



**UNIVERSITÀ DEGLI STUDI DI ROMA  
TOR VERGATA**

FACOLTÀ DI SCIENZE MATEMATICHE, FISICHE E  
NATURALI

CORSO DI LAUREA IN INFORMATICA

A.A. 2019/2020

**Tesi di Laurea**

Analisi dell'impatto di “bias” e topologia sui tempi di  
assorbimento per Dinamiche di Opinioni

**RELATORE**

Prof. Francesco Pasquale

**CANDIDATO**

Saverio Biancone

**CORRELATRICE**

Dott.ssa Sara Rizzo

# Indice

<b>1</b>	<b>Introduzione</b>	<b>1</b>
<b>2</b>	<b>La Dinamica delle Opinioni</b>	<b>2</b>
2.1	Lo studio . . . . .	2
2.2	Lo stato dell'arte . . . . .	3
<b>3</b>	<b>Studi svolti e Analisi dei Risultati</b>	<b>5</b>
3.1	Cenni al software di simulazione . . . . .	5
3.2	Analisi di Clique, Ipercubo e Ciclo . . . . .	7
3.3	Analisi del Modello di Erdös Rényi . . . . .	10
<b>4</b>	<b>Strumenti di Analisi</b>	<b>14</b>
4.1	Opinion Dynamic Simulator . . . . .	14
<b>5</b>	<b>Conclusioni e Sviluppi Futuri</b>	<b>19</b>
5.1	Conclusioni . . . . .	19
5.2	Sviluppi Futuri . . . . .	20

# Capitolo 1

## Introduzione

Questo lavoro di tesi verte sullo studio e sull'analisi di un particolare modello di Dinamica delle Opinioni in cui in un sistema binario gli agenti esprimono una preferenza verso un'opinione dominante.

Partendo dai risultati ottenuti dagli autori dell'articolo *Biased Opinion Dynamics* [1], ho cercato di trovare risposte ad alcune delle domande rimaste insolute e proposte nelle conclusioni dell'articolo stesso.

In particolare, il mio lavoro di tesi tenta di individuare tra alcune topologie caratteristiche e la dinamica di aggiornamento *Majority-Dynamics* quando il bias verso l'opinione dominante é inferiore a  $\frac{1}{2}$ .

Per far questo é stato sviluppato un software in Python in grado di simulare i processi descritti così da poter effettuare un'analisi empirica attraverso i dati ricavati.

Le topologie maggiormente trattate in questo lavoro sono state *Ipercubo* e *Modello di Erdös Rényi G(n,p)*[2] seppure spesso sia stato utile poter confrontare i dati con quelli ottenuti da *Clique* e *Ciclo*, i quali comportamenti sono stati già ampiamente osservati nell'articolo [1].

# Capitolo 2

## La Dinamica delle Opinioni

La dinamica delle opinioni ha come obiettivo quello di descrivere l'evoluzione delle opinioni all'interno di una rete sociale formata da individui interagenti.

Numerosi sono stati gli studi effettuati su modelli diversificati, contraddistinti soprattutto dalle dinamiche di interazione e il numero delle opinioni prese in esame, che hanno trovato come campo di applicazione discipline che vanno dalle Scienze Sociali alla Fisica e alla Biologia.

### 2.1 Lo studio

Il modello preso in esame descrive un sistema binario che prevede l'esistenza di due opinioni adottabili, indicate d'ora in poi come opinione 0 e opinione 1. Supponiamo inoltre che gli agenti del sistema esprimano una preferenza, nominata d'ora in poi come *bias*, verso una delle due opinioni, che definiremo dominante<sup>1</sup>. Senza perdita di generalità, d'ora in poi assumeremo 1 come tale.

Il sistema evolve in passi. Inizialmente ogni individuo condivide l'opinione 0.

Ad ogni passo, un individuo scelto con probabilità uniforme adotta l'opinione 1 (dominante) con probabilità  $\alpha$ , mentre con probabilità  $1-\alpha$  adotta una delle due opinioni

---

<sup>1</sup>La definizione di opinione dominante dipende dal contesto ed esula dagli obiettivi di questo lavoro.

possibili attraverso una delle dinamiche prestabilite.

Le dinamiche di aggiornamento dell'opinione prese in considerazione sono:

- *Voter Model*, nel quale l'individuo adotterá l'opinione di uno dei suoi vicini scelto con probabilitá uniforme.
- *Majority-Dynamics*, nel quale l'individuo adotterá l'opinione piú diffusa tra i suoi vicini. In caso di paritá verrá scelta una delle due opinioni con probabilitá uniforme.

L'obiettivo é quello di analizzare il numero di passi, in valore atteso, necessari affinché si raggiunga il consenso verso l'opinione dominante. É facile notare come, raggiunto questo stato, il sistema non possa evolvere piú. Definiremo perció tale stato *Stato di Assorbimento*.

Tale analisi é volta a rivelare, laddove esistesse, il legame che intercorre tra la topologia della rete e il numero di passi necessari a raggiungere lo stato di assorbimento, sotto una specifica dinamica. La dinamica di aggiornamento dell'opinione principalmente trattata in questo lavoro é Majority-Dynamics.

## 2.2 Lo stato dell'arte

Nel corso degli anni sono stati molteplici gli studi riguardanti la Dinamica delle Opinioni, in particolare per quanto concerne l'impatto che differenti topologie hanno sul raggiungimento del consenso.

Questo lavoro di tesi si pone come integrazione dell'articolo *Biased Opinion Dynamics: When the Devil is in the Details* [1], con cui condivide il modello, ed ha come fine ultimo quello di rispondere ad alcuni quesiti lasciati insoluti, attraverso un'analisi dei dati ottenuti simulando il processo descritto nel paragrafo precedente. Il

Voter-Model ricopre un ruolo secondario in questo lavoro in quanto, nell'articolo [1], gli autori hanno dimostrato che il tempo di assorbimento nel modello preso in esame é di  $O(\frac{1}{\alpha}n \log n)$  passi con alta probabilitá, indipendentemente dalla topologia della rete.

Differenti invece sono i legami, come dimostrato, che intercorrono tra il bias, la topologia e il tempo di assorbimento quando la dinamica di aggiornamento é Majority-Dynamics.

Per vaolori di bias superiori a  $\frac{1}{2}$  il tempo di assorbimento é pari a  $O(\frac{1}{\alpha}n \log n)$  indipendentemente dalla topologia delle reti. Appena tale valore scende al di sotto di  $\frac{1}{2}$  però il tempo necessario a raggiungere il consenso diventa esponenziale nel grado minimo.

Per quanto riguarda il Ciclo invece, il tempo di assorbimento viene raggiunto in  $O(\frac{1}{\alpha}n \log n)$  indipendentemente dal valore di  $\alpha$ .

# Capitolo 3

## Studi svolti e Analisi dei Risultati

Basandomi sui risultati descritti nell'articolo in esame [1] e riportati nel precedente capitolo, ho tentato di rispondere ad alcune delle domande rimaste insolute.

La mia attenzione é stata rivolta in particolare al tempo di assorbimento che si ottiene adottando Majority-Dynamics su alcune topologie quando il bias é inferiore a  $\frac{1}{2}$ .

Le topologie analizzate sono state:

- Ipercubo
- Clique
- Ciclo
- Modello di Erdös Rényi  $G(n,p)$  [2]

### 3.1 Cenni al software di simulazione

Per ottenere e, successivamente, analizzare i tempi di assorbimento, ho sviluppato un software in grado di simulare i processi descritti nel modello preso in esame. Di seguito é riportato lo pseudocodice di alcune delle funzioni principali utilizzate.

---

**Algorithm 1:** absorptionStateReached(*graph*: GraphTool.Graph)

---

```

for v ∈ graph.vertices do
    if v.opinion == 0 then
        return False;
return True;
```

---



---

**Algorithm 2:** runSimulationOn(*config*: SimulationConfigurator)

---

```

graph ← config.graph;
bias ← config.bias;
updateRule ← config.opinionUpdateRule;
rounds ← 0;
while ¬ absorptionStateReached(graph) do
    v ← random.choice(graph.vertices);
    if random(0,1) ≤ bias then
        | v.opinion ← 1;
    else
        | v.opinion ← updateRule.run(graph,v);
    rounds ← rounds + 1;
simulationResult ← new SimulationResult(config, rounds);
return simulationResult;
```

---

Notare come *config.opinionUpdateRule* sia un oggetto ottenuto tramite un’interfaccia che permette di implementare la dinamica di aggiornamento preferita eseguendo un *override* del metodo *run*. Di seguito l’implementazione in pseudocodice di tale metodo per Majority-Dynamics.

---

**Algorithm 3:** MajorityDynamics.run(*graph*: GraphTool.Graph, *v*: GraphTool.Vertex)

---

```

neighbors ← v.neighbors();
if  $\neg$  neighbors then
    return v.opinion;
counter0 ← 0;
counter1 ← 0;
for v  $\in$  neighbors do
    if v.opinion == 1 then
        counter1 ← counter1+1;
    if v.opinion == 0 then
        counter0 ← counter0+1;
if counter0 > counter1 then
    return 0;
if counter1 > counter0 then
    return 1;
return random.choice([0,1]);

```

---

## 3.2 Analisi di Clique, Ipercubo e Ciclo

I primi test eseguiti sono serviti a verificare l'accuratezza del software e correggerne eventuali errori. Ogni test é composto da 100 simulazioni, configurate con Majority-Dynamics e bias pari a  $\frac{1}{2}$ .

I test sono stati eseguiti su Ipercubi, Cliques e Cicli, in dimensioni che vanno da  $2^5$  fino a  $2^{12}$  vertici.

Dimensione	Misura	Ipercubo	Clique	Ciclo
32	Media	180,56	190,69	218,69
	Deviazione Standard	44,52	48,92	67,11
64	Media	390,94	420,62	544,16
	Deviazione Standard	81,66	99,10	184,28
128	Media	860,03	1.018,44	1.287,64
	Deviazione Standard	176,04	230,58	327,04
256	Media	1.922,38	2.241,15	2.912,32
	Deviazione Standard	371,70	474,27	773,10
512	Media	4.206,88	5.095,36	6.402,40
	Deviazione Standard	586,74	859,25	1.446,38
1024	Media	9.178,07	11.190,15	14.722,84
	Deviazione Standard	1.335,60	1.560,70	2.879,27
2048	Media	19.940,08	24.611,64	30.330,28
	Deviazione Standard	2.768,90	3.552,19	5.663,44
4096	Media	42.844,66	56.082,08	69.814,52
	Deviazione Standard	5.783,57	8.021,92	12.302,78

Figura 3.1: Test con  $\alpha = \frac{1}{2}$ 

I risultati ottenuti in figura 3.1 sono perfettamente in linea con quanto dimostrato teoricamente in *Biased Opinion Dynamics* [1]. Per quanto riguarda il Ciclo, il tempo di assorbimento è stato pari a  $O(\frac{1}{\alpha}n \log n)$ .

Riguardo Clique e Ipercubo, per i quali il grado minimo è pari a  $\Omega(\log n)$ , il tempo di assorbimento con bias pari a  $\frac{1}{2}$  si è rivelato essere  $O(n \log n)$ , anche questa volta confermando quanto evidenziato nell'articolo [1].

Una volta appurato che il software simulasse correttamente i processi descritti, ottenendo risultati conformi a quanto atteso, sono stati eseguiti test metodologicamente analoghi ai precedenti, adottando però un bias verso l'opinione dominante pari a  $\frac{1}{4}$ , di fatto dimezzandolo.

Dimensione	Misura	Ipercubo	Clique	Ciclo
32	Media	308,46	4.341,05	398,12
	Deviazione Standard	93,30	4.574,83	167,94
64	Media	634,29	X	968,39
	Deviazione Standard	138,56	X	291,53
128	Media	1.557,23	X	2.375,14
	Deviazione Standard	297,50	X	669,43
256	Media	3.215,51	X	5.424,30
	Deviazione Standard	452,91	X	1.250,38
512	Media	7.596,25	X	11.750,23
	Deviazione Standard	1.065,62	X	2.170,64
1024	Media	15.453,14	X	26.588,76
	Deviazione Standard	1.502,07	X	4.201,82
2048	Media	37.837,60	X	60.041,30
	Deviazione Standard	1.467,85	X	11.061,66
4096	Media	76.854,10	X	123.238,60
	Deviazione Standard	6.602,98	X	13.443,44

Figura 3.2: Test con  $\alpha = \frac{1}{4}$ 

In questo caso i dati in figura 3.2 descrivono un comportamento piú articolato per una delle topologie esaminate.

Per quanto riguarda il Ciclo e la Clique, sono stati ancora una volta confermati i risultati attesi. Lo stato di assorbimento per il Ciclo é stato nuovamente raggiunto in  $O(\frac{1}{\alpha}n \log n)$  passi in valore atteso. Nello specifico, dimezzando il valore del bias, i tempi di assorbimento per tale topologia sono raddoppiati, suggerendo una proporzionalitá inversa tra il tempo di assorbimento e il valore del bias.

Nel teorema 4.2 dell’articolo [1] viene dimostrato come il tempo di assorbimento diventi esponenziale nel grado minimo  $\Delta$  per valori di  $\alpha \leq \frac{1}{2}$ .

$$\mathbf{E}[\tau] \geq \frac{e^{\frac{\epsilon^2}{6}\Delta}}{6n}, \quad 0 < \epsilon < 1 \quad (3.2.1)$$

Questo motiva l’impossibilitá di ultimare i test per la Clique, per la quale i tempi di assorbimento sono diventati esponenziali in  $n$ , di fatto rendendo impossibile eseguire simulazioni su dimensioni superiori a  $2^5$ .

L’iper cubo però é una topologia regolare in cui ogni vertice ha grado  $\log n$ . Il bound

ottenuto (3.2.1) risulta perciò poco *tight*, non rivelando di fatto il reale comportamento assunto da tale topologia.

Dai risultati dei test si evince come l’Ipercubo abbia un comportamento assimilabile a quello del Ciclo, raggiungendo lo stato di assorbimento in  $O(n \log n)$  passi.

### 3.3 Analisi del Modello di Erdös Rényi

Questa sezione è dedicata all’analisi dei tempi di assorbimento per topologie non regolari, cioè grafi per i quali i vertici non condividono lo stesso grado.

Tali topologie sono state generate scrivendo un algoritmo aleatorio basato sul Modello di Erdös Rényi  $G(n,p)$  [2].

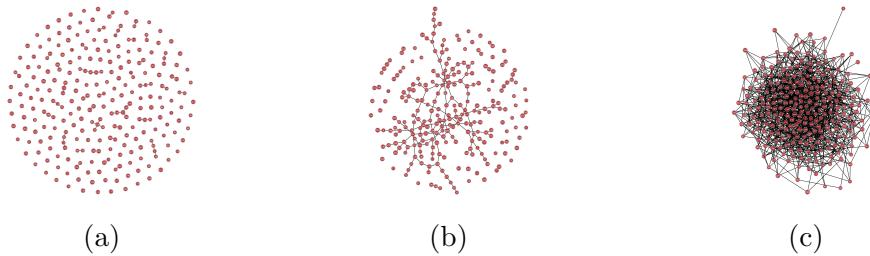
In tale modello, dato un grafo  $G(V,E)$  con  $|V| = n$ ,

$$\forall u, v \in V, P((u, v) \in E) = p. \quad (3.3.1)$$

I test eseguiti sono composti da 100 simulazioni, configurate con Majority-Dynamics e bias pari a  $\frac{1}{4}$ .

Le topologie prese in considerazione hanno dimensioni che vanno da  $2^5$  fino a  $2^{12}$  vertici e sono state generate con il modello  $G(n,p)$  descritto precedentemente, utilizzando i seguenti valori di  $p$ :

- $p = \frac{1-\epsilon}{n}$  con  $\epsilon = \frac{1}{2}$ , per il quale il grafo risulta principalmente sparso, con alta probabilità (in figura 3.4 indicato come ER\_SP).
- $p = \frac{1+\epsilon}{n}$  con  $\epsilon = \frac{1}{2}$ , per il quale il grafo presenta una *giant component* circondata da componenti più piccole di dimensione  $O(\log n)$ , con alta probabilità (in figura 3.4 indicato come ER\_GC).
- $p = \frac{\log n}{n}$ , per il quale il grafo risulta connesso e discretamente denso, con alta probabilità (in figura 3.4 indicato come ER\_DN).

Figura 3.3: Esempi ottenuti utilizzando le probabilitá descritte.  $n=256$ ,  $\epsilon=\frac{1}{2}$ 

Dimensione	Misura	ER_SP	ER_GC	ER_DN
32	Media	491,98	446,45	303,02
	Deviazione Standard	149,96	157,77	94,38
	Average Degree	0,37	1,18	4,38
64	Media	1.133,45	915,82	701,84
	Deviazione Standard	298,17	248,28	190,94
	Average Degree	0,62	1,75	6,03
128	Media	2.733,40	2.496,89	1.527,96
	Deviazione Standard	675,38	729,95	333,61
	Average Degree	0,51	1,31	6,78
256	Media	5.919,36	5.232,99	3.345,05
	Deviazione Standard	1.177,86	1.213,43	542,07
	Average Degree	0,49	1,46	7,75
512	Media	13.087,44	12.062,05	7.612,35
	Deviazione Standard	2.159,50	2.533,60	1.104,02
	Average Degree	0,61	1,47	8,90
1024	Media	29.600,63	26.097,06	17.220,86
	Deviazione Standard	4.629,69	4.311,23	2.157,82
	Average Degree	0,48	1,43	10,05
2048	Media	64.852,23	57.229,10	40.265,12
	Deviazione Standard	10.782,76	10.898,29	4.230,82
	Average Degree	0,51	1,51	11,09
4096	Media	141.961,70	136.069,30	100.124,80
	Deviazione Standard	18.411,53	22.333,29	10.659,06
	Average Degree	0,47	1,48	11,89

Figura 3.4: Test Erdös Rényi con  $\alpha = \frac{1}{4}$ 

I risultati dei test in figura 3.4 descrivono un andamento meno prevedibile di quanto ci si potesse aspettare.

Per valori di  $p$  che si allontanano dallo 0 il tempo di assorbimento inizia a diminuire, mentre gli studi compiuti sulla Clique [1] evidenziano una crescita esponenziale nel grado minimo quando  $p$  tende a 1. La curva che si ottiene perciò fissando  $n$  e facendo

variare  $p$  non risulta essere monotona crescente, in contrasto con quanto atteso.

Per indagare tale comportamento in modo piú approfondito, ho eseguito ulteriori test con la seguente configurazione:

- $n = 512$  e  $n = 1024$
- $\alpha = \frac{1}{4}$
- $p$  assume valori in  $[\frac{1-\epsilon}{n}, \frac{(1+\epsilon)\log n}{n}]$  t.c.  $p_{i+1} - p_i = 0.001$

Di seguito sono mostrati due grafici ottenuti eseguendo i test sopracitati.

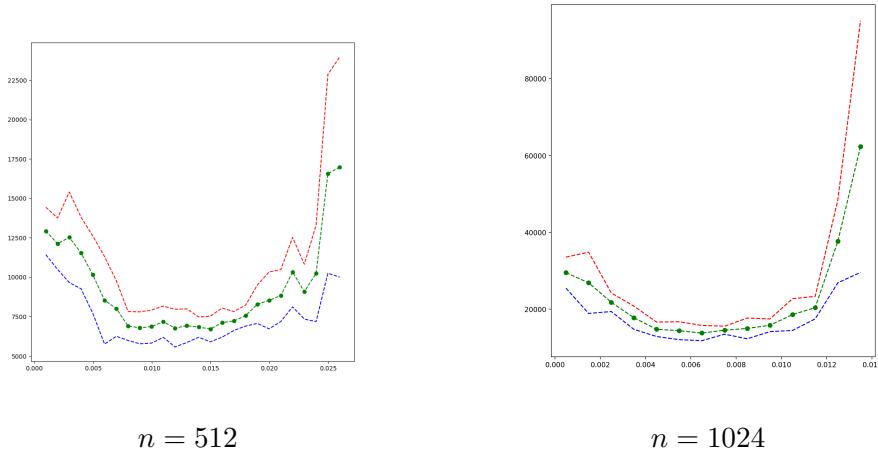


Figura 3.5: Sulle ascisse sono rappresentati i valori di  $p$  mentre sulle ordinate i tempi di assorbimento. La curva in verde rappresenta il valore medio ricavato dai test, mentre le curve in blu e rosso descrivono i bound ottenuti rispettivamente sottraendo e sommando la deviazione standard al valore medio.

Dai grafici in figura 3.5 si evince come il minimo della funzione ottenuta non si trovi in  $p = 0$ , come ci si poteva aspettare, per il quale il tempo di assorbimento é pari a  $O(\frac{1}{\alpha}n \log n)$ . Questo implica l'esistenza di un valore di  $p \geq 0$  per il quale il processo raggiunge lo stato di assorbimento in  $o(\frac{1}{\alpha}n \log n)$  rounds con alta probabilitá.

Dai dati ricavati nei diversi test eseguiti sugli Erdös Rényi é possibile osservare che:

- I valori ottenuti dalla prima serie di test descrivono un andamento decrescente del tempo di assorbimento almeno fino a  $\frac{\log n}{n}$ .
- La seconda serie di test descrive una crescita del tempo di assorbimento partendo da un punto di minimo non esplicitato, e prosegue fino al limite per  $p$  fissato per questi test, che corrisponde a  $\frac{3 \log n}{2n}$ .

Tutto ciò suggerisce come il punto di minimo  $p_{min}$  si trovi in  $[\frac{\log n}{n}, \frac{3 \log n}{2n}]$ .

Sapendo che per il modello  $G(n, p)$  il numero di archi in valore atteso è  $\binom{n}{2}p$ , se ne ricava che lo stato di assorbimento viene raggiunto più velocemente nei grafi che hanno un numero di archi  $E$ :

$$\frac{(n-1)\log n}{2} \leq |E| \leq 1 \frac{3(n-1)\log n}{4} \quad (3.3.2)$$

# Capitolo 4

## Strumenti di Analisi

### 4.1 Opinion Dynamic Simulator

Il software sviluppato fornisce degli strumenti utili a riprodurre le dinamiche descritte al fine di indagare legami inespressi che intercorrono tra alcune topologie e valori di bias.

Il linguaggio principale scelto per lo sviluppo del software é Python. Le motivazioni dietro tale scelta risiedono non soltanto nella grande diffusione e nel grande supporto di cui questo linguaggio gode, ma anche dalla presenza di numerosi moduli che permettono di rappresentare ed interagire al meglio con i grafi, strutture attraverso le quali viene rappresentata la rete sociale. Tra i vari moduli disponibili é stato scelto Graph-Tool<sup>1</sup>. Tra i punti di forza di questo modulo si annoverano una precisa modellazione di tali strutture, con possibilità di aggiungere proprietà personalizzate su vertici e archi nonché numerose funzioni di generazione e rappresentazione di alcune topologie tipiche dei grafi, quali Paths, Cicli e Cliques. Il vero punto di forza di Graph-Tool é però la sua velocit, ottenuta grazie alla possibilità di eseguire molte di queste funzioni non in Python, bens in C, linguaggio di livello pi basso e perci pi

---

<sup>1</sup><https://graph-tool.skewed.de>

rapido nell'esecuzione di alcune istruzioni fondamentali.

Alcune topologie non presenti nativamente nel modulo, sono state personalmente riscritte utilizzando comunque le strutture di base fornite. È questo il caso dell'Ipercubo e del Modello di Erdös Rényi [2], di cui è riportato lo pseudocodice.

---

**Algorithm 4:** generateHypercube( $d$ : int)

---

```

n ←  $2^d$ ;
graph ← new GraphTool.Graph(vertices=n);
for  $u \in graph.vertices$  do
    binaryIndex ← binary(u.index);
    for bit ∈ binaryIndex do
        binaryIndex ← binaryIndex.flip(bit);
        v ← graph.getVertexByIndex(int(binaryIndex));
        graph.addEdge(u,v);
return graph;

```

---



---

**Algorithm 5:** generateErdös-Rényi( $n$ : int,  $p$ : float)

---

```

if  $p \geq 1$  then
    return completeGraph(n);
graph ← new GraphTool.Graph(vertices=n);
if  $p \leq 0$  then
    return graph;
for  $i \in 0 \dots n-1$  do
    for  $j \in i+1 \dots n-1$  do
        u ← graph.getVertexByIndex(i);
        v ← graph.getVertexByIndex(j);
        graph.addEdge(u,v) with probability  $p$ ;
return graph;

```

---

Partendo da tale base, è stato sviluppato il processo di simulazione vero e proprio.

Tale processo necessita come input, oltre al grafo, di un oggetto (d'ora in poi definito come *configurator*) contenente i parametri necessari alla configurazione della simula-

zione, come ad esempio la dinamica adottata e il bias verso l'opinione dominante.

Al termine del processo di simulazione verrá generato un oggetto in grado di salvare diverse informazioni, quali:

- Un file .XML contenente la serializzazione del grafo, così che possa essere facilmente ricostruito e riutilizzato.
- Un file .XML contenente le informazioni di configurazione e il risultato (numero di step) della simulazione.
- Una rappresentazione grafica del grafo in formato .PNG
- Una cartella contenente dei file .PNG che mostrano l'andamento evolutivo della simulazione.

Al fine di ottenere dei dati quanto piú possibile affidabili e limitare l'aleatorietá di tali processi, il software dispone inoltre di un modulo volto all'esecuzione di test. In questo contesto per test si intende l'esecuzione ripetuta di una simulazione, mantenendo invariati i parametri di configurazione e la struttura del grafo. Cosí come le simulazioni, anche i test necessitano di una configurazione che include, oltre al numero di iterazioni da effettuare, l'oggetto configurator riferito alle simulazioni eseguite nel test. Ponendo come obiettivo quello di velocizzare l'esecuzione di un test e ridurre la dimensione dell'output, soprattutto per grafi con un numero elevato di vertici e archi, l'oggetto generato a seguito dell'esecuzione di un test permette il salvataggio di un ristretto numero di informazioni, quali:

- Un file .XML contenente la serializzazione del grafo, così che possa essere facilmente ricostruito e riutilizzato.

- Un file .XML contenente le informazioni di configurazione del test, i risultati di ogni singola simulazione, la media di tali valori e la relativa deviazione standard.
- Una rappresentazione grafica del grafo in formato .PNG.

Considerando la durata media di un test composto da 100 simulazioni, eseguito su un grafo particolarmente impegnativo, ad esempio con  $2^{12}$  vertici, ho scelto di inserire un modulo che permetesse di riunificare molteplici test pre-eseguiti, utilizzando come dataset l'insieme delle simulazioni eseguite e ricalcolando correttamente la media e la deviazione standard.

In questo modo è possibile suddividere un test composto da 100 simulazioni in 10 test indipendenti, da 10 simulazioni ognuno, permettendo così di ridurre sensibilmente lo stress computazionale inflitto alla macchina sul quale viene eseguito.

La scelta del formato .XML è dettata dalla necessità di avere quante più informazioni possibili salvate in maniera strutturata e coerente, grazie anche alla possibilità, fornita nativamente da Python, di formattare automaticamente files in tale estensione.

Un ulteriore vantaggio dato dalla scelta del formato .XML è la possibilità di re-importare un grafo (tramite un metodo fornito dalla libreria graph-tool), partendo dal file ottenuto dalla serializzazione dello stesso. Questo risulta molto efficiente qualora si volessero eseguire ulteriori test/simulazioni a partire da un grafo precedentemente generato e difficilmente ricostruibile (nel caso di topologie ottenute tramite processi aleatori come nel Modello di Erdős Rényi) [2]. Le rappresentazioni grafiche invece sono state salvate in .PNG, un buon compromesso tra qualità dell'output e dimensione dei file stessi. Altre opzioni facilmente implementabili sono .SVG e .PDF.

Il *core* del software è rappresentato dall'algoritmo di simulazione che esegue il processo su un grafo fornito in input, aggiornando l'opinione di ogni nodo attraverso una *vertex*

*property* definita appositamente.

L'algoritmo di simulazione supporta attualmente *Voter Model* e *Majority-Dynamics* ma il codice é stato modellato in modo da poter facilmente implementare, mediante un'interfaccia<sup>2</sup>, ulteriori dinamiche di aggiornamento.

---

<sup>2</sup>Python non fornisce nativamente le interfacce, perció é stata simulata utilizzando classi e metodi astratti.

# Capitolo 5

## Conclusioni e Sviluppi Futuri

### 5.1 Conclusioni

Questo lavoro di tesi é incentrato sull’analisi del comportamento che alcune particolari topologie assumono in merito al raggiungimento del consenso nell’ambito della dinamica di opinioni. Si pone come uno studio integrativo dell’articolo *Biased Opinion Dynamics* [1] per il quale cerca di fornire risposte ad alcune domande rimaste insolute.

Per far questo é stato sviluppato un software che permettesse di simulare le dinamiche descritte e analizzarne empiricamente i dati prodotti. I primi test hanno avuto come soggetto di analisi l’iper cubo per il quale hanno evidenziato un tempo di assorbimento che é risultato essere, nonostante il grado minimo pari a  $\Omega(\log n)$ , asintoticamente assimilabile a quello del Ciclo.

Successivamente sono stati analizzati i tempi di assorbimento per grafi non regolari, generati tramite il modello  $G(n, p)$  di Erdös Rényi [2].

In questo caso l’analisi dei dati suggerisce un andamento non monotono della funzione che lega il valore di  $p$  al tempo di assorbimento, evidenziando un punto di minimo situato nell’intervallo  $[\frac{\log n}{n}, \frac{3\log n}{2n}]$ .

Questo ha permesso di identificare il numero di archi necessari affinché il processo

termini in un numero di passi inferiore a  $\frac{1}{\alpha}n \log n$ .

## 5.2 Sviluppi Futuri

Il software di simulazione é stato modellato e realizzato affinché fosse facilmente espandibile in futuro. In questo senso risulterebbe interessante indagare analiticamente il comportamento di altre topologie, valori di bias e dinamiche di aggiornamento rimaste inesplorate.

Un'altra direzione percorribile potrebbe essere quella di implementare una struttura che permetta di astrarre maggiormente i processi di simulazioni così da poter fornire topologie generate da altre librerie e non necessariamente da Graph-Tool.

Seppure quest'ultima rappresenti un ottimo compromesso tra versatilitá, funzionalitá e prestazioni, é innegabile come il suo maggior punto di forza rappresenti al contempo una debolezza per quanto riguarda la difficoltá di importazione all'interno dell'ambiente di sviluppo, soprattutto per sistemi non *Unix-Like*.

# Bibliografia

- [1] Aris Anagnostopoulos, Luca Becchetti, Emilio Cruciani, Francesco Pasquale, and Sara Rizzo. Biased opinion dynamics: When the devil is in the details. *CoRR*, abs/2008.13589, 2020.
- [2] P Erdős and A Rényi. On random graphs i. *Publicationes Mathematicae Debrecen*, 6:290–297, 1959.