigma) = \frac{e^{-\beta H(\sigma)}}{Z} \tag{2.3}$$

Il segno negativo apporta una maggiore probabilità agli stati con energia più bassa. Valori bassi di β, corrispondenti ad alta temperatura, tendono ad "appiattire" la distribuzione, rendendo così tutte le configurazioni equiprobabili o quasi. Alti valori di β, corrispondenti a bassa temperatura, tendono ad apportare maggiore probabilità agli stati con energia minore.
È possibile ottenere, a partire dalla partition function Z, due quantità altrettanto importanti, la *mean energy* $\epsilon = -\partial(lnZ)/\partial\beta$ ed il *mean magnetic moment* $\mathcal{M} = \beta^{-1}\partial(lnZ)/\partial\beta$.

2.3 Cenni di probabilità e statistica

Di seguito vengono riportati concetti e definizioni utili alla comprensione di questo lavoro di tesi (per maggiori dettagli consultare [33]).

Spazio delle probabilità. Tripla (Ω, F, P), dove Ω è lo *spazio campionario*, cioè l'insieme dei possibili esiti (o *outcome*), F è l'insieme degli eventi, mentre $P : \mathcal{F} \to [0, 1]$ è una funzione che assegna probabilità ad eventi.

Variabile aleatoria. Definita su uno spazio di probabilità, esprime il valore numerico di un outcome di un evento. Una variabile aleatoria X è una

funzione definita sullo spazio degli outcome Ω, ed associa ad ogni elemento di Ω un valore in $\mathcal{R}$. Sia X una variabile casuale che rappresenta il valore degli outcome di un certo evento e si assuma che tale evento possa avere solo un numero finito di possibili outcome. Sia Ω lo spazio campionario dell'esperimento. Una *funzione di distribuzione* per X è una funzione p di valori reali il cui dominio è Ω e che soddisfa $p(\omega) \geq 0$ e $\sum p(\omega) = 1$, con $\omega \in \Omega$. Per ogni sottoinsieme E di Ω, definiamo la *probabilità* di E come

$$P(E) = \sum_{\omega \in E} p(\omega). \tag{2.4}$$

La funzione $p(\omega)$ è la **distribuzione di probabilità** della variabile aleatoria X.

Valore atteso. Sia X una variabile casuale discreta con spazio campionario Ω e funzione di distribuzione $p(x)$. Il *valore atteso E(X)* è definito come

$$E(X) = \sum_{x \in \Omega} xp(x) \tag{2.5}$$

a patto che tale sommatoria converga (altrimenti X non avrebbe valore atteso). Di solito, il valore atteso è anche chiamato *media*, ed è indicato con il simbolo μ. Alcune delle principali proprietà di cui gode il valore atteso sono riportate di seguito:

- Siano X ed Y due variabili aleatorie con valori attesi finiti, allora $E(X + Y) = E(X) + E(Y)$
- Sia X una variabile aleatoria con valore atteso finito e sia c una costante, allora $E(cX) = cE(X)$
- Siano X e Y due variabili aleatorie indipendenti, allora $E(X \cdot Y) = E(X)\dot{E}(Y)$.

Varianza. Sia X una variabile casuale con valore atteso $\mu = E(X)$. Allora la *varianza* di X, indicata con $V(X)$, o con σ^2, è pari a

$$V(X) = E((X - \mu^2)). \tag{2.6}$$

La varianza può essere anche scritta come

$$V(X) = \sum_{x} (x - \mu^2)p(x). \tag{2.7}$$

Una utile alternativa al calcolo della varianza è data dal seguente teorema: *sia X una variabile casuale con $E(X) = \mu^2$, allora $V(X) = E(X^2) - \mu^2$*. Per maggiori dettagli riguardati la prova di tale teorema, consultare pagina 258 di [33].
Alcune importanti proprietà della varianza:

- Sia X una variabile casuale e c una costante, allora $V(cX) = c^2V(X)$
- Siano X ed Y due variabili casuali indipendenti, allora $V(X + Y) = V(X) + V(Y)$.

Disuguaglianza di Chebyshev. Sia X una variabile casuale con valore atteso finito μ e varianza non negativa finita σ^2. Allora per qualsiasi numeri reale $k > 0$ si ha che

$$Pr(|X - \mu| \geq k) \leq \frac{\sigma^2}{k^2}. \tag{2.8}$$

Tale disuguaglianza garantisce che in qualsiasi distribuzione di probabilità, quasi tutti i valori siano vicini alla media.

2.4 Processi Markoviani

Di seguito sono introdotti i concetti alla base delle catene di Markov (consultare [34] per maggiori dettagli).
Una catena di Markov finita è un processo che si muove tra gli elementi di un insieme finito Ω nel seguente modo: se si suppone di trovarsi in $x \in \Omega$, allora la prossima posizione è scelta in accordo ad una distribuzione di probabilità fissata $P(x, \cdot)$. Più precisamente, una sequenza di variabili casuali $(X_0, X_1, \dots)$ è una **catena di Markov con spazio degli stati Ω e matrice di transizione P** se per tuttu gli $x, y \in \Omega$, per tutti i $t \geq 1$, e per tutti gli eventi $H_{t-1} = \bigcap_{s=0}^{t-1} X_s = x_s$ che soddisfano $Pr(H_{t-1} \cap X_t = x) > 0$, si ha

$$Pr(X_{t+1} = y | H_{t-1} \cap X_t = x) = Pr(X_{t+1} = y | X_t = x) = Pr(x, y). \tag{2.9}$$

L'equazione 2.9, nota come *proprietà di Markov*, sta a significare che la probabilità condizionale di procedere dallo stato x allo stato y è la stessa, non importa quale sia la lunghezza degli stati $x_0, \dots, x_{t-1}$ che precedono lo stato corrente x.

2.4.1 Irriducibilità e periodicità

Una catena di Markov P è detta **irriducibile** se, per qualsiasi coppia di stati $x, y \in \Omega$, esiste un intero t tale che $P^t(x, y) > 0$. L'affermazione sta ad indicare che è possibile andare da uno stato in un qualsiasi altro stato utilizzando solo transizioni di probabilità positive.
Sia $\mathcal{T}(x) := t \geq 1 : P^t(x, x) > 0$ l'insieme di volte in cui è possibile per la catena ritornare nella posizione iniziale x. Il **periodo** di x è definito come il *massimo comune divisore (gcd)* di $\mathcal{T}(x)$.

Lemma. *Se P è irriducibile, allora gcd($\mathcal{T}(x)$) = gcd($\mathcal{T}(y)$) $\forall x, y \in \Omega$.*
Data una catena irriducibile, il periodo della catena è definito come il periodo che è comune a tutti gli stati. La catena sarà detta *aperiodica* se tutti gli stati hanno periodo 1, altrimenti *periodica*.

Proposizione. *Se P è aperiodica ed irriducibile, allora vi è un intero r t.c. $P^r(x, y) > 0 \forall x, y \in \Omega$.*
Una catena di Markov finita (con spazio degli stati Ω finito), *irriducibile* ed *aperiodica* è detta **ergodica**.

2.4.2 Distribuzione stazionaria

Sia π una distribuzione di probabilità. Se tale distribuzione soddisfa

$$\pi = \pi P \tag{2.10}$$

allora π è detta *distribuzione stazionaria* della catena di Markov. tale distribuzione può essere riscritta anche per i singoli elementi della catena:

$$\pi(x) = \sum_{x \in \Omega} \pi(x) P(x, y) \forall x, y \in \Omega. \tag{2.11}$$

Il risultato ben noto, che si ottiene dalla teoria dei processi Markoviani è che se la catena in questione è ergodica, allora la distribuzione stazionaria è *unica*.
Una catena di Markov è detta *reversibile* se

$$\pi(x) P(x, y) = \pi(y) P(y, x) \forall x, y \in \Omega. \tag{2.12}$$

Quindi, data una catena (X_t) che soddisfa 2.12 ed ha una distribuzione stazionaria iniziale, allora la distribuzione di $(X_0, X_1, \ldots, X_n)$ è la stessa di

$(X_n, X_{n-1}, \ldots, X_0)$.
Un risultato di grande importanza è il *teorema della convergenza*: tale teorema prova che una catena di Markov irriducibile ed aperiodica converge alla sua distribuzione stazionaria.
Prima di enunciare il problema, però, è necessaria la definizione di *total variation distance*: siano π e μ due distribuzioni di probabilità sullo stesso spazio campionario Ω, allora la total variation distance è definita come

$$\|\pi - \mu\| = max_{A \subset \Omega} |\pi(A) - \mu(A)| = \frac{1}{2} \sum_{x \in \Omega} |\pi(x) - \mu(x)|. \tag{2.13}$$

Si enuncia ora, il **teorema della convergenza**:

Teorema. *Si supponga che P sia irriducibile ed aperiodica, con distribuzione stazionaria π. Allora esistono due costanti $\alpha \in (0,1) e C > 0 t.c.$*

$$max_{x \in \Omega} \|P^t(x, \cdot) - \pi\| \leq C\alpha^t. \tag{2.14}$$

Per la dimostrazione e maggiori dettagli consultare [34] (pag. 52).

2.4.3 Catena di Markov Monte Carlo

Data una matrice di transizione irriducibile P, vi è un'unica distribuzione stazionaria π che soddisfa $\pi = \pi P$, come affermato nella sezione precedente. Si consideri ora il problema inverso: data una distribuzione di probabilità π su Ω, è possibile trovare una matrice di transizione P per cui π è la distribuzione stazionaria? [34] fornisce un esempio pratico per capire come risolvere tale problema.
Il *campione casuale* da un insieme finito Ω è definito come una selezione uniformemente casuale da Ω; una selezione tale che ogni elemento ha la stessa probabilità $1/|\Omega|$ di essere scelto.
Sia $1, 2, \ldots, q$ un insieme di *colori*. Un *proper q-coloring* di un grafo $G = (V, E)$ è un assegnamento di colori ai vertici di V, con il vincolo che nodi vicini non debbano avere lo stesso colore. Per alcune tipologie di grafo, come ad esempio gli alberi, esistono semplici metodi ricorsivi che generano una colorazione casuale; per altri grafi, invece, risulta più difficile. un approccio consiste nell'utilizzare le catene di Markov per campionare: supponiamo che (X_t) sia una catena con spazio degli stati Ω e con una distribuzione stazionaria uniforme su Ω. Grazie al teorema della convergenza enunciato in 2.14, X_t è approssimativamente uniformemente distribuita quando t è grande.

tale metodo di campionamento a partire da una data distribuzione di probabilità è chiamato ***Catena di Markov Monte Carlo***. Si supponga che π sia una distribuzione di probabilità su Ω. se + possibile costruire una catena di Markov (X_t) con distribuzione stazionaria π allora, per t grande abbastanza si ha che la distribuzione di X_t è vicina a π.

2.5 Algoritmi di approssimazione

Un α***-approximation algorithm*** per un dato problema è un algoritmo con tempo di esecuzione polinomiale che, per tutte le istanze del problema, produce una soluzione il cui valore è entro un fattore α del valore della soluzione ottima.
Dato un α-approximation algorithm, α è la *performance* garantita dell'algoritmo. In letteratura è spesso anche chiamato *rapporto di approssimazione* o *fattore di approssimazione*.
Un ***polynomial time approximation scheme*** è una famiglia di algoritmi A_ϵ, in cui vi è un algoritmo per ogni $\epsilon > 0$ tale che A_ϵ è un $()1 + \epsilon$-approximation algorithm (per problemi di minimizzazione) o un $(1 - \epsilon)$-approximation algorithm (per problemi di massimizzazione).
Oltre a queste due definizioni riportate in [35], è interessante fornire le definizioni tratte dal paper di riferimento di questo lavoro di tesi [4]:
dati numeri reali non negativi $a, \tilde{a}, \epsilon$, si dice che $\tilde{a}$ approssima a in un range $(1 + \epsilon)$ se

$$a(1 + \epsilon)^{-1} \leq \tilde{a} \leq a(1 + \epsilon). \tag{2.15}$$

Sia f una qualsiasi funzione che mappa istanze del problema in numeri reali. Un ***randomised approximation scheme*** per f è un algoritmo probabilistico che, quando presentato con un'istanza x ed un numero reale $\epsilon \in (0, 1]$, restituisce un numero che, con alta probabilità, approssima $f(x)$ in un range $(1 + \epsilon)$. Il risultato di tale algoritmo, però, non deve essere ottenuto solo con *alta affidabilità*, ma anche *efficientemente*. In accordo a ciò, si definisce un approximation scheme ***fully polynomial*** se il suo tempo di esecuzione è polinomiale in ϵ^{-1} e nella taglia dell'istanza x del problema.

Capitolo 3

Logit Dynamics

I sistemi complessi sono spesso influenzati da fattori ambientali che possono influire sul modo con cui un giocatore sceglie la propria strategia; inoltre, i giocatori possono avere capacità computazionali limitate, ciò rappresenta il principale limite soprattutto in sistemi informatici e sociali, e conoscenze limitate circa i fattori esterni che possono influenzare il gioco.
Pertanto, c'è bisogno di definire dinamiche che siano capaci di modellare la *bounded rationality* (conoscenza limitata), in cui i giocatori possano prendere decisioni sbagliate, e che portino auspicabilmente il sistema a raggiungere un *equilibrio* che esista sempre, sia unico e velocemente raggiungibile.
La *Logit dynamics* di cui si discute nel corrente Capitolo, introdotta da Blume [2], modella le dinamiche di questo tipo. Per maggiori dettagli sulle motivazioni che supportano la scelta della *Logit dynamics* fare riferimento alla sezione 3.3.

3.1 Definizione

Si consideri un gioco strategico $\mathcal{G} = ([n], S_1, \ldots, S_n, u_1, \ldots, u_n)$, in cui $[n] = 1, \ldots, n$ è un insieme finito di giocatori, S_i è l'insieme finito di strategie per il giocatore $i, S = S_1 \times \cdots \times S_n$ è l'insieme dei profili di strategie ed $u_i : S \to \mathcal{R}$ è la funzione utilità del giocatore $i \in [n]$.
La *Logit dynamics* per un gioco $\mathcal{G}$ procede come segue, ad ogni step:

1. si sceglie un giocatore $i \in [n]$ a caso;

2. sia ggiorna la strategia del giocatore i in accordo alla *logit update rule* con parametro $\beta \geq 0$ sull'insieme S_i delle sue strategie. Cioè. la

dinamica sceglie una strategia $s \in S_i$ con probabilità

$$\sigma_i(s|x) = e^{\beta u_i(s,x_{-i})}/Z_i(x), \tag{3.1}$$

in cui $x = (x_1, \ldots, x_n) \in S$ è il profilo di strategie corrente, $\beta \geq 0$ e

$$Z_i(x) = \sum_{z \in S_i} e^{\beta u_i(s,x_{-i})} \tag{3.2}$$

è il fattore di normalizzazione.

Il parametro β indica il *livello di razionalità* del sistema; pertanto, quando $\beta = 0$ il giocatore i sceglie la sua strategia in maniera casuale, mentre con $\beta > 0$, la probabilità è influenzata dalle strategie che permettono un *payoff* maggiore, e per $\beta \to \infty$ il giocatore sceglie la sua *best response*. Inoltre, bisogna notare che la probabilità $\sigma_i(s|x)$ non dipende dalla strategia x_i attualmente adottata dal giocatore i. Formalmente, la dinamica può essere definita anche nel seguente modo.

Definizione 3.1.1 *Sia $\mathcal{G} = ([n], \mathcal{S}, \mathcal{U})$ un gioco strategico e sia $\beta \geq 0$. La Logit dynamics per $\mathcal{G}$ è la catena di Markov $\mathcal{M}_\beta = (X_{tt\in N}, S, P)$ dove $S = S_1 \times \cdots \times S_n$ e*

$$P(x,y) = \frac{1}{n} \cdot \begin{cases} \sigma_i(s_i|x), & \text{se } y_{-i} = x_{-i} \text{ed } y_i \neq x_i \\ \sum_{i=1}^{n} \sigma_i(s_i|x), & \text{se } x = y \\ 0, & \text{altrimenti} \end{cases} \tag{3.3}$$

con $\sigma_i(s_i|x)$ definita in 3.1. Come sarà mostrato nella sezione 3.2.1, la catena di Markov descritta in 3.3 è ergodica. Pertanto, indipendentemente dallo stato di partenza, la distribuzione della catena al tempo t convergerà alla *distribuzione stazionaria* π man mano che t tende ad infinito.
Si consideri ora un *potential game*. Per tale gioco, la distribuzione stazionaria è la ben nota *Gibbs measure*.

Teorema 3.1.1 *Se $\mathcal{G} = ([n], \mathcal{S}, \mathcal{U})$ è un potential game con funzione potenziale Φ, allora la catena di Markov descritta in 3.3 è reversibile e la sua distribuzione stazionaria è data dalla Gibbs measure*

$$\pi(x) = \frac{1}{Z} e^{\beta \Phi(x)}, \tag{3.4}$$

con $Z = \sum_{y \in S} e^{\beta \Phi(y)}$ è la costante di normalizzazione.

3.2 Proprietà

Si analizzano ora alcune proprietà della *Logit dynamics* utili alla comprensione di questo lavoro di tesi. Per ulteriori proprietà e dettagli consultare [9].

3.2.1 Ergodicità

È semplice vedere come la catena di Markov descritta in 3.3 sia ergodica. Siano $x = (x_1, \dots, x_n)$ ed $y = (y_1, \dots, y_n)$ due profili e sia $z = (z^0, \dots, z^n)$ un *cammino* di profili dove $z^0 = x, z^n = y$ e $z^i = (y_1, \dots, y_i, x_{i+1}, \dots, x_n)$ per $i = 1, \dots, n-1$. La probabilità che la catena, iniziando in x, si trovi in y dopo n passi è

$$P^n(x, y) = P^n(z^0, z^n) \geq P^{n-1}(z^0, z^{n-1})P(z^{n-1}, z^n), \tag{3.5}$$

e, ricorsivamente

$$P^n(x, y) \geq \prod_{i=1}^{n} P(z^{i-1}, z^i) > 0, \tag{3.6}$$

dove l'ultima disuguaglianza deriva da 3.3.

3.2.2 Logit dynamics e Glauber dynamics

In un *potential game*, la *Logit dynamics* è equivalente alla ben nota *Glauber dynamics* [3].
Sia $S = S_1 \times \cdots S_n$ uno spazio degli stati e sia μ una distribuzione di probabilità su S, allora la *Glauber dynamics* per μ procede come segue:
da un profilo $x \in S$ si sceglie un giocatore $i \in [n]$ in maniera casuale e si modifica la sua strategia in $y \in S_i$ con probabilità μ condizionata dagli altri giocatori che sono in x_{-i}

$$\mu(y|x_{-i}) = \frac{\mu(x_{-i}, y)}{\sum_{z \in S_i} \mu(x_{-i}, z)}. \tag{3.7}$$

È facile vedere come la catena di Markov definita dalla *Glauber dynamics* sia irriducibile, aperiodica e reversibile, con distribuzione stazionaria μ. Quando $\mathcal{G} = ([n], \mathcal{S}, \mathcal{U})$ è un *potential game* con funzione potenziale Φ, la *Logit dynamics* definisce la stessa catena di Markov che la *Glauber dynamics* definisce per la *Gibbs distribution* π in 3.4. In accordo a quanto appena detto, la *Logit dynamics* per un *potential game* e la *Glauber dynamics* per la *Gibbs distribution* sono due modi di guardare la stessa catena di Markov. Grazie

a tale analogia, ci si può riferire alla terminologia utilizzata dai fisici per indicare le quantità coinvolte, in particolare, il parametro β è detto *inverse temperature* ed il fattore di normalizzazione Z della *Gibbs distribution* 3.4 è detto *partition function*.

3.3 Movitazioni

Nel lavoro introduttivo alla *Logit dynamics* [2], Blume sottolinea che essa evidenzia due concetti chiave del comportamento strategico: *lock-in* e *bounded rationality*.
La prima proprietà stabilisce che, una volta che un giocatore effettua una scelta, si lega ad essa per qualche tempo: assumendo che tale regola valga nella realtà, giustifica una regola per la scelta di una strategia che tiene conto solamente delle strategie attualmente giocate dagli altri partecipanti e non le strategie che sono state precedentemente scelte. Questo è esattamente ciò che la *Logit update rule* fa in 3.1.
La seconda proprietà interviene nella dinamica in due modi: nel comportamento *miopico* dei giocatori che considerano solo il guadagno attuale e non una prospettiva di guadagni futuri e l'*informazione limitata* che tali giocatori hanno. È evidente che la *Logit update rule* in 3.1 considera tali aspetti. Inoltre, poiché la dinamica descrive una catena di Markov, l'evoluzione del sistema viene trattata in maniera chiara e semplice, permettendo analisi avanzate attraverso gli strumenti forniti dalla teoria dei processi markoviani (alcuni descritti in 2.4). Non essendoci restrizioni circa la struttura dei giochi o la loro funzione utilità, tali dinamiche possono essere applicate ad ogni sistema di interesse.
Un'altra caratteristica, che rende tali dinamiche appetibili, è la forte presenza di *probabilità* e *casualità*. L'approccio probabilistico della *Logit dynamics* è motivato anche dalla volontà di modellare sistemi complessi, intrinsecamente casuali e, pertanto, descrivibili solo attraverso modelli probabilistici.
La *Logit dynamics*, quindi, soddisfa una serie di proprietà interessanti che motivano il loro impiego come modello utilizzato per studiare l'evoluzione dei giochi.

3.4 Alcuni Esperimenti

In letteratura sono presenti diversi esperimenti atti a paragonare le predizioni della *Logit dynamics* con dati reali.
McKelvey e Palfrey, in [36], si focalizzano su esperimenti che coinvolgono

giochi con due persone in cui vi è un unico equilibrio Nash. I dati reali sono stati raccolti da esperimenti che sono stati eseguiti in più di trent'anni.
Per ogni esperimento, è stata calcolata la stima a *maximum likelihood* del parametro β ed è stato analizzato quanto bene il modello approssimi i dati. I risultati mostrano che la *Logit dynamics* predice deviazioni sistematiche dall'equilibrio Nash. Ciononostante, gli autori fanno notare che, sebbene ci sia consistenza con il parametro β negli esperimenti fatti, vi siano alcuni aspetti dei dati reali che non trovano spiegazione in tale dinamica.
Cameter et al. [37] considerano sette giochi e stimano il parametro β utilizzando il 70% dei soggetti, ed utilizzano tale stima per predire il comportamento del rimanente 30% dei soggetti. Mentre, in un altro test, utilizzano sei giochi per stimare β ed utilizzano tali stime sul settimo gioco. I risultati mostrano che ci sono giochi in cui le predizioni della *Logit dynamics* sono paragonabili ai risultati di dinamiche più forti che, però, sono intrattabili analiticamente e si suppone che non convergeranno mai in qualsiasi gioco.

Capitolo 4

Il lavoro di Jerrum e Sinclair

Questo capitolo focalizza l'attenzione sul lavoro che ha gettato le fondamenta per lo sviluppo di questa area di ricerca e sulle quali si basa questa tesi e tanti altri lavori. L'articolo di cui si parla è "*Polynomial-time approximation algorithms for the Ising model*", scritto da Mark Jerrum ed Alistair Sinclair nel 1993 [4].
In breve, [4] presenta un algoritmo randomizzato che calcola la partition function di un sistema di Ising ferromagnetico qualunque, con grado di accuratezza arbitrario. Il tempo di esecuzione di tale algoritmo è polinomiale nella taglia del sistema (i.e. il numero di siti) e variabile in base ad un parametro che controlla l'accuratezza del risultato. L'algoritmo si basa su una simulazione Monte Carlo di una catena di Markov ergodica opportunamente definita. Si può già anticipare che gli stati della catena non sono, come accade di solito, configurazioni di spin di Ising, ma spanning subgraph del grafo di interazione del sistema. Le performance dell'algoritmo sono garantite da prove rigorose e poggiano sulla proprietà di *rapidly mixing* della catena di Markov: converge alla sua distribuzione d'equilibrio in un numero *polinomiale* di passi.
Si prosegue con l'analisi dettagliata dell'algoritmo proposto da Jerrum e Sinclair, descrivendo prima il modo con cui si trasforma il modello di Ising in un nuovo dominio, in cui le configurazioni sono spanning subgraph del grafo di interazione; prosegue con la costruzione di un *fully polynomial randomised approximation scheme (fpras)* per il calcolo della partition function ed infine analizzando la catena di Markov definita sulle nuove configurazioni.
Nei capitoli successivi, l'algoritmo proposto verrà identificato come "*algoritmo JS*".

4.1 Spins world e Subgraphs world

L'obiettivo è costruire un algoritmo per il seguente problema.
Istanza: una matrice reale simmetrica $(V_{ij} : i, j \in [n])$ delle energie di interazione, un numero reale B che rappresenta il campo esterno ed un numero reale positivo β.
Output: la partition function

$$Z = Z(V_{ij}, B, \beta) = \sum_{\sigma} exp(-\beta H(\sigma)), \tag{4.1}$$

Dove l'Hamiltoniana $H(\sigma)$ è data da

$$H(\sigma) = -\sum_{\{i,j\} \in E} V_{ij}\sigma_i\sigma_j - B\sum_{k \in [n]} \sigma_k \tag{4.2}$$

ed E è l'insieme di coppie non ordinate i, j con $V_{ij} \neq 0$.
Tale algoritmo tratta il caso *ferromagnetico* del modello di Ising, che è caratterizzato da energie di interazione V_{ij} non negative. È importante tenere a mente che piuttosto di calcolare *esattamente* la partition function, si preferisce approssimarla.
Una strategia, rivelatasi molto utile per problemi di questo tipo, prevede la simulazione di una catena di Markov opportuna. Un'applicazione diretta di questa strategia alla partition function di Ising procederebbe come segue: si considerino le configurazioni del sistema di Ising, cioè i 2^n possibili vettori di spin $\sigma \in -1, +1^n$, come gli stati della catena di Markov. Si scelgano poi le probabilità di transizione tra gli stati così da rendere la catena *ergodica* e quindi, nella distribuzione stazionaria, la probabilità di essere nello stato σ è $Z^{-1}exp(-\beta H(\sigma))$. Un modo ragionevole per raggiungere ciò, spesso utilizzato, è permettere che le transizioni occorrano tra configurazioni di sping che differiscano in una sola componente, e scegliere le probabilità di transizione in accordo alla regola di Metropolis [38]. Se la catena di Markov risultante è *rapidly mixing*, cioè se converge rapidamente alla distribuzione stazionaria indipendentemente dalla scelta dello stato iniziale, allora può essere usata efficacemente per campionare le configurazioni σ da una distribuzione che è vicina alla distribuzione stazionaria. Raccogliendo sufficienti campioni di configurazioni, usando differenti valori di B e β, dovrebbe essere possibile stimare la partition function Z con buona accuratezza.
Purtroppo, la catena di Markov così descritta (lo *spin-world process*) non è rapidly mixing. È ben noto che i sistemi ferromagnetici di Ising esibiscono tipicamente una transizione di fase ad un certo valore del parametro β, per

valori al di sopra del punto critico, il sistema si stabilizza in uno stato in cui vi è una preponderanza di spin di uno o l'altro segno. transizioni tra gli stati con maggioranza di +1 e stati con maggioranza di -1 occorrono raramente, semplicemente perché la distribuzione stazionaria assegna un peso totale di basso valore alle configurazioni con spin bilanciati.
Il problema causato dall'assenza di *rapid mixing* nello *spin-world process* può essere aggirato simulando una catena di Markov differente: il *subgraphs-world process*. Sebbene le due catene di Markov siano strutturalmente differenti, ed inoltre, il *subgraphs-world process* non abbia alcun significato fisico, quest'ultimo ha una forte connessione con la *partition function* del modello di Ising e, cosa fondamentale nell'applicazione corrente, è *rapidly mixing*.
Tale processo sarà descritto in dettaglio nella sezione 4.3.

Un sottografo si dice *spanning* se include tutti i vertici del grafo "genitore" (in generale gli *spanning subgraph* non sono connessi). Le configurazioni del *subgraph-world* sono *spanning subgraph* del grafo di interazione $([n], E)$. Per semplificare la notazione, siano

$$\lambda_{ij} = tanh\beta V_{ij} \tag{4.3}$$

$$\mu = tanhB\beta \tag{4.4}$$

Ad ogni configurazione $X \subseteq E$ è assegnato un *peso*, in accordo alla formula

$$w(X) = \mu^{|odd(x)|} \prod_{\{i,j\}\in X} \lambda_{ij}, \tag{4.5}$$

in cui la notazione odd(X) sta ad indicare l'insieme di tutti i vertici che hanno grado dispari nel grafo X.
La partition function per il *subgraphs-world* è

$$Z' = \sum_{X\subseteq E} w(X). \tag{4.6}$$

La formula 4.6 è conosciuta come "*the high temperature expansion*".
Una relazione interessante è quella tra le partition function Z e Z': esse sono correlate in maniera semplice. Si definisca

$$A = (2cosh\beta B)^n \prod_{\{i,j\}\in E} cosh\beta V_{i,j}, \tag{4.7}$$

si noti che A è una funzione che può essere facilmente calcolata, poiché è composta dai parametri che specificano il sistema di Ising.
Il seguente risultato classico [39] lega le due partition function:

Teorema 4.1. $Z = AZ'$.

Tale teorema ha portato gli autori di questo lavoro a considerare un sistema della meccanica statistica le cui configurazioni sono *spanning subgraph* di $([n], E)$. In seguito sarà definita una catena di Markov i cui stati sono queste configurazioni, e la cui distribuzione stazionaria assegna probabilità $\pi(X) = w(X)/Z'$ alla configurazione X. questo processo, analizzato in dettaglio nella sezione 4.3, sarà dimostrato essere *rapidly mixing* ed, inoltre, fornirà un mezzo efficiente per campionare le configurazioni con probabilità approssimativamente proporzionali ai loro pesi.
Poiché Z' è una somma pesata delle configurazioni, ci si aspetta che tale procedura dia informazioni utili su Z' stessa e, di conseguenza, sulla partition function originale Z.
Per la dimostrazione di tale teorema, riferirsi a [4] (pag. 1091).

4.2 Stima della Partition Function

Si analizza ora, e descrive, un *efficiente* algoritmo di approssimazione per il calcolo della partition function Z di un sistema di Ising ferromagnetico (definizione di "algoritmo di approssimazione efficiente" al paragrafo 2.5).
Gli autori dell'articolo assumono di trattare con un modello computazionale in cui l'aritmetica è effettuata con un'accuratezza perfetta ed in cui le operazioni aritmetiche e le funzioni standard (i.e. l'esponenziazione) abbiano costo unitario. Inoltre, la taglia dell'istanza del problema è data da n, cioè il numero di siti.
Dato un sistema di Ising ferromagnetico $\langle \lambda_{ij}, \mu \rangle$ on λ_{ij} e μ definiti nella sezione precedente in 4.3 e 4.4, sia Ω l'insieme di *subgraphs-world configuration* e sia π la distribuzione di probabilità su Ω: $\pi(X) = w(X)/\sum_{X'} w(X') = w(X)/Z'$, dove w è la funzione peso definita in 4.5. È importante notare che, poiché il sistema è ferromagnetico, $w(X) \geq 0$ per ogni $X \in \Omega$, pertanto π è una distribuzione di probabilità.

Generatore. Un *generatore* per le *subgraphs-world configuration* è un algoritmo probabilistico che prende in input un sistema di Ising ferromagnetico nella forma $\langle \lambda_{ij}, \mu \rangle$, più una tolleranza reale positiva δ, e restituisce un elemento di Ω preso da una distribuzione p che soddisfa la seguente *variation distance*

$$\|p - \pi\| \leq \delta. \tag{4.8}$$

Risulta possibile costruire un generatore efficiente per le *subgraphs-world configuration*, come enunciato nel seguente teorema.

Teorema 4.2. *Esiste un generatore per le subgraphs-world configuration che, su input* $\langle \lambda_{ij}, \mu \rangle$ *e* δ*, ha un tempo d'esecuzione limitato da un polinomio in* n*,* μ^{-1} *e* $log\delta^{-1}$*. Nello specifico, il tempo di esecuzione del generatore è* $O(m^2\mu^{-8}(log\delta^{-1}+m))$*, in cui* $m = |E|$ *è il numero di interazioni non nulle.*

È importante notare che la presenza di μ^{-1} nel bound sul tempo implica che il generatore sia inefficiente per sistemi con un campo esterno molto basso. Come detto in precedenza, la costruzione di un generatore con tali proprietà basato sulla simulazione di una catena di Markov opportunamente definita, è descritta e giustificata nella sezione 4.3. Per il momento gli autori dell'articolo assumono il Teorema 4.2 e mostrano come i campioni prodotti dal generatore possano essere usati per ottenere un efficiente algoritmo di approssimazione per la partition function Z.
Jerrum e Sinclair partono da una considerazione: suppongono di voler stimare il valore di una quantità fisica associata ad un sistema di Ising ferromagnetico. Il primo passo è esprimere tale quantità come *valore atteso* di una *variabile casuale* opportunamente definita sulle configurazioni del *subgraphs world*. Dopodiché assumono di poter stimare tale quantità campionando delle configurazioni in maniera casuale con l'aiuto del generatore del Teorema 4.2, e calcolandone infine la *media campionaria*. Precisamente, sia f una funzione non negativa a valori reali definita sull'insieme Ω di configurazioni del *subgraphs world* di un sistema di Ising ferromagnetico. Considerando Ω come uno spazio campionario con distribuzione di probabilità $\pi(X) = w(X)/Z'$, la funzione f diventa una variabile casuale, con valore atteso

$$E(f) = \frac{1}{Z'} \sum_{X \in \Omega} w(X) f(X). \tag{4.9}$$

Partendo dal generatore del Teorema 4.2 è semplice ottenere una stima di $E(f)$: si costruisce un campione indipendente X_i di configurazioni di taglia s, e si calcola la media campionaria $s^{-1} \sum_i f(X_i)$. Facendo sì che la taglia s sia abbastanza grande, si può raggiungere un qualsiasi grado di accuratezza desiderato con una ragionevole affidabilità. Inoltre, si può ridurre drasticamente la probabilità che la stima cada al di fuori del range accettabile di accuratezza ripetendo l'intero processo t volte e calcolando la *mediana* dei t risultati. L'efficienza di tale esperimento, quindi, dipende da quanto sono grandi i valori di s e t; ciò, a sua volta, dipende dalla *varianza* della variabile casuale f o, più precisamente, dal rapporto $max(f)/E(f)$, dove $max(f)$ denota il valore massimo che f assume su Ω. Il seguente Lemma quantifica i risultati ottenuti:

Lemma 4.1. *Sia f una variabile casuale a valori reali e non negativi definita sull'insieme Ω di subgraphs-world configuration di un sistema di Ising ferromagnetico, e siano ξ, η numeri reali tali che $0 < \xi \leq 1$ e $0 < \eta \leq 1/2$. Allora vi è un esperimento nella forma descritta in precedenza che utilizza un totale di $504\xi^{-2}\lceil lg\eta^{-1}\rceil max(f)/E(f)$ campioni dal generatore, ognuno con input $\langle \lambda_{ij}, \mu \rangle$ e tolleranza $\delta = \xi E(f)/8max(f)$, e produce un output Y che soddisfa: $Pr(Y$ approssima $E(f)$ in un range $1+\xi) \geq 1-\eta$.*

Data l'importanza ed il ruolo cruciale che tale Lemma ha avuto nello sviluppo di questo lavoro di tesi, la sua dimostrazione è descritta per esteso e nel dettaglio nel paragrafo 4.2.1.
Il Lemma 4.1 afferma chiaramente che, qualora si impiegasse tale tecnica, è necessario assicurarsi che il rapporto $max(f)/E(f)$ non sia troppo grande per la variabile casuale f in considerazione. In particolare per questo caso, il criterio di efficienza adottato prevede che il rapporto sia limitato da una funzione polinomiale in n, la taglia del sistema.
Si analizzi ora com'è possibile applicare tale tecnica per calcolare la partition function $Z(V_{ij}, V, \beta)$ richiamando anche il Teorema 4.1 della sezione precedente (ricordando che la funzione A può essere calcolata direttamente). In accordo a tale teorema, ci si concentra principalmente sul calcolo di Z'; il primo passo è quello di scrivere Z' esplicitamente in funzione di μ come segue:

$$Z' \equiv Z' = \sum_{X \subseteq E} \mu^{|odd(X)|} \prod_{\{i,j\} \in X} \lambda_{ij} = \sum_{k=0}^{\lfloor n/2 \rfloor} c_k \mu^{2k}. \tag{4.10}$$

È importante notare che solo le potenze pari di μ devono essere incluse nella somma, poiché il numero di vertici di grado dispari nel sottografo X è necessariamente pari (si veda il Teorema di Eulero). Pertanto, si considera Z' come un polinomio in μ^2 con coefficienti

$$c_k = \sum_{X : |odd(X)| = 2k} \prod_{\{i,j\} \in X} \lambda_{ij}. \tag{4.11}$$

Nel caso ferromagnetico, tutti i coefficienti c_k sono positivi, pertanto $Z'(\mu)$ è una funzione crescente in μ.
Chiaramente, i coefficienti c_k dipendono da λ_{ij} e di conseguenza dalle energie di interazione del sistema V_{ij} e dal parametro β. Gli autori dell'algoritmo però considerano tali quantità come valori fissati, ed analizzano cosa accade quando si fa variare μ. Nella terminologia dello *spin world*, ciò corrisponde a sottomettere il sistema con interazioni fisse e con temperatura fissa ad

un campo esterno variabile. Lo scopo di tale algoritmo, però, è quello di valutare la partition function dato uno specifico valore del campo esterno B; pertanto, ciò si riduce a valutare il polinomio $Z'(\mu) = \sum c_k \mu^{2k}$ nel punto $\mu = tanh\beta B$.
Il punto di partenza è l'osservazione che il valore di $Z'(\mu)$ quando $\mu = 1$ può essere calcolato direttamente; per capire meglio ciò, è importante notare da 4.10 e 4.11 che

$$Z'(1) = \sum_{k=0}^{\lfloor n/2 \rfloor} c_k = \sum_{X \subseteq E} \prod_{\{i.j\} \in X} \lambda_{ij} = \prod_{\{i,j\} \in E} (1 + \lambda_{ij}). \tag{4.12}$$

Vediamo ora com'è possibile collegare il valore desiderato $Z'(tanh\beta B)$ al valore $Z'(\mu)$ calcolati in alcuni punti intermedi: $tanh\beta B < \mu < 1$.
Il meccanismo per legare i valori di Z' in due punti $\mu = \mu_0$ e $\mu = \mu_1$, con $1 \geq \mu_0 > \mu_1 \geq 0$, è il seguente: si consideri la variabile casuale $f(X) = (\mu_1/\mu_0)^{|odd(X)|}$ sulle configurazioni del sistema quando $\mu = \mu_0$. Il valore atteso di f è dato da

$$E_{\mu_0}(f) = \frac{1}{Z'(\mu_0)} \sum_{k=0}^{\lfloor n/2 \rfloor} c_k \mu_0^{2k} \left(\frac{\mu_1}{\mu_0}\right)^{2k} = \frac{Z'(\mu_1)}{Z'(\mu_0)}.- \tag{4.13}$$

La notazione $E_{\mu_0}(f)$ indica che il valore atteso di f è calcolato rispetto ad un particolare valore di μ, in tal caso μ_0. Come discusso precedentemente in questa sezione, la quantità $Z'(\mu_1)/Z'(\mu_0)$ può essere stimata utilizzando la tecnica di campionamento descritta. Per il Lemma 4.1, tale processo sarà efficiente a patto che il rapporto $max(f)/E_{\mu_0}(f)$ non assuma un valore troppo elevato; chiaramente, questo non può essere garantito per valori di μ_0 e μ_1 arbitrari, ma se tali valori sono ragionevolmente vicini tra loro, allora il rapporto è limitato da un valore piuttosto stretto.
Gli autori dell'articolo provano questa affermazione nel seguente modo: innanzitutto, sicuramente $max(f) \leq 1$; è sufficiente, quindi, ottenere un *lower bound* sul valore atteso $E_{\mu_0}(f)$. tale bound è fornito dal seguente Lemma:

Lemma 4.2. *Siano μ_0 e μ_1 due numeri reali arbitrari nel range $[0, 1]$ che soddisfano $\mu_1 < \mu_0 \leq \mu_1 + n^{-1}$. Allora il rapporto $Z'(\mu_1)/Z'(\mu_0)$ è limitato inferiormente da 1/10.*

Una parte della dimostrazione di tale Lemma sarà analizzata nel capitolo successivo, per la prova completa si veda [4] (pag.1098). Il Lemma 4.2 suggerisce che è possibile portare il valore noto $Z'(1)$ al valore desiderato

$Z'(tanh\beta B)$ effettuando esperimenti statistici su una sequenza di valori intermedi di μ che sono a distanza n^{-1} l'uno dall'altro. Nello specifico, sia $r < n$ un numero naturale che soddisfa

$$\frac{n-r}{n} > tanh\beta B \geq \frac{n-r-1}{n}, \tag{4.14}$$

e sia la sequenza (μ_k) per $0 \leq k \leq r+1$ definita da

$$\mu_k = \begin{cases} (n-k)/n, & \text{per } 0 \leq k \leq r \\ tanh\beta B, & \text{per } k = r+1 \end{cases} \tag{4.15}$$

È importante notare che $\mu_k \in [0,1]$ e che $\mu_{k+1} < \mu_k \geq \mu_{k+1} + n^{-1}$. In accordo alla discussione precedente, si può stimare il rapporto $Z'(\mu_{k+1})/Z'(\mu_k)$ in maniera efficiente per ogni k. Ciò è sufficiente per produrre una stima di $Z'(tanh\beta B)$, poiché si ha

$$Z'(tanh\beta B) = Z'(1) \times \prod_{k=0}^{r} \frac{Z'(\mu_{k+1})}{Z'(\mu_k)}. \tag{4.16}$$

A questo punto, gli autori del lavoro formulano il loro algoritmo di approssimazione per la *partition function Z*.
Si assume che l'input dell'algoritmo consista di un sistema di Ising ferromagnetico nella forma $\langle V_{ij}, B, \beta \rangle$ e di un valore positivo reale $\epsilon \in [0,1]$ che specifica l'accuratezza desiderata; come di consueto, sia $\lambda_{ij} = tanh\beta V_{ij}$.
Step 1. Calcolare

$$A = (2cosh\beta B)^n \prod_{\{i,j\} \in E} cosh\beta V_{ij} \tag{4.17}$$

$$Z'(1) = \prod_{\{i,j\} \in E} (1 + \lambda_{ij}). \tag{4.18}$$

Step 2. Definire la sequenza (μ_k) per $0 \leq k \leq r+1$ in accordo a 4.14 e 4.15. Per ogni $k = 0, 1, \ldots, r$ effettuare quanto segue: sia $f(X) = (\mu_{k+1}/\mu_k)^{|odd(X)|}$ una funzione per ogni subgraphs-world configuration X, tale che $E_{\mu_k}(f) = Z'(\mu_{k+1})/Z'(\mu_k)$. Utilizzando la tecnica del Lemma 4.1 applicata al sistema quando $\mu = \mu_k$, con $\xi = \epsilon/2n$ ed $\eta = 1/4n$, calcolare una quantità Y_k che soddisfi: $Pr(Y_k$ approssima $Z'(\mu_{k+1})/Z'(\mu_k)$ in un range $1 + \epsilon/2n) \geq 1 - 1/4n$.
Step 3. Restituire il prodotto

$$A \times Z'(1) \times \prod_{k=0}^{r} Y_k. \tag{4.19}$$

Teorema 4.3. *L'algoritmo sopra descritto è un fpras per la partition function Z di un sistema di Ising ferromagnetico.*

Prova. L'output dell'algoritmo è il prodotto delle quantità A e $Z'(1)$ calcolate in maniera esatta nello *Step 1*, insieme ad $r+1 \leq n$ variabili casuali Y_k derivanti dagli esperimenti nello *Step 2*. Da 4.16 e dalla proprietà sulle Y_k espressa nello *Step 2*, risulta immediato che il prodotto approssima $Z(V_{ij}, B, \beta)$ in un range $(1+\epsilon/2n)^n \leq 1+\epsilon$ con probabilità almeno $(1-1/4n)^n \geq 3/4$. resta solo da dimostrare che il tempo di esecuzione dell'algoritmo è limitato da un polinomio in n ed ϵ^{-1}.
Step 1 e *Step 3* possono essere chiaramente eseguiti in tempo $O(n^2)$. Ora si consideri l'operazione dello *Step 2* per un particolare valore di k; ricorrendo ai lemma 4.1 e 4.2, si può vedere che il processo per calcolare la stima di Y_k richiede $N = 20160\epsilon^{-2}n^2\lceil lg4n \rceil$ chiamate al generatore del Teorema 4.2. Inoltre, la tolleranza richiesta per ogni chiamata è $\delta = \epsilon/160n$, ed il valore di μ non è mai inferiore ad n^{-1}; dal Teorema 4.2 segue che il tempo di esecuzione per ogni chiamata è limitato da $q(n, \epsilon^{-1})$, per qualche polinomio $q(\cdot,\cdot)$. Il tempo totale di esecuzione dello *Step 2* è quindi $O(nNq(n,\epsilon^{-1}))$, che è una funzione polinomiale in n ed ϵ^{-1}. L'algoritmo, quindi, soddisfa tutti i requisiti di un fpras. □
È importante notare che il Teorema 4.2 fornisce già un *upper bound* sul polinomio q che comprare alla fine della prova precedente. Da questo, è facile vedere che il tempo d'esecuzione totale del *fpras* del Teorema 4.1 è $O(\epsilon^{-2}m^2n^{11}log\ n(log(\epsilon^{-1}n)+m))$.
Si può assumere (w.l.o.g.) che $\epsilon \geq 2^{-m}$, poiché altrimenti si potrebbe calcolare Z in maniera esatta attraverso algoritmi di *forza bruta* in tempo $O(m\epsilon^{-1})$. Pertanto il polinomio del tempo d'esecuzione diventa $O(\epsilon^{-2}m^3n^{11}log\ n)$.

4.2.1 Prova del Lemma 4.2.1

Questa sezione è dedicata alla prova del Lemma 4.1. Tale prova ha giocato un ruolo cruciale nello sviluppo di questo lavoro di tesi, pertanto è stata dimostrata per intero e separatamente.
Sia $Var(f)$ la varianza $f : Var(f) = E(f^2) - E(f)^2$.
Il generatore Teorema 4.2 seleziona gli elementi di Ω da una distribuzione p che è leggermente differente da π. In accordo a ciò, la varianza e la media di f definite nel rispetto di tale distribuzione sono pari a

$$E'(f) = \sum_{X\in\Omega} p(X)f(X); Var'(f) = \sum_{X\in\Omega} p(X)f(X)^2 - E'(f)^2. \qquad (4.20)$$

Poiché la *variation distance* soddisfa $\|p - \pi\| \leq \delta$, si ha

$$|E(f) - E'(f)| \leq \delta max(f) = \xi E(f)/8 \tag{4.21}$$

$$|Var(f) - Var'(f)| \leq 4\delta max(f)^2 = 3\xi E(f) max(f)/8. \tag{4.22}$$

Ora sia X_i un insieme di campioni indipendenti di taglia s prodotto dal generatore, e sia $Y_0 = s^{-1} \sum_i f(X_i) lamediacampionaria$.
Chiaramente Y_0 ha valore atteso $E'(f)$ e varianza $s^{-1}Var'(f)$. Pertanto, applicando la disuguaglianza di Chebyshev 2.3, si ha

$$Pr\left(\left|Y_0 - E^{prime}(f)\right| > \frac{\xi}{3}E'(f)\right) \leq \frac{9}{\xi^2}\frac{Var'(f)}{sE'(f)^2}. \tag{4.23}$$

Ma, se $|Y_0 - E'(f)| \leq \frac{\xi}{3}E'(f)$ allora, da 4.22,

$$\begin{aligned} |Y_0 - E(f)| &\leq |Y_0 - E'(f)| + |E'(f) - E(f)| \\ &\leq \frac{\xi}{3}E'(f) + \frac{\xi}{8}E(f) \\ &\leq \frac{\xi}{3}\left(1 + \frac{\xi}{8}\right)E(f) + \frac{\xi}{8}E(f) \\ &\leq \frac{\xi}{2}E(f). \end{aligned} \tag{4.24}$$

Da notare che questo, a sua volta, implica che Y_0 approssima $E(f)$ in un range $1 + \epsilon$.
inoltre, riapplicando 4.22 si ha

$$\frac{Var'(f)}{E'(f)} \leq \frac{Var(f) + \frac{\xi}{8}E(f)max(f)}{\left(\frac{7}{8}E(f)\right)^2} \leq \frac{\frac{11}{8}E(f)max(f)}{\left(\frac{7}{8}E(f)\right)^2} < \frac{2max(f)}{E(f)}, \tag{4.25}$$

dove nella seconda disuguaglianza è stato utilizzato il bound indipendente dalla distribuzione $Var(f) \leq E(f)max(f)$, valido per qualsiasi variabile casuale non negativa f.
Combinando 4.24 e 4.25 con 4.23 e scegliendo la taglia dei campioni

$$s = 72\xi^{-2}max(f)/E(f), \tag{4.26}$$

si ha

$$Pr(Y_0 \text{ non approssima } E(f) \text{ in un range } 1 + \xi) \leq \frac{18}{\xi^2 s}\frac{max(f)}{E(f)} = \frac{1}{4}. \tag{4.27}$$

Ora si consideri l'esecuzione dell'esperimento appena descritto, un numero t di volte indipendenti e sia Y la mediana dei risultanti t valori di Y_0. In vista di 4.27, la probabilità che Y fallisca nell'approssimare $E(f)$ in un range $+\xi$ è al più

$$\sum_{i=(t+1)/2}^{t} \binom{t}{i} \left(\frac{1}{4}\right)^{i} \left(\frac{3}{4}\right)^{t-i} \leq \left(\frac{1}{4}\right)^{\frac{t}{2}} \left(\frac{3}{4}\right)^{\frac{t}{2}} \sum_{i=(t+1)/2}^{t} \binom{t}{i} \leq \left(\frac{3}{16}\right)^{\frac{t}{2}} 2^t = \left(\frac{3}{4}\right)^{\frac{t}{2}}. \tag{4.28}$$

Scegliendo

$$t = 6\lceil lg\eta^{-1}\rceil + 1, \tag{4.29}$$

tale probabilità è limitata superiormente da $\eta^{3\lceil lg(\frac{4}{3})\rceil} < \eta$. La variabile casuale Y pertanto soddisfa i requisiti del Lemma 4.1.
Il numero totale di campioni richiesti dal generatore è $s \cdot t$, che è limitato superiormente da $504\xi^{-2}\lceil lg\,\eta^{-1}\rceil max(f)/E(f)$. □

4.3 Analisi del subgraphs-world process

In questa sezione si presenta il *subgraphs-world process* ed il relativo algoritmo per simulare la catena di Markov ergodica opportunamente definita. Inoltre saranno enunciati (ed alcuni anche dimostrati) i teoremi principali che hanno portato al raggiungimento di questo risultato.
Sia $\mu > 0$. Prendendo spunto dalla forma che Z' assume in 4.6, si definisce il *subgraphs-world process* $\mathcal{MC}_{Ising}$ come segue.
Lo *spazio degli stati* Ω della catena di Markov $\mathcal{MC}_{Ising}$ è l'insieme di tutti gli *spanning subgraph* $X \subseteq E$; da notare che $|\Omega| = 2^m$, dove $m = |E|$ è il numero di coppie non ordinate i, j con $\lambda_{ij} \neq 0$.
Dati $X, X' \in E$ con $X \neq X'$, la *probabilità di transizione* da X a X' è data da

$$p(X, X') = \begin{cases} 1/2m, & \text{se } |X \oplus X'| = 1 \text{ e } w(X') \geq w(X) \\ w(X')/2mw(X), & \text{se } |X \oplus X'| = 1 \text{ e } w(X') < w(X); \\ 0, & \text{altrimenti.} \end{cases} \tag{4.30}$$

dove $X \oplus X'$ denota la differenza simmetrica di X ed X'. Le *transizioni* in $\mathcal{MC}_{Ising}$ sono perturbazioni in cui un singolo arco può essere aggiunto o cancellato da un sottografo.
Gli autori dell'articolo, a questo punto, definiscono un algoritmo per simulare la catena appena descritta:
supponiamo che lo stato corrente della catena sia $X \in \Omega$; allora, le transizioni da X possono essere scelte in accordo al seguente modello:

1. Con probabilità $1/2$ porre $X' = X$, altrimenti
2. Scegliere un arco $e \in E$ in maniera uniformemente casuale, e sia $Y = X \oplus e$ la differenza simmetrica di X ed e;
3. Se $w(Y) \geq w(X)$ allora porre $X' = Y$; se $w(Y) < w(X)$ allora con probabilità $w(Y)/w(X)$ porre $X' = Y$, altrimenti $X' = X$.

È importante notare che non c'è bisogno di calcolare le funzioni peso $w(X)$ e $w(Y)$ da capo ad ogni iterazione: poiché X ed Y differiscono per un solo arco, il quoziente $w(Y)/w(X)$ può essere calcolato utilizzando solo due prodotti. La catena di Markov $\mathcal{MC}_{Ising}$ e *irriducibile* ed *aperiodica*; pertanto vi è una distribuzione stazionaria ben definita su Ω che è indipendente dallo stato iniziale. Sia $\pi : \Omega \to \mathcal{R}$ definita da $\pi(X) = w(X)/\sum_{X'} w(X') = w(X)/Z'$. gli autori a questo punto dimostrano che π così definita è una distribuzione stazionaria su Ω: per $X, X' \in \Omega$, sia $q(X, X') = \pi(X)p(X, X')$, q è simmetrica nei suoi due argomenti. Se $X = X'$ allora non vi è nulla da provare; se $|X \oplus X'| > 1$, allora $p(X, X') = 0$ e di conseguenza $q(X, X') = 0$. Infine, è chiaro verificare dalla definizione di probabilità di transizione $p(X, X')$ che

$$q(X, X') = (2m)^{-1} min\{\pi(X), \pi(X')\}, \text{ se } |X \oplus X'| = 1. \tag{4.31}$$

Poiché q è simmetrica, allora vale la cosiddetta condizione del *detailed balance*:

$$\pi(X)p(X, X') = q(X, X') = \pi(X')p(X', X). \tag{4.32}$$

Supponiamo, così com'è in questo caso, che la funzione $p(\cdot, cdot)$ descriva le probabilità di transizione di una catena di Markov ergodica: come precedentemente affermato nella sezione 2.4.2, se vi è una qualsiasi funzione $\pi :\to \mathcal{R}$ che soddisfa la condizione 4.32 insieme alla condizione di normalizzazione $\sum_{X \in \Omega} \pi(X) = 1$, allora la catena di Markov è *reversibile* e π è la sua distribuzione stazionaria. Pertanto, la distribuzione stazionaria della catena di Markov $\mathcal{MC}_{Ising}$ è data da $\pi(X) = w(X)/Z'$, come affermato precedentemente, e quindi si può utilizzare tale catena per campionare le configurazioni

$X \in \Omega$ con probabilità approssimativamente proporzionali a $w(X)$.
Come spiegato informalmente prima, se la catena di Markov $\mathcal{MC}_{Ising}$ viene utilizzata come base per un'efficiente procedura di campionamento per le configurazioni allora deve per forza essere *rapidly mixing*: si avvicina all'equilibrio dopo un numero polinomiale di passi. Inoltre, se è possibile per tale catena evolvere, a partire da uno stato iniziale appositamente definito, allora la distribuzione nello stato finale sarà molto vicina alla distribuzione stazionaria, dopo aver effettuato solo un numero polinomiale di passi. Da notare che questo è un requisito non banale: poiché il numero degli stati nella catena è esponenzialmente grande, gli autori dell'articolo chiedono che questo converga dopo aver visitato solo una piccola frazione del suo spazio degli stati.
L'argomento che loro propongono per dimostrare che la catena è *rapidly mixing* si divide in due parti 4.4, in cui stimano la conduttanza di $\mathcal{MC}_{Ising}$. Data una catena di Markov ergodica reversibile, la *conduttanza* [40], [41] è definita da:

$$\Phi = min\{\sum_{X \in S \text{and} X' \notin S} q(X, X') / \sum_{X \in S} \pi(X)\} \tag{4.33}$$

dove la minimizzazione avviene su tutti i sottoinsiemi S degli stati con $0 < \sum_{X \in S} \pi(X) \leq 1/2$ e $0 < \Phi \leq 1$.
Una catena con valore di conduttanza grande ha minor probabilità di "restare bloccata" in una qualsiasi regione piccola dello spazio degli stati, quindi ci si aspetta che converga velocemente. Tale intuizione viene utilizzata nel seguente teorema:

Teorema 4.4. *Sia Φ la conduttanza stazionaria π e $min_X\, p(X, X) \geq 1/2$. Sia $p^{(t)}$ la distribuzione dello stato al tempo t, dato che lo stato iniziale è X_0. Allora la variation distance $\|p^{(t)} - \pi\|$ soddisfa*

$$\|p^{(t)} - \pi\| \leq \frac{(1 - \Phi^2)^t}{\pi(X_0)}. \tag{4.34}$$

Per la prova riferirsi a [4] (pag. 1100).
Il Teorema 4.34 permette di analizzare il rate di convergenza di una catena reversibile esaminando la sua struttura di transizione, come si riflette nella conduttanza. In particolare, se si riesce ad assicurare una *variation distance* al più pari a δ allora è chiaro che sono sufficienti $\Phi^{-2}(ln\delta^{-1} + ln\pi(X_0)^{-1})$ passi. Pertanto, la proprietà di *rapid mixing* generalmente segue da un *lower bound* polinomiale inverso sulla conduttanza. Tale bound è disponibile per la catena $\mathcal{MC}_{Ising}$ definita prima, nello specifico, si ha il seguente teorema:

Teorema 4.5. *La conduttanza della catena di Markov $\mathcal{MC}_{Ising}$ è limitata inferiormente da $\mu^4/4m$.*

La prova completa è disponibile in [4] (pag. 1101).
A questo punto è possibile analizzare la prova del Teorema 4.2, enunciato ed utilizzato nella sezione precedente. *Prova del Teorema 4.2.* Il generatore opera come segue: dato in input un sistema di Ising ferromagnetico nella forma $\langle \lambda_{ij}, \mu \rangle$, con $0 < \mu \leq 1$ ed una tolleranza $\delta \in (0, 1]$, simula la catena di Markov associata $\mathcal{MC}_{Ising}$ per $16m^2\mu^{-8}(ln\delta^{-1} + m)$ passi, iniziando nello stato $X_0 = \emptyset$. Poiché $\lambda_{ij} < 1$ per tutti i, j e $\mu \leq 1$, è chiaro che $w(X_0) \geq w(X)$ per ogni configurazione X; pertanto $\pi(X_0) \geq 2^{-m}$. Ricorrendo al Teorema 4.4, gli autori concludono che il numero specificato di passi è sufficiente per assicurare una *variation distance* pari al più a δ. □

4.3.1 Miglioramenti Precedenti

Nel lavoro [6], Rinaldi raggiunge l'obiettivo di computare la funzione di partizione Z in tempo polinomiale basandosi su quanto mostrato da Jerrum e Sinclair: l'intuizione è quella di far girare un'altra dinamica che goda di due importanti proprietà, ovvero la convergenza "rapida" alla distribuzione stazionaria ed il poter utilizzare tale stazionaria per calcolare quella di nostro interesse. Tale dinamica è quella del *subgraphs-world process* proposta in [4]. Il lavoro di Rinaldi, si mostra corretto, ma non utilizzabile nella vita reale a causa dei tempi d'esecuzione proibitivi, tuttavia pone le basi per ulteriori sviluppi e miglioramenti.
L'analisi della prova del Lemma 4.1, descritta nella sezione 4.2.1, ha portato alla conclusione che i *bound* definiti in tale prova possono essere migliorati, e di conseguenza, andare a migliorare i valori di s e t, poiché dipendenti solo da tali *bound*. Così facendo è possibile ridurre il numero di passi impiegati dall'algoritmo, e quindi il suo *running time*.

Capitolo 5

Partition Function

L'idea sviluppata da Jerrum e Sinclair in [4] apre la strada alla possibilità di calcolare la *partition function*, quindi analizzare la *Logit dynamics*, in contesti reali. Infatti il loro lavoro mostra il primo algoritmo di approsimazione *dimostrabilmente efficiente*, per il calcolo della partition function Z di un sistema di Ising arbitrario. Tale algoritmo, però, non è sufficientemente scalabile ed utilizzabile solo su reti di piccole dimensioni, come dimostrato in [6].
Il significato di *efficienza*, infatti, per gli autori è inteso come complessità *polinomiale* nella dimensione dell'input del problema, cioè gli n siti del sistema.
Tale affermazione è sostenuta dalla dimostrazione del Teorema 4.3, il quale termina con la prova che il tempo totale di esecuzione dell'algoritmo che propongono per calcolare Z è pari a $O(\epsilon^{-2}m^3n^{11}logn)$.
Come evidenziato dal Teorema 4.2, il tempo d'esecuzione del generatore è pari a $O(m^2\mu^{-8}(log\delta^{-1}+m))$. Quindi, considerando l'algoritmo per il calcolo di Z, e tenendo presente che $\mu \geq 1/n$ e $\delta \geq \epsilon/n$, il tempo d'esecuzione del generatore in questo caso è $O(m^2n^8(log(\epsilon^{-1}n)+m))$.
Si analizzi ora la complessità dei singoli passi dell'algoritmo descritti nella sezione 4.2. *Step1* e *Step3* hanno una complessità pari ad $O(n^2)$, mentre quella di *Step2* è limitata da un polinomio in funzione di n ed ϵ^{-1}. L'assunzione $\epsilon \geq 2^{-m}$ che il Teorema 4.3 effettua, si ottiene la seguente relazione: $\epsilon \geq 2^{-m} \geq log\left(\frac{1}{1/2^m}\right) = log2^m = m\,log2$. Congiungendo i *bound* trovati, si arriva alla complessità finale dell'algoritmo enunciata nella prova del Teorema 4.3.
Grazie a questo risultato, è possibile utilizzare l'algoritmo per il calcolo di Z, considerando il *running time* polinomiale nel numero di siti. Nonostante tale

complessità polinomiale, il tempo d'esecuzione resta comunque "alto" anche per sistemi con un piccolo numero di nodi.
L'obiettivo di questo lavoro di tesi, descritto nel corrente capitolo, è analizzare l'algoritmo proposto da Jerrum e Sinclair ed intervenire dove possibile così da ridurre il tempo d'esecuzione totale.

5.1 I parametri ξ ed η

L'algoritmo proposto per il calcolo di Z, presenta, tra i parametri dello *Step 2*, due quantità definite come

$$\xi = \epsilon/2n \text{ ed } \eta = 1/4n. \tag{5.1}$$

Questi valori si ripresentano nella prova del Lemma 4.1, descritta nella sezione 4.2.1, ξ in particolare si ritrova nel calcolo di s in 4.26 ed η nel calcolo di t in 4.29. Considerando che s è la taglia dell'insieme dei campioni di configurazioni da calcolare, utilizzando il generatore in 4.3, e t è il numero di volte in cui gli esperimenti devono essere ripetuti, e ricordando che l'algoritmo per il calcolo di Z, in particolare lo *Step 2*, si avvale di tale processo per calcolare le quantità Y_k, migliorare i valori di ξ ed η comporta un miglioramento nel numero di passi che l'algoritmo deve eseguire.
Si consideri la prova del Teorema 4.3: dagli esperimenti dello *Step 2* risultano $r+1 \leq n$ variabili casuali Y_k. Riprendendo l'equazione 4.16 e l'enunciato del Teorema 4.1, si ha

$$Z = A \times Z'(1) \times \prod_{k=0}^{r} Y_k \tag{5.2}$$

di conseguenza, gli autori dell'articolo affermano (avvalendosi dell'enunciato del Lemma 4.1) che tale prodotto approssima Z in un range $(1+\epsilon/2n)^n \leq 1+\epsilon$ con probabilità almeno $(1-1/4n)^n \geq 3/4$.
Tale bound in realtà può essere migliorato: infatti, come è mostrato in [6], è facile notare che gli autori assumono che il passo *"r-esimo"* dei k passi da effettuare nello *Step 2* sia in realtà il passo *"n-esimo"*, per cui r è limitato superiormente da n ed utilizzano tale limite nel calcolo delle probabilità e dei valori. Quindi, è possibile migliorare ξ ed η definendoli come segue

$$\xi = \epsilon/2r \text{ ed } \eta = 1/4r. \tag{5.3}$$

Poiché $r < n$ e si trova al denominatore, i valori di conseguenza aumentano, andando a migliorare il numero di iterazioni *st* necessarie allo *Step 2* per ottenere Y_k.

5.2 Stima del valore atteso

Nel capitolo precedente, data una variabile casuale f ed una distribuzione di probabilità π su Ω, è stato definito in 4.9. Inoltre, si mostra come sia possibile ottenere una stima di tale valore atteso: è sufficiente costruire un insieme di taglia s di configurazioni utilizzando il generatore del Teorema 4.2, e calcolare la *media campionaria* di f definita su tali valori. Per riuscire ad ottenere un risultato quanto più accurato possibile, si può ripetere tale esperimento un numero t di volte e calcolare la *mediana* dei t risultati. Il Lemma 4.1 fornisce i valori appropriati di s e t per effettuare tali esperimenti. L'algoritmo per il calcolo di Z, ed in particolare lo *Step 2*, si avvale di tale processo per calcolare le quantità Y_k che approssimano i valori attesi $E_{\mu_k(f)}$, con f opportunamente definita.

5.3 Stima della Partition Function

A differenza del lavoro proposto da Jerrum e Sinclair, il teorema 4.4 qui non è utilizzato. Tale teorema, consente di investigare il tasso di convergenza di una catena reversibile, esaminando la sua struttura di transizione, come pensato nella conduttanza. In particolare, se si vuole assicurare una *variation distance* di al più δ allora è chiaro che $\Phi^{-2}(ln\,\delta^{-1} + ln\,\pi(X_0)^{-1})$ passi siano sufficienti.
Usando invece la definizione di t_{rel}, il Teorema 2.3, Teorema 3.1 e Teorema 2.6 del lavoro [5], invece è possibile affermare che t_{mix} è al più $\rho(ln\,\delta^{-1} + ln\,\pi(X_0)^{-1})$.

Metodi Spettrali. Sia P la matrice di transizione di una catena di Markov con stato degli spazi finito Ω e siano etichettati gli autovalori di P in ordine non-crescente

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_{|\Omega|}.$$

È noto che $\lambda_1 = 1$ e, se P è ergodica allora $\lambda_2 < 1$ e $\lambda_{|\Omega|} > -1$. Si denoti con λ^* il maggior valore assoluto tra gli autovalori a parte λ_1. Il *relaxation time* t_{rel} di una catena di Markov ergodica $\mathcal{M}$ è definito come

$$t_{rel} = \frac{1}{1-\lambda^*} = \{\frac{1}{1-\lambda_2}, \frac{1}{1+\lambda_{|\Omega|}}\}.$$

Il tempo di rilassamento è collegato al mixing time dal Teorema 2.3 riportato di seguito.

Teorema 5.1 (2.3)**.** *(Relaxation Time). Sia P la matrice di transizione di una catena di Markov ergodica reversibile, con spazio degli stati Ω e distribuzione stazionaria π. Allora*

$$(t_{rel} - 1) \cdot log\left(\frac{1}{2\epsilon}\right) \leq t_{mix}(\epsilon) \leq t_{rel} \cdot log\left(\frac{1}{\epsilon\pi_{min}}\right), \tag{5.4}$$

dove $\pi_{min} = min_{x\in\Omega}\pi(x)$.

Il Teorema 3.1 afferma che il secondo autovalore della matrice di transizione della catena di Markov della logit dynamics con *inverse noise* β per un gioco di potenziale è sempre maggiore in valore assoluto dell'ultimo autovalore. Quindi, per questi giochi, $t_{rel} = \frac{1}{1-\lambda_2}$.

Teorema 5.2 (3.1)**.** *Sia $\mathcal{G}$ un gioco di potenziale ad n giocatori con spazio dei profili S e sia P la matrice di transizione della catena di Markov della logit dynamics con inverse noise β per $\mathcal{G}$. Siano $1 = \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_{|S|}$ gli autovalori di P. Allora $\lambda_2 \geq |\lambda_|S||$.*

Per maggiori dettagli sulla prova, consultare [5] (pag. 8).
Il Teorema 2.6 è un caso speciale del *Path Comparison Theorem* ottenuto considerando una catena di Markov $\mathcal{M}$ con distribuzione stazionaria π ed una catena di Markov $\hat{\mathcal{M}}$ con probabilità di transizione $\hat{P}(x, y) = \pi(y)$.

Teorema 5.3 (2.6)**.** *(Canonical paths). Sia $\mathcal{M}$ una catena di Markov ergodica reversibile definita sullo spazio degli stati Ω con matrice di transizione P e distribuzione stazionaria π. Per ogni coppia di profili $x, y \in \Omega$, sia $\Gamma_{x,y}$ un $\mathcal{M}$-path. La congestione ρ dell'insieme di path è definito come*

$$\rho = max_{e\in E}\left(\frac{1}{Q(e)}\sum_{x,y:e\in\Gamma_{x,y}}\pi(x)\pi(y)|\Gamma_{x,y}|\right). \tag{5.5}$$

È possibile affermare che $\frac{1}{1-\lambda_2} \leq \rho$.

Inoltre, mentre in [4] viene utilizzato il Teorema 7 per dare un limite superiore alla Φ che compare nella formula, ora è necessario dare un *bound* alla ρ che compare nella nuova formula.
Per calcolare tale *bound* si utilizza un approccio che unisce la prova del Teorema 5.1 di [5] e la prova del Teorema 7 di [4]. Il primo passo effettuato nel Teorema 5.1 è quello di definire un ordine arbitrario di oggetti (in quel caso vertici di un grafo), in questo lavoro viene definito allo stesso modo sugli

archi.
Sia l un ordinamento degli archi $\{1, 2, \cdots, n\}$ di G e siano rinominati così che $1 <_l 2 <_l \cdots <_l n$. In seguito, il Teorema 5.1 definisce per ogni due stati della catena un *canonical path* tra di essi. La stesso viene effettuato sulla catena di Markov che caratterizza questo lavoro. In particolare il canonical path definito è esattamente quello definito da [4] all'inizio del Teorema 7. Per ogni coppia di stati $I, F \in \Omega$, è specificato un *canonical path* da I (lo stato iniziale) ad F (lo stato finale). Il *canonical path* procede attraverso un numero di stati intermedi utilizzando solamente transizioni valide della catena di Markov. Ad ogni *canonical path* è assegnato un peso che è il prodotto delle probabilità stazionarie allo stato iniziale e finale; quindi il peso del path da I ad F è $\pi(I)\pi(F)$, a prescindere dagli stati intermedi del path. Un'altra cosa che avviene nel Teorema 5.1 consiste nel mostrare l'iniettività della funzione f_e (Lemma 5.3). In questo lavoro la funzione da considerare non è f_e ma la funzione $\eta_{T \to T'}$.
Per prima cosa è necessario definire un'applicazione iniettiva dall'insieme dei *canonical path* usando una data transizione che porti ad Ω. Da tenere a mente che $cp(T, T')$ denota l'insieme di tutte le coppie $(I, F) \in \Omega^2$ tali che il *canonical path* da I ad F impieghi la transizione $T \to T'$. Si definisca l'applicazione $\eta_{T \to T'} : cp(T, T') \to \Omega$ con $\eta_{T \to T'}(I, F) = I \oplus F \oplus (T \cup T')$ per ogni $(I, F) \in cp(T, T')$.
È possibile verificare che $\eta_{T \to T'}$ sia iniettiva dimostrando che I ed F sono unicamente determinate da $U = \eta_{T \to T'}(I, F)$. Infatti, dato U, è possibile computare $U \oplus (T \cup T') = I \oplus F$ e quindi il covering unicamente definito $C_1, C_2, \ldots, C_r$ di $I \oplus F$. L'arco $e = T \oplus T'$, aggiunto o eliminato dalla transizione $T \to T'$, mostra quale percorso, C_i, è stato *"rilassato"*, e quanto il rilassamento di C_i ha progredito. A partire dallo stato T', si potrebbe completare il rilassamento di C_i ed i consecutivi percorsi così da scoprire lo stato finale F; allo stesso modo, sarebbe possibile utilizzare il processo inverso per recuperare lo stato iniziale I. Quindi l'applicazione $\eta_{T \to T'}$ è iniettiva, come affermato. □
L'ultimo passo effettuato dal Teorema 5.1 in [5] è provare il Lemma 5.2. Quest'ultimo afferma che

Lemma 5.1 (5.2)**.** *Per ogni coppia di profili* $\boldsymbol{x}$*,* $\boldsymbol{y} \in S$*, per ogni ordinamento* l *dei vertici di* G *e per ogni arco* $(u, v) \in \Gamma^l_{x,y}$*,*

$$\frac{\pi(x) \cdot \pi(y)}{min\{\pi(u), \pi(v)\}} \leq e^{\beta |E^l_i| (\delta_1 + \delta_0)} \pi(z), \tag{5.6}$$

dove $\boldsymbol{z} = f^l_{u,v}(x, y)$ *ed* i *è la componente in cui i profili* $\boldsymbol{u}$ *e* $\boldsymbol{v}$ *differiscono.*

L'equazione 25 del lavoro [4] ha una forma molto simile a quella appena presentata dal Lemma riportato sopra, fatta eccezione per il termine $e^{\beta|E_i^l|(\delta_1+\delta_0)}\pi(z)$ che è sostituito da μ^{-4}, infatti l'equazione 25 si presenta come segue: per ogni istanza di applicazione $U = \eta_{T\to T'}(I,F)$ è richiesto che

$$w(U)w(T) \geq \mu^{-4}w(I)w(F) \text{ e } w(U)w(T') \geq \mu^{-4}w(I)w(F). \tag{5.7}$$

Ora è possibile calcolare il *bound* su ρ così come mostrato nella prova del Lemma 5.4 di [5]. Si afferma quindi che

$$\rho(\Gamma^l) \leq 2m^2\mu^{-4}w(I)w(F). \tag{5.8}$$

Prova. A partire da

$$P(u,v) = \frac{1}{m}\frac{e^{-\beta\phi(v)}}{e^{-\beta\phi(v)}+e^{-\beta\phi(u)}} \geq \frac{e^{-\beta\, min\{\phi(u),\phi(v)\}}}{2m} \tag{5.9}$$

è possibile affermare che

$$Q(u,v) \geq \frac{min\left\{\pi(u),\pi(v)\right\}}{2m}. \tag{5.10}$$

Quindi,

$$\rho(\Gamma^l) = \max_{\mathcal{M}-edge(u,v)}\left(\frac{1}{Q(u,v)}\sum_{x,y:(u,v)\in\Gamma^\star_{x,y}}\pi(x)\cdot\pi(y)\cdot|\Gamma^l_{x,y}|\right)$$

(per l'equazione 18 e $|\Gamma^l_{x,y}| \leq m$)

$$\leq \max_{\mathcal{M}-edge(u,v)}\left(2m^2\sum_{x,y:(u,v)\in\Gamma^l_{x,y}}\frac{\pi(x)\cdot\pi(y)}{min\left\{\pi(u),\pi(v)\right\}}\right)$$

(per il Lemma 5.4 e $w(U)w(T) \geq \mu^{-4}w(I)w(F)$)

$$\leq \max_{\mathcal{M}-edge\mathbf{e}}\left(2m^2\sum_{x,y:\mathbf{e}\in\Gamma^l_{x,y}}\mu^{-4}w(I)w(F)\pi(\eta_{T\to T'}(I,F))\right)$$

$$\leq \max_{\mathcal{M}-edge\mathbf{e}}\left(2m^2\mu^{-4}w(I)w(F)\sum_{x,y:\mathbf{e}\in\Gamma^l_{x,y}}\pi(\eta_{T\to T'}(I,F))\right)$$

(per il Lemma 5.3) $\leq 2m^2\mu^{-4}w(I)w(F)$. $\square$

Capitolo 6

Mean Magnetic Moment

La *partition function* Z è un oggetto fondamentale nella fisica statistica, ma la derivata parziale di $ln\ Z$ rispetto a β e B ha un'importanza nettamente maggiore, per i motivi che si vedranno in seguito. Tale quantità è il *mean magnetic moment* che, come anticipato in 4.1, è definita come

$$\mathcal{M} = \beta^{-1}\partial(lnZ)/\partial\beta. \tag{6.1}$$

Esiste un algoritmo di approssimazione polinomiale per $\mathcal{M}$, il quale si basa sulla visione del *mean magnetic moment* come valore atteso di variabili casuali opportunamente definite nel *subgraphs-world*.
È possibile stimare tale valore atteso in maniera diretta, simulando il *subgraphs-world process* per un numero polinomiale di passi.
Tenendo a mente che nella distribuzione del *subgraph-world* ogni configurazione X occorre con probabilità $w(X)/Z'$.

Lemma 6.1. *Supponiamo che la configurazione $X \in \Omega$ sia scelta in maniera casuale in accordo alla distribuzione del subgraph-world. Allora:*

- $Pr(|odd(X)| > 0) \leq \mu^2/2,\ se\ \sum \lambda_{ij} \geq 1$;
- $Pr(|odd(X)| = 2) \geq \mu^2/10,\ se\ \sum \lambda_{ij} \geq 1\ e\ \mu \leq n^{-1}$.

Segue la dimostrazione, tratta da [4] (pag. 1111).

Prova del Lemma 6.1 . Si dimostra, attraverso una semplice relazione, che

$$Pr(|odd(X)| = 2) \geq \mu^2 Pr(|odd(X)| = 0). \tag{6.2}$$

Sia Ω_k l'insieme $X \in \Omega : |odd(X) = 2k|$. Si associ ad ogni configurazione $X \in \Omega_0$ l'insieme $S(X) = X' \in \Omega : |X' \oplus X| = 1 \subseteq \Omega_1$. È semplice verificare che i sottoinsiemi $S(X) : X \in \Omega_0$ sono a due a due disgiunti, e che $\sum_{X' \in \Omega_1} w(X') \geq w(X)\mu^2$ per ogni $X \in \Omega_0$. (Per $X = \emptyset$ c'è bisogno della condizione $\sum \lambda_{ij} \geq 1$.) Quindi $\sum_{X \in \Omega_1} w(X) \geq \mu^2 \sum_{X \in \Omega_0} w(X)$, e 6.2 segue dalla divisione per Z'.
Segue da 6.2 che $Pr(|odd(X)| > 0) \geq \mu^2/(1+\mu^2) \geq \mu^2/2$, e questo deriva dalla prima parte del lemma. Inoltre, il Lemma 4.2 assicura che $Pr(|odd(X)| = 0) \geq 1/10$ quando $\mu \leq n^{-1}$. Combinando questa osservazione con 6.2 porta alla seconda parte del lemma. □
Viene enunciato ora il teorema principale che consente di realizzare l'algoritmo di approssimazione per $\mathcal{M}$.

Teorema 6.1. *Esiste un fpras per il mean magnetic moment $\mathcal{M} = \beta^{-1}\partial(lnZ)/\partial\beta$, dove Z è la partition function di un sistema di Ising ferromagnetico.*

La prova completa è disponibile in [4] (pag. 1105). Per ora è analizzata solamente la parte della dimostrazione che interessa questo lavoro di tesi.
Come accennato prima, bisogna esprimere la quantità **M** come valore atteso nel *subgraphs-world* differenziando il logaritmo dell'espansione data nel Teorema 5.2 rispetto a B, ma prima di effettuare tale operazione, gli autori forniscono dei calcoli preliminari.
Poiché $\mathcal{M} = 0$ quando $B = 0$, essi assumono che $B > 0$. Ricordando che $w(X) = \Lambda(X)\mu^{|odd(X)|}$, dove $\mu = tanh\,\beta\,B$ per definizione, e $\Lambda(X)$ è indipendente da B, si ha

$$\begin{aligned} \frac{\partial}{\partial B} w(X) &= \Lambda(X)|odd(X)|\mu^{|odd(X)|-1}(sech\,\beta\,B)^2\beta \\ &= \beta\, w(X)\,|odd(X)|\,(tanh\,\beta\,B)^{-1}(sech\,\beta\,B)^2 \\ &= w(X)\frac{2\,\beta\,|odd(X)|}{sinh\,2\beta\,B}. \end{aligned} \tag{6.3}$$

Inoltre, dalla definizione di A fornita in 4.7,

$$\frac{\partial}{\partial\, B} ln\, A = \frac{\partial}{\partial\, B}\, n\, ln\, cosh\,\beta\, B = n\,\beta\, tanh\,\beta\, B. \tag{6.4}$$

Dopo aver definito tali identità, gli autori calcolano $\mathcal{M}$ utilizzando l'espan-

sione del Teorema 4.1 come punto di partenza:

$$\begin{aligned}\mathcal{M} = \frac{1}{\beta}\frac{\partial}{\partial B} ln\, Z &= \frac{1}{\beta}\frac{\partial}{\partial B} ln\, A + \frac{1}{\beta}\frac{\partial}{\partial B} ln\, Z' \\ &= n\, tanh\, \beta\, B + \frac{1}{\beta\, Z'}\sum_X \frac{\partial}{\partial B}\, w(X) \\ &= n\, tanh\, \beta\, B + \frac{1}{Z'}\sum_X w(X)\, \frac{2\,|odd(X)|}{sinh\, 2\,\beta\, B}.\end{aligned} \tag{6.5}$$

Utilizzando la notazione, come fatto in precedenza,

$$E(f) = (Z')^{-1}\sum_X w(X) f(X) \tag{6.6}$$

per esprimere il valore atteso di una variabile casuale f nel *subgraphs-world*, l'identità appena espressa può essere scritta in maniera più compatta come

$$\mathcal{M} = n\, tanh\, \beta\, B + \frac{2}{sinh\, 2\,\beta\, B}\, E|odd(X)|. \tag{6.7}$$

6.1 Approssimazione della funzione odd(X)

Per approssimare $\mathcal{M}$ in un range $1+\epsilon$ è sufficiente, poiché entrambi i termini di 6.7 sono positivi, stimare $E|odd(X)|$ in un range $1+\epsilon$. Gli autori propongono di raggiungere tale obiettivo utilizzando la catena di Markov $\mathcal{MC}_I sing$ analizzata in 4.3, per fornire un numero polinomiale di campioni di configurazioni X dalla distribuzione del *subgraphs-world*, e restituire la media di $|odd(X)|$ sui campioni. Come discusso nella sezione che precede il Lemma 4.1, tale approccio porterà ad un *fpras* per $\mathcal{M}$ a patto che il rapporto $max\,|odd(X)|/E|odd(X)|$ sia limitato da un polinomio in n.
Per dare un tale *bound* a questo rapporto, gli autori procedono per casi, definendone due. Analizziamo solo quello di nostro interesse, cioè il *Caso I*, ed in particolare il sottocaso *Caso I(a)*: *Caso I.* $\sum \lambda_{ij} > 1$. *Caso I(a).* $\mu \geq n^{-1}$. In questo range è possibile stimare $\mathcal{M}$ con un esperimento diretto. Dal Lemma 6.1,

$$E|odd(X)| \geq 2\, Pr(|odd(X)| > 0) \geq \mu^2 \geq n^{-2}, \tag{6.8}$$

dove, ovviamente, $max\,|odd(X)| \geq n$.
Pertanto il rapporto $max\,|odd(X)|/E|odd(X)|$ è limitato inferiormente da n^3.

6.1.1 Migliore approssimazione per odd(X)

La sezione precedente, mostra come l'approssimazione del *mean magnetic moment* $\mathcal{M}$ sia fortemente influenzata dalla stima del valore atteso della funzione $odd(X)$.
In questa sezione si analizza tale approssimazione, in particolare facendo riferimento alla sua realizzazione presente nel lavoro [6]. Infatti, sebbene sia corretto definire $max(f) = n$ e $min(f) = 2$ (in accordo al Teorema di Eulero), non è accurato dal punto di vista pratico sfruttare il *bound* mostrato in 6.8. Approssimare $E|odd(X)|$ con i valori proposti in tale relazione, risulta in un valore atteso non veritiero, in quanto, assumendo che il valore selezionato sia μ^2, e tenendo presente il fatto che μ è un valore prossimo ad 1 nella maggior parte dei casi, ciò starebbe a significare che *il numero atteso di nodi del grafo della configurazione aventi grado dispari* sia prossimo ad 1.
Come è semplice immaginare, tale stima presenta scarsa accuratezza. Quindi si è pensato ad un algoritmo che consenta di determinare il valore atteso di $odd(X)$ in maniera dinamica, basandosi sull'istanza di grafo in input, avvalendosi della teoria dei *subgraphs-world*.
Si passa ora a sviluppare la relazione di partenza 6.8 per giungere così ad un nuovo *bound* per $E|odd(X)|$. Il valore atteso di $odd(X)$ può essere riscritto come la sommatoria di $x \cdot p(x)$, dove la probabilità è quella che la cardinalità dell'insieme composto dai nodi a grado dispari della configurazione X sia pari a $2x$, per x che varia da 1 ad $n/2$.

$$E|odd(X)| = \sum_{i=0}^{n/2} 2i \ \cdot \ Pr(|odd(X)| = 2i), \tag{6.9}$$

tale probabilità equivale a

$$Pr(|odd(X)| \ = \ 2i) = \sum_{X:|odd(X)|=2i} Pr(X), \tag{6.10}$$

dove $Pr(X)$ rappresenta la probabilità di trovarsi in una delle possibili configurazioni, ed è pari a

$$Pr(X) = \frac{W(X)}{Z'} \tag{6.11}$$

ma come mostrato da 4.5, $W(X) = \mu^{|odd(X)|} \cdot \prod_{\{i,j\}\in X} \lambda_{i,j}$, quindi

$$\begin{aligned} Pr(X) &= \frac{\mu^{|odd(X)|} \cdot \prod_{\{i,j\}\in X} \lambda_{i,j}}{Z'} \\ &= \frac{\mu^{|odd(X)|} \cdot \lambda^{|edges(X)|}}{Z'}. \end{aligned} \tag{6.12}$$

A questo punto è possibile sostituire nella formula iniziale per riscrivere il valore atteso come

$$\begin{aligned} E|odd(X)| &= \sum_{i=0}^{n/2} 2i \cdot \sum_{X:|odd(X)|=2i} \frac{\mu^{|odd(X)|} \cdot \lambda^{|edges(X)|}}{Z'} \\ &= \frac{1}{Z'} \sum_{i=0}^{n/2} 2i \cdot \mu^{2i} \cdot \sum_{X:|odd(X)|=2i} \lambda^{|edges(X)|}. \end{aligned} \tag{6.13}$$

Come si evince da 6.13, calcolare il valore esatto di $E|odd(X)|$, richiede l'iterazione di tutte le possibili configurazioni X, così da poter valutare la sommatoria $\sum_{X:|odd(X)|=2i} \lambda^{\sharp edges(X)}$.

6.1.2 Algoritmi sviluppati

Innanzitutto, è bene ricordare la definizione di sottografo e *spanning subgraph*: un grafo H si dice sottografo di un grafo G se i vertici di H sono un sottoinsieme dei vertici di G e gli archi di H sono un sottoinsieme degli archi di G. Siano $G = (V, E)$ ed $H = (V_1, E_1)$ due grafi. H è un sottografo di G se e solo se $V_1 \subseteq V$ ed $E_1 \subseteq E$. Uno *spanning subgraph* H di un grafo G è un sottografo che contiene tutti i vertici di G, cioé $V_1 = V$.
È semplice intuire che, supponendo di avere un grafo input $G = (V, E)$ con $|E| = m$, il numero di possibili *spanning subgraph* è pari a 2^m. La strategia adottata, per l'iterazione degli *spanning subgraph* è quella di mappare ogni sottografo con una stringa di bit di lunghezza m, in cui il bit i-esimo rappresenta l'arco i-esimo, con 0 se non presente in tale configurazione o 1 se presente.
Una volta effettuato tale mapping è possibile enumerare le varie stringhe, per poter contare il grado di ogni nodo e verificare il numero di nodi a grado dispari per tale configurazione, così da poter calcolare il valore atteso di $odd(X)$ secondo la formula 6.13.

Questa prima versione dell'algoritmo `Estimate_F` può essere vista come un approccio *naive* alla risoluzione del problema. Infatti non tiene conto

Algorithm 1 Estimate $odd(X)$ - *naive* version

```
1: procedure ESTIMATE_F(graph)
2:     summation ← 0
3:     Z', μ, λ ← compute_Z'(graph)
4:     for subgraph ∈ spanning_subgraphs_iterator(graph) do
5:         odds ← count_odds(subgraph)                ▷ Odd degree nodes
6:         summation ← summation + odds · μ^odds · λ^|edges(subgraph)|
7:     end for
8:     return summation / Z'
9: end procedure
```

del tempo impiegato nell'iterazione di tutti i possibili *spanning subgraph* del grafo in input.
Inoltre, da una analisi dei valori di *odds* ed $|edges(subgraph)|$, si evince che il maggior contributo apportato alla sommatoria mostrata in 6.13 deriva dai sottografi iniziali, cioè quei sottografi in cui c'è una maggior variazione negli archi presenti. Ad esempio, la figura 6.2 mostra quali sono i sottografi a

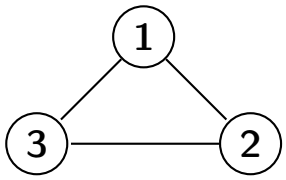
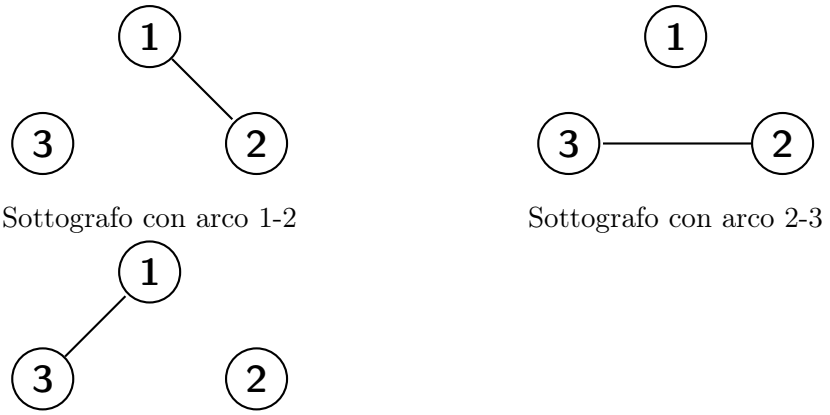


Figura 6.1: Grafo Input

maggior contributo per il calcolo di $E|odd(X)|$ sul grafo in input 6.1. Tenendo in considerazione tale osservazione, si è passati ad una seconda versione dell'algoritmo che riducesse i tempi d'esecuzione, andando ad analizzare solamente gli *spanning subgraph* a contributo maggiore. Tale obiettivo è stato raggiunto analizzando i primi sottografi aventi un numero di archi non superiore a $log\, m$. Purtroppo, l'algoritmo 2 soffre di inaccuratezza della stima di $E|odd(X)|$, in quanto l'imposizione di una soglia comporta, inevitabilmente, la perdita di una parte, seppur meno significativa, della sommatoria, andando così ad ottenere alla fine un valore sottostimato.
Per evitare che ciò accadesse, si è pensato di adottare una tecnica di approssimazione del contributo apportato dai restanti *spanning subgraph*. Tale



Sottografo con arco 1-2

Sottografo con arco 2-3

Sottografo con arco 3-1

Figura 6.2: Sottografi a contributo maggiore

contributo è identificato per semplicità da X_2, mentre la quantità derivante dai *subgraph* a contributo maggiore è indicata da X_1. In particolare, il contributo dei *subgraph* è dovuto al valore di $\lambda^{|edges(X)|_{max}}$: essendo λ un reale compreso tra 0 ed 1 è ovvio che decresca al crescere dell'esponente. Proprio per questo, solo poche configurazioni apportano un contributo significativo, mentre ne esistono numerose aventi $\lambda^{|edges(X)|_{max}}$ molto basso a causa dell'esponente elevato.
L'idea alla base del calcolo dell'approssimazione di X_2 consiste nel suddividere il resto dei sottografi in k gruppi e calcolarne il contributo per ognuno

Algorithm 2 Estimate $odd(X)$ - *threshold* version

```
1: procedure ESTIMATE_F(graph, threshold)
2:     summation ← 0
3:     Z', μ, λ ← compute_Z'(graph)
4:     for subgraph ∈ spanning_subgs_iterator(graph, threshold) do
5:         odds ← count_odds(subgraph)                ▷ Odd degree nodes
6:         summation ← summation + odds · μ^odds · λ^|edges(subgraph)|
7:     end for
8:     return summation, Z', μ, λ
9: end procedure
```

di essi, così da avere una stima più precisa.
Per ogni gruppo:

- si conti n_g, il numero di elementi appartenenti al gruppo, (somma dei coefficienti binomiali da *start* ad *end*)
- si calcoli m_g, il massimo numero di edges nel gruppo
- si valuti il minimo di f_g nel gruppo, ottenuto in corrispondenza del minimo numero di edges o del massimo
- si calcoli $n_g \cdot \lambda_g^m \cdot f_g$ e lo si sommi alla stima

Alla fine di tale processo la stima accumulata sarà pari ad $E|odd(X)|$.
Di fondamentale importanza è la scelta di k, ovvero del numero di grup-

Algorithm 3 Approximate X_2

```
1: procedure APPROXIMATE_SUBGS(m, f, λ, k)
2:     approx_sum ← 0
3:     step ← (m − log(m)) / k
4:     for start ← log(m) + 1 to m by step do
5:         if start + step ≤ m then
6:             end ← start + step
7:         else
8:             end ← m
9:         end if
10:        n_g ← 0
11:        m_g ← end
12:        for j ← start to end do
13:            n_g ← n_g + (m choose j)
14:        end for
15:        f_g ← min(f(start), f(end))
16:        approx_sum ← approx_sum + n_g · λ^{m_g} · f_g
17:    end for
18:    return approx_sum
19: end procedure
```

pi da creare: numerosi esperimenti hanno mostrato che la stima di X_2 si mantiene fissa ad un valore, a partire da un determinato k, in particolare ciò si verifica da $k = \frac{(m-log(m))}{2} + 1$ in poi. Quindi si è ritenuto opportuno fissare tale valore come numero di gruppi in cui suddividere i *subgraph* in

base al numero di archi, a partire da $log(m) + 1, log(m) + (m - log(m))/k$, aumentando di $(n - log(m))/k$, fino ad arrivare ad m.
L'algoritmo di approssimazione `Approximate_Subgs` [3] necessita dei parametri di input m e λ, mostrando quindi una dipendenza dall'algoritmo `Estimate_F` [2], in quanto è quest'ultimo a restituire tali valori in output.
Oltre a λ ed m, l'algoritmo `Approximate_Subgs` [3] riceve in input anche la funzione da valutare agli estremi di ogni intervallo: tale funzione è $f(x) = 2x \cdot \mu^{2x}$.
Quindi, la stima dinamica di $E|odd(X)|$, si riduce a

Algorithm 4 Compute $E|odd(X)|$

1: **procedure** COMPUTE_E_ODD_X($graph, threshold, m, k$)
2: $f(x) = 2x \cdot \mu^{2x}$
3: $summation, Z', \mu, \lambda \leftarrow Estimate\_F(graph, threshold)$
4: $X_2 \leftarrow Approximate\_Subgs(m, f, \lambda, k)$
5: $E|odd(X)|_{graph} \leftarrow (summation + X_2)/Z'$
6: **return** $E|odd(X)|_{graph}$
7: **end procedure**

6.1.3 Enumerazione degli spanning subgraph: algoritmo L

Si analizza in questa sezione il modo in cui vengono efficacemente enumerate tutte le possibili configurazioni di un grafo.
Come anticipato in precedenza, si è scelto di enumerare gli *spanning-subgraph* in maniera analoga a come si enumerano le possibili sequenze di n bit. Ad esempio, avendo un grafo $G = (V, E)$ rappresentato come sequenza di archi $G = [(1, 2), (2, 3), (3, 1)]$, può avere 2^3 possibili configurazioni, come visibile dalle righe della tabella 6.1. Esistono diversi modi per generare tutte le permutazioni di una data sequenza. Un classico algoritmo, che è allo stesso tempo semplice e flessibile, è basato sul trovare la prossima permutazione in ordine lessicografico, se esiste. Può gestire valori ripetuti, andando a generare i distinti sottoinsiemi di permutazioni per ognuno di essi. Anche per permutazioni semplici è significativamente più efficiente rispetto al generare i valori per il codice di Lehmer in ordine lessicografico (possibilmente utilizzando il sistema di numerazione fattoriale) e convertendo tali permutazioni. Per utilizzarle, è necessario ordinare la sequenza in maniera crescente (il che fornisce la sua permutazione minimale lessicografica), ed in seguito ripete, avanzando alla prossima permutazione fin quando ve n'è una. Questo metodo risale a Narayana Pandita, sviluppato nel XIV secolo in India, ed è

Tabella 6.1: Rappresentazione delle possibili configurazioni

$(1,2,0)$ $(1,3,0)$ $(2,3,0)$
$(1,2,0)$ $(1,3,0)$ $(2,3,1)$
$(1,2,0)$ $(1,3,1)$ $(2,3,0)$
$(1,2,1)$ $(1,3,0)$ $(2,3,0)$
$(1,2,0)$ $(1,3,1)$ $(2,3,1)$
$(1,2,1)$ $(1,3,0)$ $(2,3,1)$
$(1,2,1)$ $(1,3,1)$ $(2,3,0)$
$(1,2,1)$ $(1,3,1)$ $(2,3,1)$

stato più volte riscoperto da allora, in particolare la versione utilizzata in questo lavoro è l'algoritmo detto *"L"* descritto da Donald Knuth in *"The Art of Computer Programming"* [42].

Algorithm 5 Algorithm L

procedure UNIQUE_PERMUTATIONS(a)
    Trovare il maggior indice k tale che $a[k] < a[k+1]$.
    if tale indice non esiste **then**
        La permutazione è l'ultima
    end if
    Trovare il maggior indice $l > k$ tale che $a[k] < a[l]$
    Sostituire il valore $a[k]$ con quello di $a[l]$
    Invertire la sequenza da $a[k+1]$ fino ad $a[n]$ incluso
    return a
end procedure

Capitolo 7

Implementazione e testing

Lo studio dell'articolo di Jerrum e Sinclair, l'analisi dei teoremi e dell'algoritmo derivante, la realizzazione dei miglioramenti nella stima della *partition function* e del lavoro svolto sul *mean magnetic moment* mostrati nei capitoli 5 e 6 sono sempre stati accompagnati da implementazione e testing. In tal modo si è potuto costruire in maniera incrementale, a partire da ciò che era già stato sviluppato in precedenza, analizzando accuratamente gli effetti delle scelte intraprese, decidendo di conseguenza sulla strada da percorrere. Si descrive ora, nel dettaglio, come sono state realizzate le fasi di implementazione e testing.

7.1 Implementazione

Gli algoritmi proposti nei Capitoli 5 e 6 sono stati implementati utilizzando il linguaggio **Python**.

Python. È un linguaggio di alto livello pubblicato nel 1991 dal suo autore *Guido Van Rossum*: è general-purpose, interpretato, dinamico e largamente utilizzato. La sua filosofia di progettazione enfatizza la leggibilità del codice, e la sua sintassi permette ai programmatori di esprimere concetti in un minor numero di linee di codice rispetto ad altri linguaggi come C++ o Java. Il linguaggio fornisce costrutti pensati per produrre programmi su piccola e larga scala.
Python supporta molteplici paradigmi di programmazione, inclusa quella object-oriented, imperativa, funzionale e procedurale. È caratterizzato da un sistema di tipizzazione dinamico, dalla gestione automatica della memoria ed ha una libreria standard completa. Gli interpreti Python sono disponibili

per i più diffusi sistemi operativi. Il design di Python offre supporto alla

Figura 7.1: Logo del linguaggio Python

programmazione funzionale nella tradizione Lisp. Il linguaggio è dotato delle funzioni *map() reduce()* e *filter()*; *list comprehension*, *dizionari*, *insiemi* e *generators*.
La filosofia al cuore del linguaggio è riassunta dal documento "*The Zen of Python*" (PEP 20), che include aforismi come:

- *Beautiful is better than ugly*
- *Explicit is better than implicit*
- *Simple is better than complex*
- *Complex is better than complicated*
- *Readability counts*

Un obiettivo importante degli sviluppatori Python è quello di rendere divertente il suo utilizzo, ciò è rispecchiato anche dal nome che deriva dai "*Monty Python*", il celebre gruppo comico britannico, la cui commedia acutamente intellettuale riscosse grande successo tra gli anni '60 ed '80.

7.1.1 Librerie utilizzate

Se la potenza di un linguaggio di programmazione può essere misurata in librerie di cui dispone, allora Python è da considerarsi un linguaggio estremamente potente. Il *Python Package Index* (PyPI), che è il più grande repository di software per il linguaggio Python, attualmente contiene 86483 pacchetti, ognuno dei quali liberamente scaricabile ed integrabile all'interno della propria applicazione, essendo rilasciato secondo una delle molteplici licenze Open Source.

NumPy. È il pacchetto scientifico fondamentale per fare computazione scientifica in Python. Fornisce al linguaggio l'oggetto array multidimensionale, vari oggetti derivati (come array maschera e matrici) ed un vasto assortimento di routine per eseguire velocemente operazioni su array, incluse

operazioni matematiche, logiche, manipolazioni di forma, ordinamenti, selezioni, I/O, trasformate di Fourier discrete, algebra lineare di base, operazioni statistiche di base, simulazioni casuali e molto altro. Maggiori dettagli alla pagina dedicata [43].

joblib. È un insieme di strumenti che forniscono supporto per il calcolo parallelo in Python. In particolare, joblib offre:

- Caching su disco trasparente dei valori di output e lazy re-evaluation (pattern memoize)
- Semplici strumenti di supporto alla computazione parallela
- Logging e tracing dell'esecuzione

Joblib è ottimizzato per essere veloce e robusto, in particolare su molti dati ed ha specifiche ottimizzazioni per gli array numpy. Maggiori dettagli alla pagina dedicata [44].

7.1.2 Strutture e librerie implementate

Strutture

Grafo. Il sistema di Ising ferromagnetico è rappresentato attraverso un grafo e costituisce l'input dell'algoritmo. Si è scelto di strutturare il grafo come una lista di triple, di cui ogni tripla rappresenta un arco: estremi dell'arco ed energia di interazione. Per ulteriori dettagli consultare 2.2.
Per convenienza, in alcuni casi, è stata anche utilizzata la classica rappresentazione in dizionario, le cui entry hanno come chiave il nodo ed i valori ad essa associati sono i suoi vicini.

Sottografo. Un sottografo è rappresentato anch'esso come lista di triple, le cui tuple sono gli estremi del grafo ed un numero binario che indica se tale arco è presente nel sottografo in questione.

Librerie

Durante l'implementazione di questo lavoro di tesi, è stato necessario costruire delle librerie che offrissero supporto alla manipolazione di grafi e sottografi, oltre a varie procedure di generazione di grafi casuali ed iterazione di bitset ordinati.
Lo stile di scrittura del codice implementato rispecchia le linee guida del

linguaggio, concentrandosi in particolar modo sulla leggibilità e l'efficienza. Infatti lo stile di scrittura "*pythonico*" porta a sfruttare feature del linguaggio mirate a rendere l'operazione da compiere semplice ed efficiente. Un riferimento da cui prendere spunto per maggiori dettagli è disponibile qui [45].

Graph Utils. Permette di eseguire le operazioni più comuni da effettuare allo startup dell'algoritmo, cioè la lettura del sistema ferromagnetico da file e trasformarlo nella rappresentazione dizionario del grafo associato. Offre inoltre una procedura utile al cambio di rappresentazione da dizionario a lista di coppie ed infine una procedura che aggiunge ad una lista di coppie l'energia di interazione (di default posta ad 1). Per maggiori dettagli sul concetto di energia di interazione consultare 4.1.

Subgraph Utils. È la libreria maggiormente utilizzata in quanto offre procedure di creazione, manipolazione ed iterazione dei subgraph world, in particolare, è possibile iterare bitset (ordinati e non) di lunghezza arbitraria, subgraph (ordinati e non) a partire dal grafo di input. Infine vi è un iteratore (*generator* nella terminologia di Python) di sottografi che fornisce tale struttura in formato lista di archi, ognuno dei quali già provvisto di energia di interazione.
La realizzazione di *generator* avviene attraverso l'utilizzo della parola chiave ***yield***. È necessario innanzitutto specificare cosa sia un *generator*: si tratta di un iteratore, ma su cui è possibile iterare una sola volta in quanto esso non memorizza il valore, ma lo genera *on-the-fly*.
La parola chiave ***yield*** è utilizzata come *return*, a parte il fatto che la funzione restituirà un generatore. Tale costrutto è di vitale importanza quando si ci si trova a dover restituire una grande quantità di dati di cui ci sarà bisogno di leggerli solo una volta, ed è proprio il caso dell'enumerazione di tutti i possibili sottografi di un grafo.
Altra accortezza da tenere a mente qualora si abbia a che fare con l'iterazione su grandi quantità di dati è quella di utilizzare ***xrange*** ovunque possibile. Tale funzione, ha la medesima funzionalità della ben nota *range*, ma a differenza di quest'ultima è un generatore ed in quanto tale permette di risparmiare enormi quantità di memoria (a meno che gli oggetti restituiti non vengano salvati), grazie al fatto esiste un solo oggetto nello stesso istante. Quando *xrange* è invocata, crea una lista contenente il numero di oggetti richiesto: questi sono tutti create contemporaneamente ed esistono nello stesso istante; come è facile intuire, ciò si rivela un problema quando il

numero di oggetti è considerevole. D'altro canto *xrange*, non crea alcun oggetto immediatamente, ma vengono creati solo quando si inizia ad utilizzare il generatore iterando su di esso.

Random Graph. Offre supporto alla creazione di grafi random che rispettano il formato lista di triple, le cui tuple sono gli estremi e l'energia di interazione. Le procedure implementate consentono la creazione di grafi casuali a partire dal numero di nodi desiderati e la probabilità di creazione dell'arco, con la possibilità di specificare o meno il numero di archi richiesti.

Unique Permutation. Fornisce una procedura di iterazione su tutte le permutazioni uniche di una sequenza, in maniera efficiente. Consiste nell'implementazione Python dell'algoritmo L di D. Knuth. Per maggiori dettagli consultare 6.1.3.

Steps Manager. Offre una procedura di generazione del numero di passi da eseguire nella simulazione della partition function Z, in funzione del numero di archi, parametri di precisione descritti e versione dell'algoritmo da utilizzare.

7.2 Testing

In questa sezione, si mostrano e descrivono i risultati dell'esecuzione degli algoritmi implementati nella sezione precedente.
La fase di *testing*, come già accennato, è stata di fondamentale importanza durante lo sviluppo del codice in quanto ha consentito di verificare se le ottimizzazioni apportate ai teoremi stessero portando ai risultati attesi.
Innanzitutto è stato necessario confrontarsi con i risultati ed i tempi ottenuti in [6] utilizzando grafi di dimensioni simili e parametri equivalenti, in seguito si è potuto esplorare la bontà del codice sviluppato aumentando le dimensioni dell'input e facendo variare i parametri.
I parametri su cui ci si è concentrati sono B, β ed ϵ. Tenendo presente la definizione di $\mu = tanh\, B\, \beta$ fornita in 4.4, in accordo al Teorema 4.2, tale quantità incide sulla complessità dell'algoritmo del generatore, per cui sono stati scelti dei valori che permettono di ottenere un $\mu \approx 1$: tale scelta comporta due vantaggi, cioè annullare quasi del tutto il contributo che μ apporta al numero di passi del generatore e ridurre il numero di iterazioni richieste dallo *Step 2* per il calcolo di Z. Infatti, come descritto in 4.14 e 4.15, avere μ prossimo ad 1 implica avere pochi valori intermedi μ_k da calcolare e quindi

consente di ridurre il numero k di iterazioni da eseguire.
Il parametro β, già analizzato nel capitolo 3 "*Logit Dynamics*", rappresenta il livello di *razionalità* del sistema, nella maggior parte dei test sono stati utilizzati valori bassi di β in quanto si è interessati a sapere cosa succede quando vi è bassa razionalità e quindi i giocatori hanno poca probabilità di giocare la loro *best response*. Sono stati effettuati anche test al variare di questo parametro per verificare la bontà dell'algoritmo.
Per quanto riguarda il parametro ϵ, esso specifica l'accuratezza desiderata, per cui è stato scelto sempre molto basso.
Ultimo parametro da analizzare è t, descritto nel Lemma 4.1. Tale valore risulta essere maggiore di 1, quindi è sufficiente porlo pari a 2.
Nei paragrafi successivi verranno mostrati i risultati dei test ottenuti in diverse configurazioni, concentrandosi in particolare al confronto con la versione *JS* proposta da Jerrum e Sinclair, e con l'algoritmo proposto in [6], a cui d'ora in avanti si farà riferimento come algoritmo Partition.

Configurazione Hardware e Software I test sono stati effettuati su un server messo a disposizione dal Dipartimento di Ingegneria dell'Informazione ed Elettrica e Matematica Applicata (DIEM) dell'Università degli Studi di Salerno. La macchina virtuale VMWare dedicata all'esecuzione dei test era configurata nel seguente modo:

- Processore: AMD Opteron 6376
- RAM: 16GB
- Disco: 70GB
- Kernel Linux: 3.13.0-83-generic x86_64
- Sistema Operativo: Ubuntu 14.04.4 LTS
- Python: 2.7.6
- GCC: 4.8.2
- NumPy: 1.11.0

7.2.1 Confronto Preliminare

Il primo confronto è un paragone preliminare che aiuta a rendersi conto dell'effettivo miglioramento introdotto dalla nuova versione dell'algoritmo che

Algoritmo	B	β	s	t	ϵ	n^o passi	Tempo
JS	20	0.4	79778	25	0.9	1994450	1g 17h 45min
Partition	20	0.4	1	2	0.9	3	406 ms
Partition2	20	0.4	1	2	0.9	2	284 ms

Tabella 7.1: n=5, m=10, B=20, β=0.4.

Algoritmo	B	β	s	t	ϵ	n^o passi	Tempo
JS	20	0.1	79778	25	0.9	1994450	1g 18h 56min
Partition	20	0.1	58	2	0.9	116	863 ms
Partition2	20	0.1	1	2	0.9	116	292 ms

Tabella 7.2: n=5, m=10, B=20, β=0.1.

verrà identificata come "Partition2". Vengono messi a confronto i running time dell'algoritmo JS, Partition ed Partition2. Le seguenti tabelle mostrano i tempi di esecuzione dei vari algoritmi sullo stesso grafo composto da 5 nodi e 10 archi. Da questi primi test è evidente come l'algoritmo Partition abbia ridotto in maniera impressionante il tempo d'esecuzione dell'algoritmo iniziale, rendendo tale procedura utilizzabile in pratica, ma si può già intuire che l'algoritmo Partition2 proposto da questo lavoro sia più efficiente. Infatti, nonostante la taglia esigua dell'input di questi primi test, è già possibile apprezzare il miglioramento del running time che in Tabella 7.1 dimezza ed in Tabella 7.2 si riduce a quasi un terzo.
Mentre l'algoritmo JS non può essere applicato con un valore di precisione ϵ basso, a causa del numero di iterazioni che cresce a dismisura, con l'algoritmo Partition ciò può avvenire, ma restando sullo 0.1, invece l'algoritmo Partition2 consente di scendere anche a valori di 10^{-4} o 10^{-6} in funzione della taglia dell'input, avendo comunque un tempo d'esecuzione accettabile. Le seguenti tabelle mettono a confronto l'algoritmo Partition e l'algoritmo Partition2 considerando $\epsilon = 0.1$: In questo caso è molto evidente la maggiore efficienza dell'algoritmo Partition2, nel calcolo dell'approssimazione di Z. È possibile verificare dalla Tabella 7.5 come al decrescere di ϵ aumentino

B	β	s	t	ϵ	n^o passi	Tempo
20	0.4	1	2	0.1	3	406 ms
20	0.1	2156	2	0.1	4321	17s 335 ms

Tabella 7.3: $\epsilon = 0.1$ algoritmo Partition.

B	β	s	t	ϵ	n^o passi	Tempo
20	0.4	1	2	0.1	2	291 ms
20	0.1	2156	2	0.1	4312	1s 605 ms

Tabella 7.4: $\epsilon = 0.1$ algoritmo Partition2.

ϵ	Tempo (s)
0.1	0.316015
0.2	0.295347
0.3	0.295403
0.4	0.295231
0.5	0.295524
0.6	0.294863
0.7	0.293953
0.8	0.309947
0.9	0.292618
1.0	0.298054

Tabella 7.5: Tempi algoritmo Partition2 n=10, ϵ crescente.

i tempi di esecuzione, seppur di poco, ciò è dovuto al numero di step da effettuare che, si ricorda da 4.26, 4.29 e 5.1, che s dipende da ξ, il quale varia in funzione di ϵ ed n, mentre t dipende da η che è funzione di n.
Quindi all'aumentare del numero di nodi, aumentano anche i suddetti valori; in particolare l'algoritmo Partition, sebbene sia polinomiale, non può essere utilizzato in applicazioni pratiche che spesso presentano un gran numero di nodi. L'algoritmo Partition2 presentato in questo lavoro di tesi punta ad ottimizzare il numero totale degli *st* step da eseguire, in ognuno dei quali simulare la catena di Markov per il numero di passi, inizialmente come specificato dal Teorema 4.2, poi migliorato dal nuovo bound 5.8 che ricordiamo essere pari a $2m^2\mu^{-4}w(I)w(F)$, cioè nel nostro caso si concretizza in $2m^2\mu^{-4}(log\frac{1}{\delta}+1)$.

7.2.2 Dimensioni

Gli esperimenti che seguono, considerano un numero di nodi ed archi crescente e per ogni esecuzione dell'algoritmo si hanno questi parametri di input: $B = 20, \beta = 0.4, \epsilon = 0.1$.
L'asse delle ascisse dei grafici riportano il numero di archi del grafo di input, mentre le ordinate indicano il tempo d'esecuzione. È importante osservare la scala temporale utilizzata in quanto può risultare molto differente nel confronto tra l'algoritmo Partition2 e l'algoritmo Partition. Come è

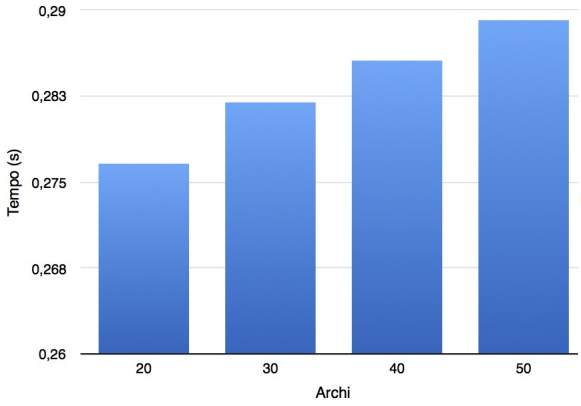


Figura 7.2: Algoritmo Partition2

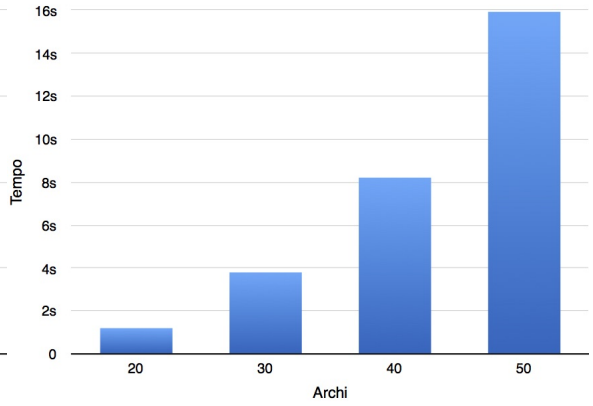


Figura 7.3: Algoritmo Partition

n = 10

possibile notare dai risultati, l'algoritmo Partition2 ha un netto margine di guadagno in termini di efficienza rispetto all'algoritmo Partition, il quale inizia a soffrire già con un grafo avente 20 nodi ed 90 archi, superando il minuto d'esecuzione contro i 250 ms della versione proposta da questo lavoro di tesi. Di seguito viene mostrato l'andamento del running time per l'algoritmo Partition2 al variare della taglia del grafo nel numero di nodi ed archi.

L'esecuzione riportata in Tabella 7.6 è stata tenuta fuori dal precedente grafico per motivi di scala: è possibile apprezzare l'efficienza dell'algoritmo Partition2 anche in questo caso, in quanto impiega circa 400 secondi per calcolare la funzione di partizione su un grafo con 500 nodi e 5000 archi, che già si avvicina ad un'istanza del mondo reale.
Grazie ai miglioramenti ottenuti, si è scelto di eseguire anche un test su grafi

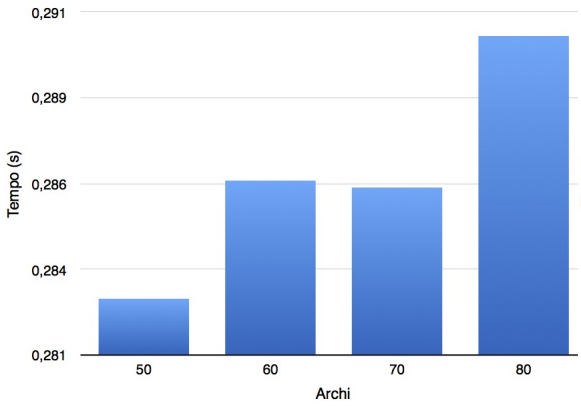


Figura 7.4: Algoritmo Partition2

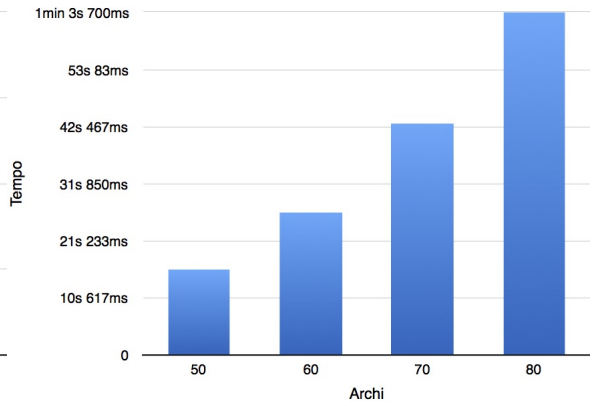


Figura 7.5: Algoritmo Partition

n = 15

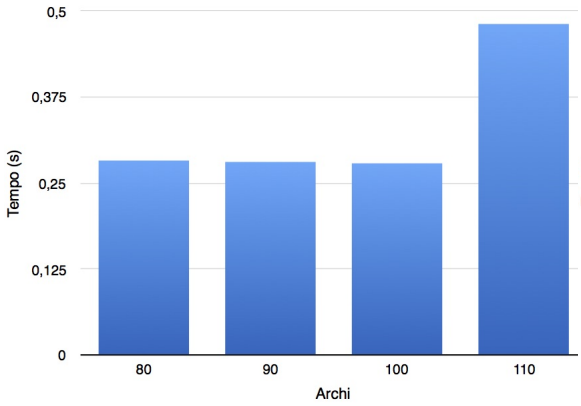


Figura 7.6: Algoritmo Partition2

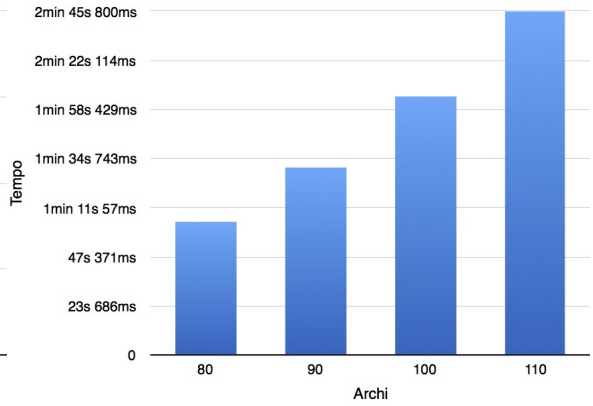


Figura 7.7: Algoritmo Partition

n = 20

di dimensioni maggiori, sempre al variare dei parametri n ed m, mantenendo i parametri $B = 20, \beta = 0.4, \epsilon = 0.1$ invariati.

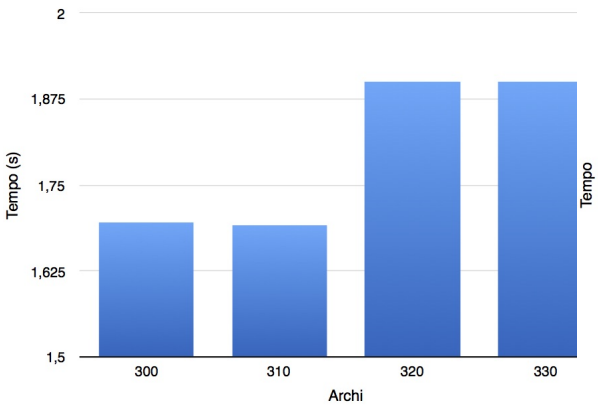


Figura 7.8: Algoritmo Partition2

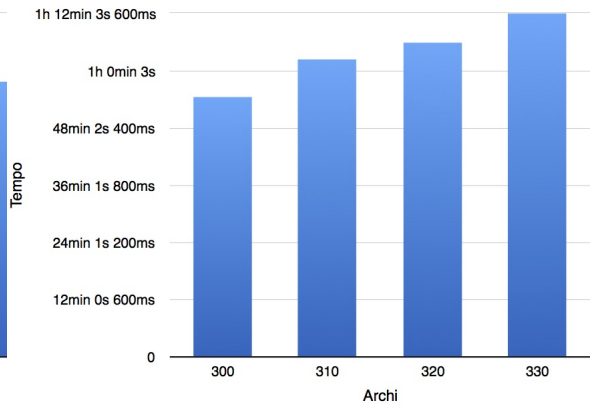


Figura 7.9: Algoritmo Partition

$n = 35$

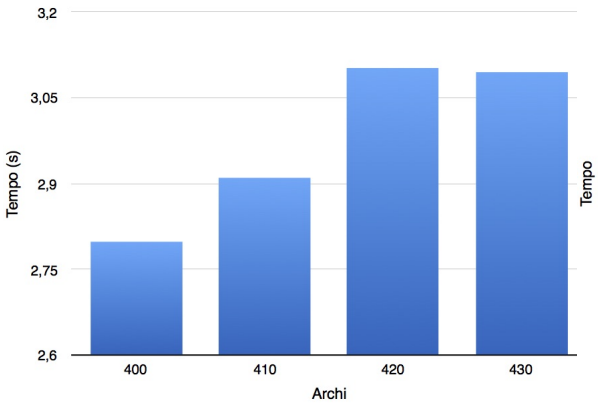


Figura 7.10: Algoritmo Partition2

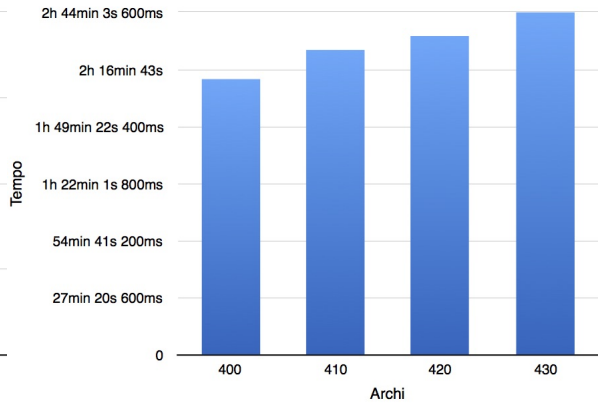


Figura 7.11: Algoritmo Partition

$n = 40$

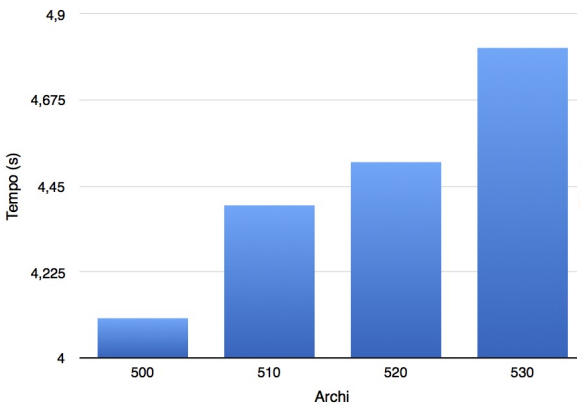


Figura 7.12: Algoritmo Partition2

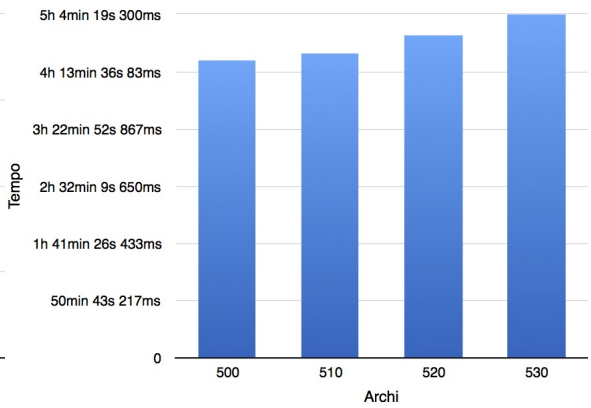


Figura 7.13: Algoritmo Partition

$$n = 45$$

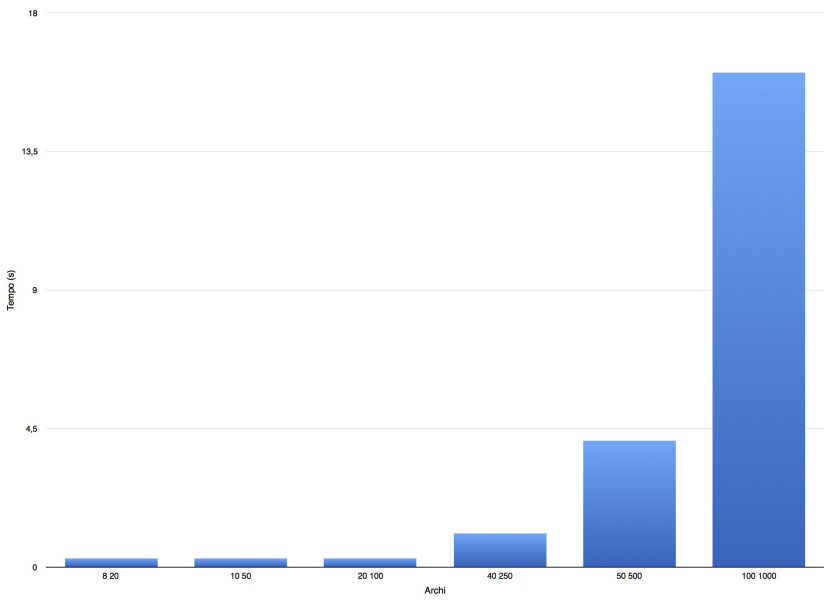


Figura 7.14: Algoritmo Partition2 al variare di n ed m.

n	m	B	β	ϵ	s	t	Tempo (s)
500	1000	20	0.4	0.1	1	2	399.465704

Tabella 7.6: Algoritmo Partition2 $n = 500$, $m = 5000$.

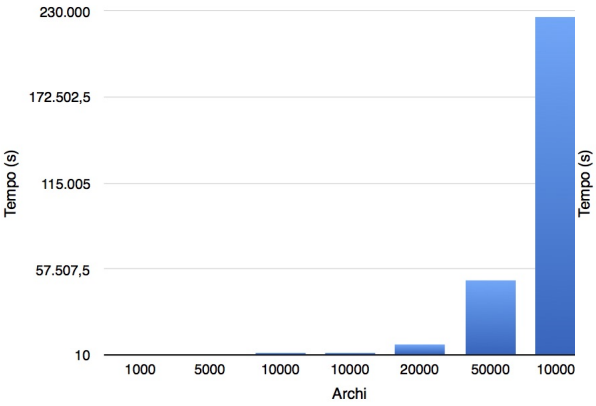


Figura 7.15: Scala lineare

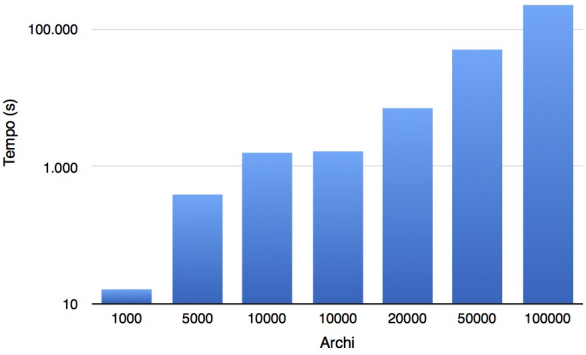


Figura 7.16: Scala logaritmica

Grafi di dimensioni maggiori.

7.2.3 Parametro ϵ

I seguenti grafici riportano i risultati ottenuti nei test di accuratezza. Il parametro ϵ è stato fatto variare da 10^{-1} a 10^{-4}.
È possibile notare come il tempo aumenti notevolmente nel passaggio da $\epsilon = 10^{-3}$ a $\epsilon = 10^{-4}$, ma sia comunque accettabile anche considerando il fatto che l'algoritmo Partition non è in grado di ottenere tempi ragionevoli già a partire da ϵ minore di 10^{-1}.

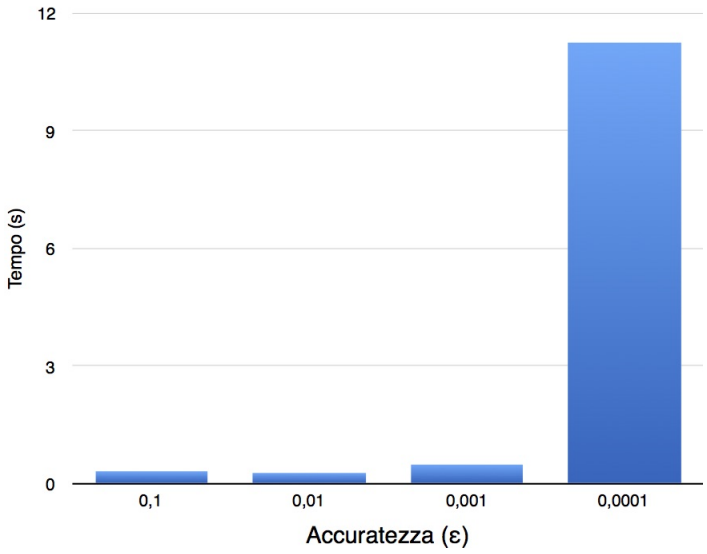


Figura 7.17: Algoritmo Partition2 $n = 8, m = 20$.

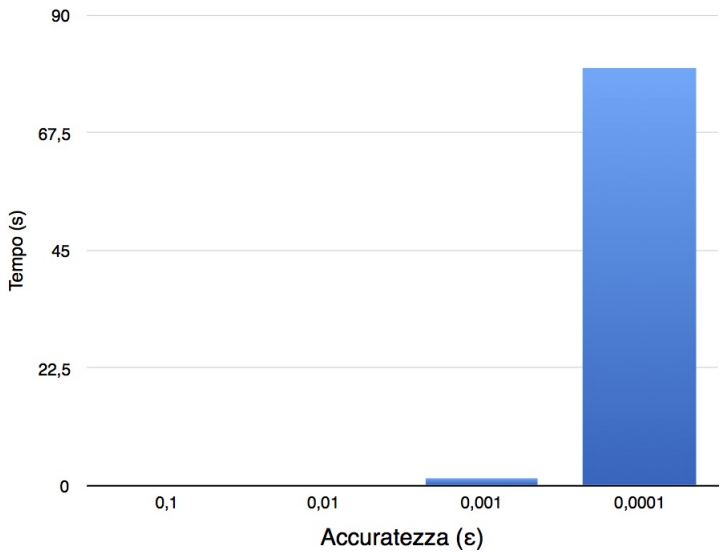


Figura 7.18: Algoritmo Partition2 $n = 10, m = 50$.

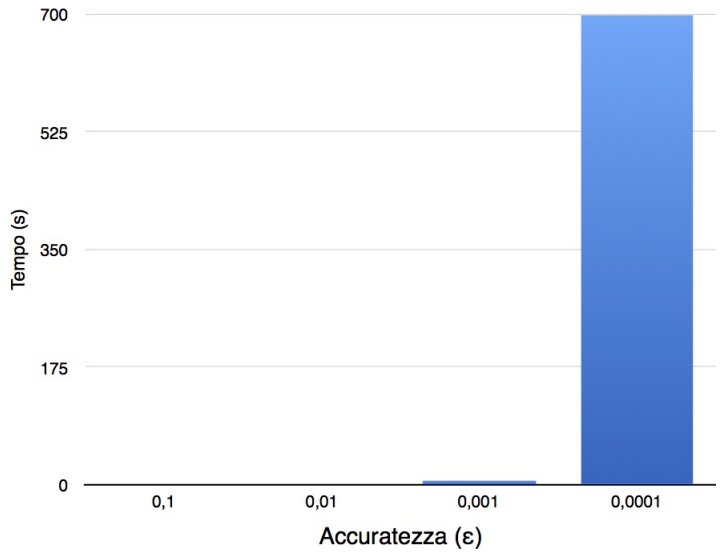


Figura 7.19: Algoritmo Partition2 $n = 20, m = 100$.

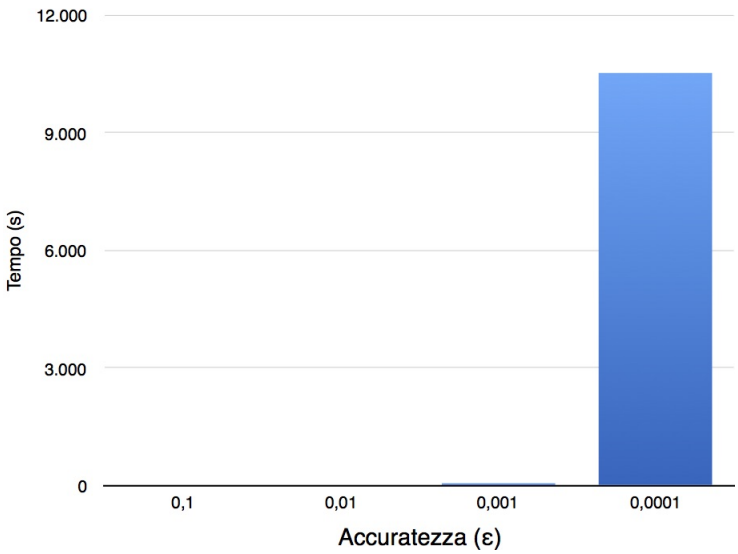


Figura 7.20: Algoritmo Partition2 $n = 40, m = 250$.

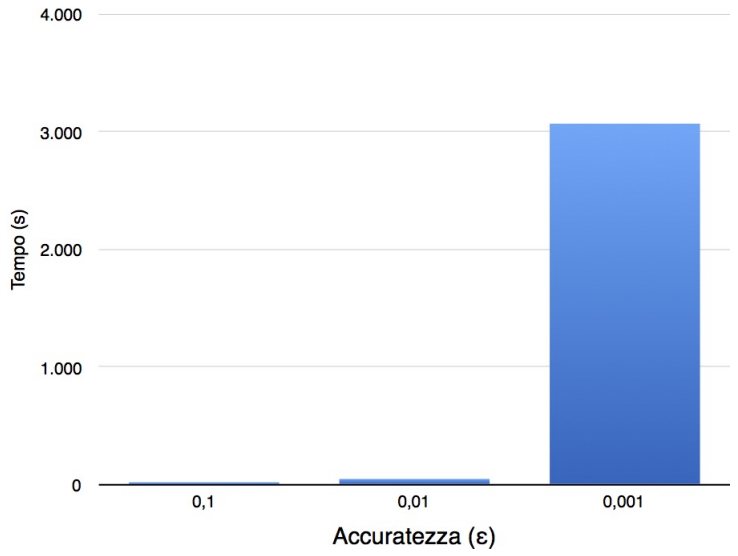


Figura 7.21: Algoritmo Partition2 $n = 100, m = 1000$.

B	β	μ
20	0.4	0.99999977492967584
15	0.122119	0.95000040724956369
10	0.147222	0.90000009697914363
12	0.104679	0.84999866466785234
10	0.109861	0.799999176077972
8	0.121619	0.74999865489104856

Tabella 7.7: Parametri utilizzati.

7.2.4 Parametro μ

I grafici presenti in questa sezione rappresentano l'andamento dell'algoritmo Partition2 al variare dei valori B e β. Come si ricorda in 4.4, da tali valori dipende μ e di conseguenza sulla complessità del generatore atto a simulare la catena di Markov, per il Teorema 4.2. Il valore di B è stato fatto variare da 20 ad 8 in tutti i test seguenti, modificando di conseguenza β (parametro di razionalità) per mantenere μ in un range tra 1 e 0.75, come mostrato in Tabella 7.7. È possibile notare dai risultati ottenuti come, al decrescere

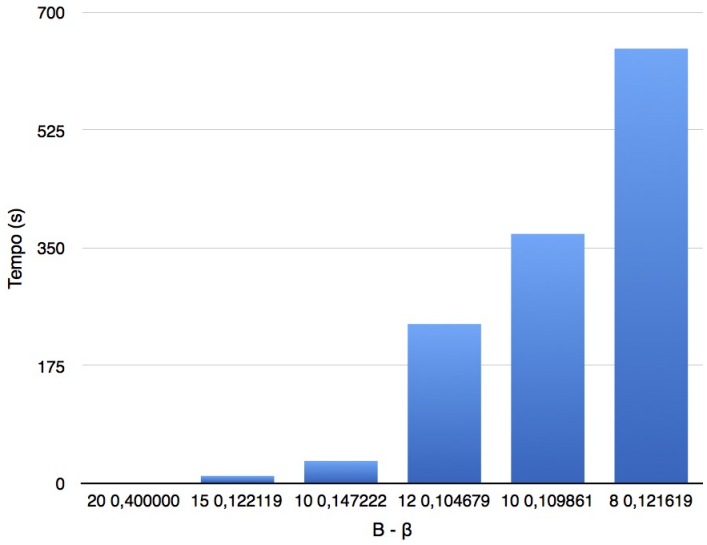


Figura 7.22: Algoritmo Partition2 $n = 8, m = 20$.

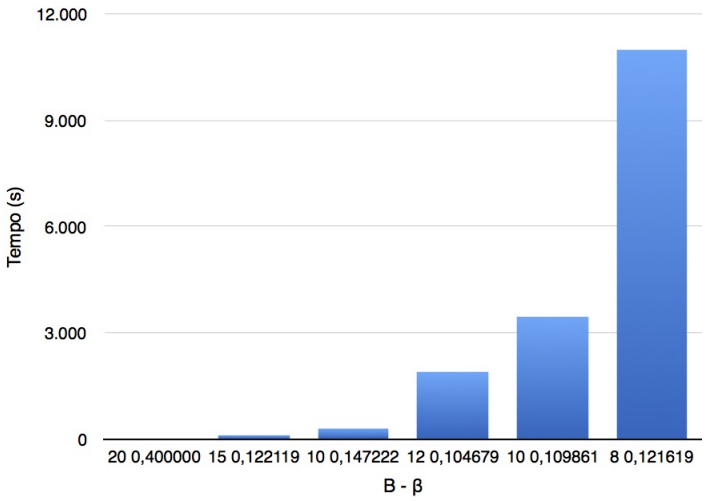


Figura 7.23: Algoritmo Partition2 $n = 10, m = 50$.

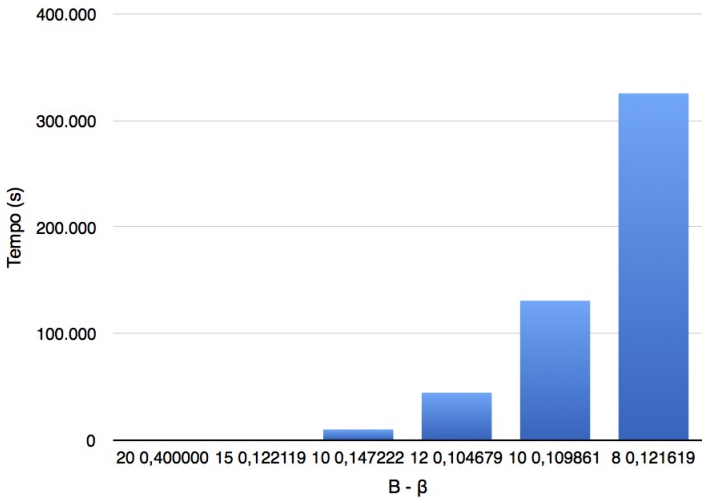


Figura 7.24: Algoritmo Partition2 $n = 20, m = 100$.

di μ il tempo cresca piuttosto linearmente, confermando quanto indicato dal Teorema 4.2.

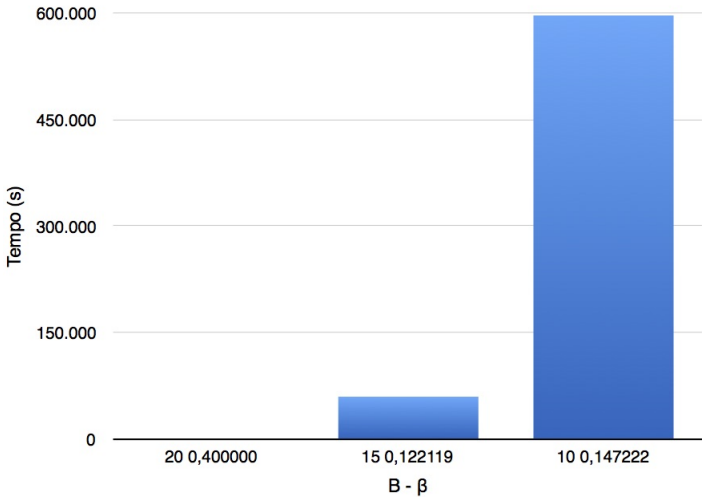


Figura 7.25: Algoritmo Partition2 $n = 40, m = 250$.

7.3 The Effectiveness of Advertising

Questa sezione presenta un esempio di applicazione dell'algoritmo sviluppato: la simulazione di una campagna pubblicitaria. si hanno due prodotti da promuovere e si è interessati a calcolare la *magnetizzazione* del sistema, ovvero il numero atteso di persone che sceglieranno di utilizzare un prodotto piuttosto che l'altro. Per raggiungere tale scopo è necessario sfruttare il lavoro proposto nel Capitolo 6 riguardante il *mean magnetic moment*.

7.3.1 Scenario

Si consideri una *network* $G = (V, E)$ con $|V| = n$, in cui ogni vertice rappresenta un giocatore (agente) avente a disposizione due possibili scelte, cioè i due prodotti, detti anche *strategie*, denotati da 0 e 1.
Ad ogni agente della rete è assegnato un *profilo* $x = (x_1, \cdots, x_n) \in \Omega = 0, 1^n$.
Ad ogni arco $e = (u, v) \in E$ è associato un *two-player game* con funzione potenziale

$$\Phi_e(x) = \begin{cases} -a, & \text{se } x_u = x_v = 1; \\ -b, & \text{se } x_u = x_v = 0; \\ 0, & \text{altrimenti} \end{cases} \tag{7.1}$$

con $a, b \geq 0$. Si assume che ogni agente preferisca coordinarsi con i suoi vicini piuttosto che giocare una strategia differente.
Si consideri ora la situazione in cui ogni agente debba scegliere una strategia e giocare tale strategia in ogni gioco in cui esso è coinvolto, cioè su ogni arco adiacente. Questa situazione modella un gioco con funzione potenziale

$$\Phi(x) = \sum_{e \in E} \Phi_e(x).$$

È noto che la *Logit Dynamics* per questo tipo di gioco converge alla distribuzione

$$\pi(x) = \frac{e^{\beta \Phi(x)}}{Z},$$

dove Z è il fattore di normalizzazione e β rappresenta il livello di razionalità. Tenendo a mente la definizione di *mean magnetic moment* fornita in 6.1

$$\mathcal{M} = \beta^{-1} \partial(lnZ)/\partial\beta,$$

il Teorema 6.1 che afferma *"Esiste un fpras per il mean magnetic moment $\mathcal{M} = \beta^{-1}\partial(lnZ)/\partial\beta$, dove Z è la partition function di un sistema di Ising ferromagnetico."*

nodi	archi	s	t	Tempo
10	50	5950	2	3min 48s
20	100	13218	2	6min 31s
40	250	27754	2	12min 46s
100	1000	71361	2	34min 15s
500	5000	362075	2	3h 24min 17s

Tabella 7.8: Simulazioni *mean magnetic moment*.

Sono stati apportati notevoli miglioramenti al metodo di approssimazione della funzione *odd(X)*, come mostrato in 6.1.1, che hanno portato alla realizzazione degli algoritmi presentati in 6.1.2.
La simulazione è stata realizzando avvalendosi dell'algoritmo 4, il quale a sua volta fa uso di 2. Sono stati scelti i parametri che hanno fornito i migliori risultati nei tempi d'esecuzione degli esperimenti precedenti, cioè $B = 20, \beta = 0.4$ ed $\epsilon = 0.1$. Il grafico seguente riporta i tempi di alcune esecuzioni, al variare del grafo di input. La Tabella 7.8 fornisce maggiori

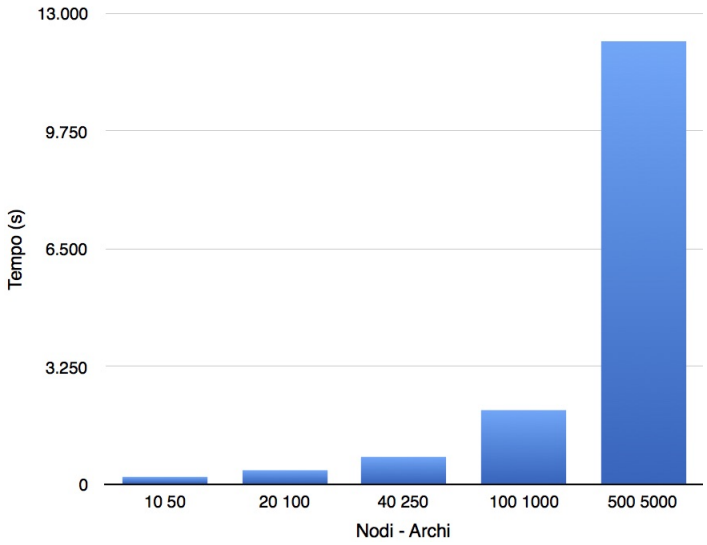


Figura 7.26: Algoritmo Partition2 $n = 40, m = 250$.

informazioni riguardo i dettagli di esecuzione dei test. Il lavoro [6] riporta nel Capitolo 6 i risultati del test effettuato per un simile scenario e la Ta-

nodi	archi	s	t	Tempo
10	50	62245	2	14h 20m 16s

Tabella 7.9: Simulazioni *mean magnetic moment*.

bella 7.9 mostra il tempo impiegato per eseguire un test su un grafo con 10 nodi e 50 archi. È evidente che tale implementazione sia impraticabile in situazioni reali, e allo stesso tempo è possibile apprezzare l'efficienza del lavoro proposto, che porta a termine lo stesso esperimento in circa 4 minuti rispetto alle 14 ore impiegate in precedenza. Inoltre, l'algoritmo che calcola il *mean magnetic moment* presentato in questa tesi presenta tempi ragionevoli anche su istanza di dimensioni discrete, come nel caso del grafo a 500 nodi e 5000 archi su cui impiega circa 3 ore e 30 minuti per calcolarne la magnetizzazione.

7.4 Test su dati reali

Il lavoro svolto ha reso possibile l'obiettivo preposto: testare l'algoritmo **Partition2** su dataset provenienti dal mondo reale.

7.4.1 Dataset Gnutella

Il dataset utilizzato è una delle network messe a disposizione dallo *Stanford Network Analysis Project* (SNAP) [46] e rappresenta una sequenza di 9 *snapshots* della rete di condivisione file *peer-to-peer* Gnutella a partire da Agosto 2002. I nodi rappresentano gli *host* nella topologia della rete Gnutella e gli archi rappresentano le connessioni tra di essi. In Tabella 7.10 vengono riportate informazioni dettagliate sul dataset.

Nodi	6301
Archi	20777
Nodi nella più grande WCC	6299 (1.000)
Archi nella più grande WCC	20776 (1.000)
Nodi nella più grande SCC	2068 (0.328)
Archi nella più grande SCC	9313 (0.448)
Coefficiente di average clustering	0.0109
Numero di triangoli	2383
Frazione di triangoli chiusi	0.006983
Diametro (*shortest path* più lungo)	9

Tabella 7.10: Statistiche del grafo Gnutella (SNAP).

L'algoritmo **Partition2** ha terminato la sua esecuzione in 59.129,02 secondi, cioè in circa 16 ore e 42 minuti.

7.4.2 Dataset Facebook

Il dataset utilizzato in quest'ultimo esperimento è anch'esso messo a disposizione dallo *Stanford Network Analysis Project* (SNAP) [46] e rappresenta uno *snapshot* di un sottografo della rete di amicizie di Facebook. Gli utenti sono rappresentati dai nodi e le relazioni di amicizia sono rappresentate dagli archi. Il grafo in questione è una vera e propria rete sociale e presenta tutte le caratteristiche tipiche di questo tipo di reti, come l'alto numero di triangoli ed un discreto coefficiente di *average clustering*. La Tabella 7.11 mostra ulteriori informazioni sul dataset.

L'algoritmo **Partition2** ha terminato la sua esecuzione in 684.060,44 se-

Nodi	4039
Archi	88234
Nodi nella più grande WCC	4039 (1.000)
Archi nella più grande WCC	88234 (1.000)
Nodi nella più grande SCC	4039 (1.000)
Archi nella più grande SCC	88234 (1.000)
Coefficiente di average clustering	0.6055
Numero di triangoli	1612010
Frazione di triangoli chiusi	0.2647
Diametro (*shortest path* più lungo)	8

Tabella 7.11: Statistiche del grafo Facebook (SNAP).

condi, cioè circa 7 giorni. Considerando le caratteristiche della rete, il tempo impiegato non è eccessivo.
Questo prova l'effettiva possibilità di poter essere utilizzato su reti del mondo reale.

Capitolo 8

Conclusioni e sviluppi futuri

In questo lavoro di tesi è stata proposta una versione migliorata dell'algoritmo per il calcolo della *partition function* Z di un sistema di Ising ferromagnetico arbitrario in 5, che fosse superiore nelle prestazioni rispetto a quello presentato nel lavoro [6].
I risultati ottenuti durante la fase di testing, analizzati nella sezione 7.2, in particolare dal confronto con la versione sviluppata in [6], rendono possibile verificare il netto miglioramento ottenuto nel *running time* dell'algoritmo.
Tali risultati sono stati il frutto di una attenta analisi dei Teoremi, Lemmi ed assunzioni (a volte approssimative) del lavoro di Jerrum e Sinclair [4], spesso seguita da una rivisitazione che puntasse alla correttezza ma soprattutto all'efficienza.
Le performance raggiunte consentono, ora, di utilizzare il lavoro sviluppato per calcolare la funzione di partizione Z in una particolare dinamica, e quindi di computare la probabilità stazionaria in tempo un *polinomiale* trattabile in casi concreti, caratteristica che fino ad ora non era possibile ottenere con gli algoritmi precedenti, i quali sebbene avessero complessità polinomiale non consentivano di trattare casi reali, a causa del *running time* molto sensibile anche ad un piccolo incremento nella taglia dell'input.
L'efficienza dell'algoritmo proposto deriva dal lavoro svolto nell'ottimizzazione dei valori s e t che rappresentano, rispettivamente, la taglia dell'insieme di configurazioni su cui eseguire il generatore del Teorema 4.2 ed il numero di ripetizioni dell'esperimento necessarie su cui calcolare la mediana. Il generatore del *subgraphs-world process*, descritto in 4.2, è stato migliorato notevolmente sia dal punto di vista dell'ottimizzazione del codice, come del resto tutto il codice implementato, ma soprattutto nella sua complessità computazionale: il precedente tempo d'esecuzione era limitato da

$O(m^2\mu^{-8}(log\,\delta^{-1}+m))$, pertanto era fortemente influenzato dal numero di archi del grafo di input; grazie all'integrazione del lavoro [5], come mostrato in 5.3, la complessità è stata portata a $O(2m^2\mu^{-4}(log\,\delta^{-1}+1))$, riducendo così di molto la dipendenza dal numero di archi.
Gli sviluppi futuri più interessanti riguardano il generatore. La direzione da prendere sarebbe quella di lavorare sulla catena di Markov da simulare ed in particolare un importante passo avanti prevederebbe la sua parallelizzazione, operando sulla matrice di transizione: tale miglioramento porterebbe un notevole incremento prestazionale, consentendo di effettuare esperimenti anche su reti sociali di elevata dimensione, così da poterne computare la magnetizzazione e ad esempio poter contribuire concretamente nel campo del *computational advertising*.

Bibliografia

[1] D. Easley and J. Kleinberg, *Networks, crowds, and markets: Reasoning about a highly connected world*. Cambridge University Press, 2010.

[2] L. E. Blume, "The statistical mechanics of strategic interaction," *Games and economic behavior*, vol. 5, no. 3, pp. 387–424, 1993.

[3] F. Martinelli, "Lectures on glauber dynamics for discrete spin models," in *Lectures on probability theory and statistics*, pp. 93–191, Springer, 1999.

[4] M. Jerrum and A. Sinclair, "Polynomial-time approximation algorithms for the ising model," *SIAM Journal on computing*, vol. 22, no. 5, pp. 1087–1116, 1993.

[5] V. Auletta, D. Ferraioli, F. Pasquale, P. Penna, and G. Persiano, "Convergence to equilibrium of logit dynamics for strategic games," in *Proceedings of the twenty-third annual ACM symposium on Parallelism in algorithms and architectures*, pp. 197–206, ACM, 2011.

[6] I. Rinaldi, "An efficient approximation algorithm for computing the gibbs measure," 2016.

[7] C. Alós-Ferrer and N. Netzer, "The logit-response dynamics," *Games and Economic Behavior*, vol. 68, no. 2, pp. 413–427, 2010.

[8] H. P. Young, "The diffusion of innovations in social networks," *The economy as an evolving complex system III: Current perspectives and future directions*, vol. 267, 2006.

[9] D. Ferraioli, "Logit dynamics for strategic games mixing time and metastability," 2012.

[10] V. Auletta, D. Ferraioli, F. Pasquale, and G. Persiano, "Metastability of logit dynamics for coordination games," in *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pp. 1006–1024, SIAM, 2012.

[11] S. J. P., "Social network analysis: A handbook (2nd edition). thousand oaks, ca: Sage publications.,"

[12] U. d. P. Dipartimento di Scienze Sociale, "`http://sna.dss.unipi.it/Analisi%20delle%20reti.html`,"

[13] T. Ferdinand, "Gemeinschaft und gesellschaft, leipzig: Fues's verlag. (translated, 1957 by charles price loomis as community and society, east lansing: Michigan state university press),"

[14] D. Emile, "De la division du travail social: étude sur l'organisation des sociétés supérieures, paris: F. alcan. (translated, 1964, by lewis a. coser as the division of labor in society, new york: Free press),"

[15] G. Simmel, "Soziologie, leipzig: Duncker & humblot.,"

[16] M. Bronislaw, "The family among the australian aborigines: A sociological study. london: University of london press.,"

[17] A. R. Radcliffe-Brown, "The social organization of australian tribes. sydney, australia: University of sydney oceania monographs, no.1.,"

[18] A. R. Radcliffe-Brown, "On social structure. journal of the royal anthropological institute 70: 1–12. doi:10.2307/2844197,"

[19] L.-S. Claude, "Les structures élémentaires de la parenté. paris: La haye, mouton et co. (translated, 1969 by j. h. bell, j. r. von sturmer, and r. needham, 1969, as the elementary structures of kinship, boston: Beacon press),"

[20] N. S. F., "The theory of social structure. london: Cohen and west,"

[21] P. Talcott, "The structure of social action: A study in social theory with special reference to a group of european writers. new york: The free press,"

[22] P. Talcott, "The social system. new york: The free press,"

[23] B. Peter, "Bureaucracy in modern society. new york: Random house, inc.,"

[24] B. Peter, "The american journal of sociology, (65)6: 545-556,"

[25] B. Peter, "Exchange and power in social life.,"

[26] B. Hogan, "`TheNetworkedIndividual:AProfileofBarryWellman`,"

[27] G. Mark, "Introduction for the french reader. sociologica 2: 1–8,"

[28] W. Barry, "Structural analysis: From method and metaphor to theory and substance. pp. 19-61 in b. wellman and s. d. berkowitz (eds.) social structures: A network approach, cambridge, uk: Cambridge university press.,"

[29] M. Nicholas, "Theories and theory groups in contemporary american sociology. new york: Harper and row, 1973,"

[30] T. Charles, "An urban world. boston: Little brown, 1974,"

[31] W. Barry, "Structural analysis: From method and metaphor to theory and substance. pp. 19-61 in social structures: A network approach, edited by barry wellman and s. d. berkowitz. cambridge: Cambridge university press,"

[32] B. A. Cipra, "An introduction to the ising model," *American Mathematical Monthly*, vol. 94, no. 10, pp. 937–959, 1987.

[33] C. M. Grinstead and J. L. Snell, *Introduction to probability*. American Mathematical Soc., 2012.

[34] D. A. Levin, Y. Peres, and E. L. Wilmer, *Markov chains and mixing times*. American Mathematical Soc., 2009.

[35] D. P. Williamson and D. B. Shmoys, *The design of approximation algorithms*. Cambridge university press, 2011.

[36] R. D. McKelvey and T. R. Palfrey, "Quantal response equilibria for normal form games," 1993.

[37] T. H. Ho, C. F. Camerer, and J.-K. Chong, "Self-tuning experience weighted attraction learning in games," *Journal of Economic Theory*, vol. 133, no. 1, pp. 177–198, 2007.

[38] C. Gelatt, M. Vecchi, *et al.*, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[39] G. F. Newell and E. W. Montroll, "On the theory of the ising model of ferromagnetism," *Reviews of Modern Physics*, vol. 25, no. 2, p. 353, 1953.

[40] A. Sinclair, *Algorithms for random generation and counting: a Markov chain approach*. Springer Science & Business Media, 2012.

[41] A. Sinclair and M. Jerrum, "Approximate counting, uniform generation and rapidly mixing markov chains," *Information and Computation*, vol. 82, no. 1, pp. 93–133, 1989.

[42] E. K. Donald, "The art of computer programming," *Generating All Tuples and Permutations*, vol. 4, 1999.

[43] "Numpy." `http://www.numpy.org/`, 2006.

[44] "joblib." `https://pythonhosted.org/joblib/`, 2010.

[45] I. de Jon, "Python performance tips." `https://wiki.python.org/moin/PythonSpeed/PerformanceTips`, 2004.

[46] J. Leskovec and A. Krevl, "SNAP Datasets: Stanford large network dataset collection." `http://snap.stanford.edu/data`, June 2014.