UNIVERSITÀ
DI TRENTO

Dipartimento di Ingegneria e Scienza dell'Informazione

Corso di Laurea in
Informatica

Elaborato finale

Influential spreaders in multilayer networks

Supervisore

Montresor Alberto

Laureando

Masina Gabriele

Anno accademico 2019/2020

Ringraziamenti

...thanks to...

Indice

Sommario

Diversi sistemi reali possono essere modellati tramite una rete in cui le entità, sono legate se tra loro c'è una relazione. In particolare in molti casi le relazioni tra queste entità possono essere di varia natura. Questo tipo di sistema può essere rappresentato usando una *multilayer network*.

Lo scopo di questa tesi è l'individuazione degli *influential spreaders* in una *multilayer network*, ovvero di quei nodi nella rete che durante un processo di diffusione permettono di infettare la porzione di rete maggiore. Questi nodi sono particolarmente utili in quanto possono essere sfruttati per massimizzare la diffusione di un'informazione all'interno della rete o, al contrario, isolati se si vuole che l'informazione resti il più possibile confinata.

Tali nodi sono facilmente identificabili simulando i processi di diffusione, che però risultano computazionalmente molto costosi e quindi non applicabili a reti molto grandi. Pertanto l'obiettivo è stato quello di individuarli analizzando la topologia della rete. Una fase di ricerca iniziale ha permesso di raccogliere, studiare e successivamente implementare diversi algoritmi di centralità già utilizzati per risolvere questo problema sia in grafi che in *multilayer networks*.

Questi algoritmi sono stati applicati a diversi tipi di reti *multilayer*: reti *multiplex* estratte a partire da dataset biologici e sociali reali e reti *multilayer* generate casualmente a partire da grafi derivati da applicazioni peer-to-peer.

I valori di centralità dei nodi restituiti da ogni algoritmo sono stati confrontati con quelli reali, ottenuti simulando dei processi di diffusione nelle varie reti. In questo modo è stato possibile assegnare ad ogni algoritmo un punteggio ottenuto su ciascuna rete e individuare così quali algoritmi possono essere utilizzati per trovare gli *influential spreaders* in una *multilayer network*. Due due gli algoritmi testati hanno ottenuto buone performance su tutte le reti analizzate, mentre le prestazioni degli altri sono state scarse o altalenanti.

1 Introduzione

Le reti permettono di rappresentare una grande quantità di sistemi in cui diverse entità sono legate tra loro mediante delle relazioni. Esempi di questo tipo sono le reti di interazione sociale, in cui le persone sono legate da relazioni di amicizia, di collaborazione o di contatto sui social networks, le reti dei trasporti, come quelle ferroviaria e aerea che collegano diverse città, e le reti biologiche, come quella alimentare o quella delle interazioni tra molecole all'interno di un organismo.

Utilizzando questi modelli è particolarmente utile lo studio dei processi di diffusione di un'informazione in una rete per comprendere le dinamiche di diversi scenari. Tra questi, la diffusione di un virus in una società, di una notizia in una rete di contatti, una fake news in un social network e l'accumulo di un ritardo nelle stazioni di una rete dei trasporti.

In questi processi, inizialmente un nodo (così sono chiamate le entità all'interno di una rete) è infetto; in ogni instante un nodo infetto può contagiare con una certa probabilità i nodi con cui è in relazione rendendoli a loro volta infetti. Di particolare interesse è l'individuazione dei cosiddetti *influential spreaders*, ovvero dei nodi che, se infettati, permettono di 'contagiare' una grossa parte della rete. Tali nodi, infatti, possono essere sfruttati per garantire una efficace diffusione di un'informazione nella rete o, al contrario, per evitarne la propagazione. Ad esempio, per diffondere una notizia importante in un social network sarà opportuno che questa venga pubblicata da un account molto seguito, mentre per fermare un'epidemia può essere utile isolare i luoghi o le persone che hanno un ruolo più centrale nella società.

1.1 *Influential spreaders* in un grafo

Generalmente, una rete viene modellata tramite un grafo, ossia una struttura in cui i nodi sono connessi tramite archi se tra loro esiste una relazione.

Il modo più affidabile per l'individuazione degli *influential spreaders* in un grafo consiste nel simulare il processo di diffusione, detto anche *spreading process*, partendo da ogni nodo e classificare come nodi più influenti quelli da cui si riesce ad infettare la porzione di rete maggiore. Tuttavia, poiché la trasmissione tra un nodo e un suo vicino è probabilistica, occorrerebbe effettuare un gran numero di simulazioni prima di avere una stima accurata dell'influenza di un nodo sulla rete e ciò rende questo metodo computazionalmente molto costoso.

Un problema molto studiato è quello dell'individuazione degli *influential spreaders* tramite l'analisi della topologia delle rete. Diversi studi mostrano come alcuni algoritmi di centralità possono fornire una buona indicazione di quali sono i nodi più influenti nella diffusione [10][3][13]. Tra questi, alcuni richiedono una conoscenza globale della topologia della rete come *PageRank*, *Betweenness Centrality* e *k-core*, mentre per altri come Degree Centrality o μ-PCI ogni nodo ha bisogno solo di una conoscenza locale.

1.2 *Influential spreaders* in una multilayer network

In molti degli scenari reali, però, i nodi sono collegati tramite diversi tipi di relazione, che non possono essere rappresentati con un grafo. Tra questi troviamo la rete dei profili online collegati dalle loro

interazioni in diversi social networks (follow, amicizie, condivisioni, ecc.) e la rete delle stazioni di una città connesse da tratte di diversi mezzi di trasporto (rete ferroviaria, rete degli autobus, piste ciclabili, ecc.).

Strutture che permettono di rappresentare queste realtà sono le *multilayer networks*, ossia reti in cui sono presenti differenti tipi di relazione tra i nodi. Fornendo un modello più fedele delle reti reali, la loro analisi può dare risultati più accurati rispetto a quelli ottenibili modellando la realtà con un grafo.

In questa tesi si affronterà il problema dell'individuazione degli *influential spreaders* in una *multilayer network*, calcolando diverse misure di centralità e confrontando i risultati con quelli ottenuti simulando degli *spreading processes*.

2 Definizioni

Definizione 1 (Multilayer Network)**.** Una *multilayer network* è una coppia $(\mathcal{G}, \mathcal{E})$ dove $\mathcal{G} = \{G_i \mid i \in \{1, 2, \dots L\}\}$ è un insieme di *grafi* detti *layers* dove un nodo può appartenere ad uno o più *layers* ed $\mathcal{E} = \{E_{ij} \subseteq V_i \times V_j \mid i, j \in \{1, 2, \dots, L\}, i \neq j\}$ è l'insieme delle *inter-connessioni* tra nodi appartenenti a layer differenti.

Definizione 2. Data una *multilayer network* $\mathcal{M} = (\mathcal{G}, \mathcal{E})$ indichiamo con $|V|$ il numero totale di nodi, ossia la cardinalità dell'insieme $V = \bigcup_i V_i$, con N la somma del numero di nodi di ogni layer, ossia $N = \sum_i |V_i|$ e con E il numero totale di archi, $E = \sum_i |E_i| + \sum_{i \neq j} |E_{ij}|$.

Definizione 3 (Multiplex Network)**.** Una *multiplex network* è un tipo di *multilayer network* dove le uniche *inter-connessioni* ammesse sono tra un nodo e le sue controparti negli altri layer, ossia $E_{ij} \subseteq \{(v, v) \mid v \in V_i \cap V_j\}$

Definizione 4 (Grafo aggregato)**.** Sia $\mathcal{M} = (\mathcal{G}, \mathcal{E})$ una *multilayer network*. Definiamo il *grafo aggregato* $G_{agg} = (V_{agg}, E_{agg})$ derivato da $\mathcal{M}$ dove $V_{agg} = V$ ed $E_{agg} = \left(\bigcup_i E_i\right) \cup \left(\bigcup_{i \neq j} E_{ij}\right)$

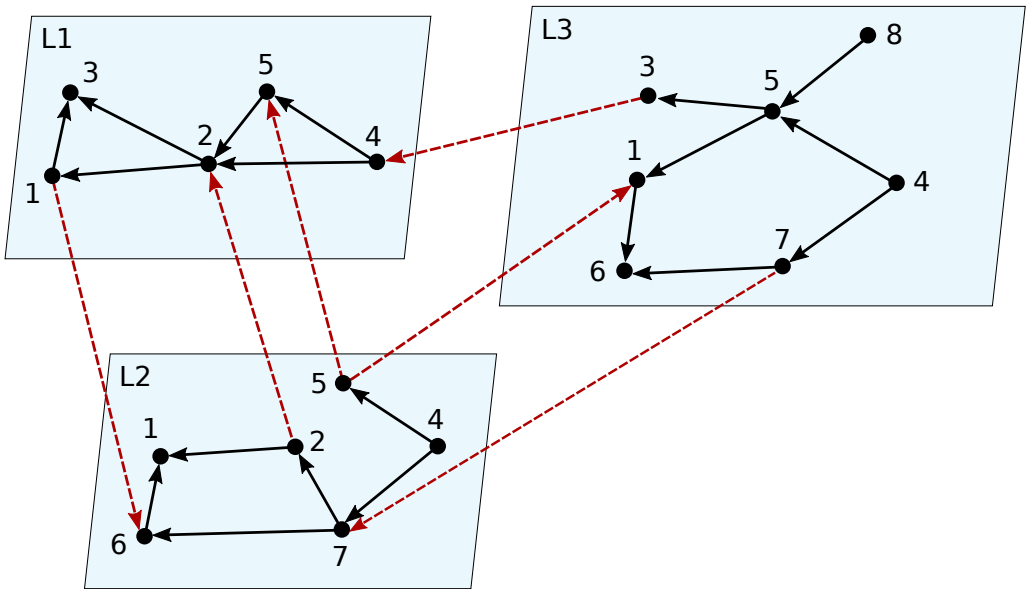

Figura 1: Rappresentazione di una *multilayer network* con tre layer: L1, L2 e L3. In rosso sono evidenziate le *inter-connessioni*, ossia gli archi che collegano nodi in layer differenti.

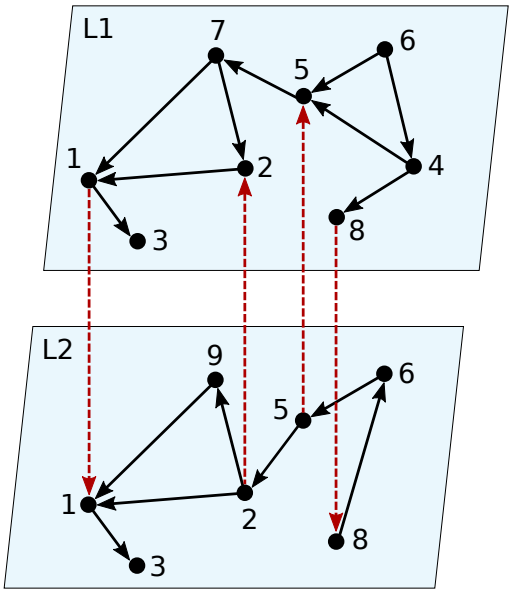

Figura 2: Rappresentazione di una *multiplex network* con due layer: L1 e L2. In questo tipo di rete le uniche *inter-connessioni* possibili sono tra un nodo e le sue controparti negli altri layer.

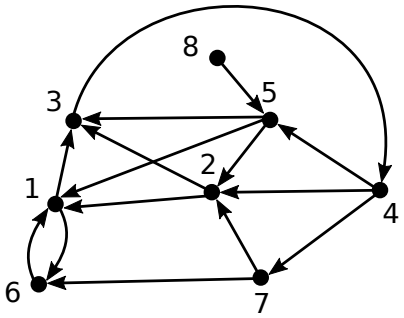

Figura 3: Grafo aggregato derivato dalla *multilayer network* rappresentata in figura 1.

3 Algoritmi utilizzati

Dopo una fase di ricerca, sono stati selezionati ed implementati alcuni algoritmi di centralità solitamente usati per l'individuazione degli *influential spreaders*. Tali algoritmi assegnano ad ogni nodo un valore proporzionale alla sua centralità e quindi all'influenza esercitata da tale nodo sulla rete.

Questi possono essere divisi in tre categorie:

- algoritmi applicati sul grafo aggregato;
- algoritmi applicati separatamente su ogni layer;
- algoritmi applicati sull'intera struttura *multilayer*.

3.1 Algoritmi sul grafo aggregato

Diverse misure di centralità sono utilizzate per identificare degli *influential spreaders* in grafi 'classici' ad un solo layer, detti anche *monoplex* [10][3][13]. Essendo definite per grafi ad un solo layer, sono state applicate al *grafo aggregato* derivato da ogni *multilayer network*. Le misure utilizzate sono state *PageRank* [12], *k-core* [4] e *degree*.

3.1.1 *PageRank* (*aggPR*)

Definizione 5 (*PageRank*)**.** Dato un grafo $G = (V, E)$, il valore di *PageRank* di un nodo v è definito come

$$PR(v) = \alpha \sum_{w:v \in G.adj(w)} \frac{PR(w)}{G.outdegree(w)} + (1-\alpha)\frac{1}{|V|} \tag{3.1}$$

con $0 \leq \alpha \leq 1$.

Questa misura di centralità fu originariamente definita per misurare l'importanza di una pagina web. Secondo questo criterio, una pagina è tanto più importante quante più pagine importanti hanno un link verso di essa. *PageRank* può essere visto come un modello di comportamento di un utente che, trovandosi inizialmente in una pagina casuale, continua a visitare i link che trova, ma occasionalmente riparte da un'altra pagina casuale. Il parametro α è detto $dampingfactor$ e regola la probabilità che si riparta da una pagina casuale ed è stato impostato a 0.85.

Analogamente può essere utilizzata per misurare più genericamente l'influenza di un nodo in un grafo.

Questa definizione ricorsiva è stata così calcolata in modo iterativo:

$$\begin{aligned} PR_0(v) &= \frac{1}{n} \\ PR_i(v) &= \alpha \sum_{w:v \in G.adj(w)} \frac{PR_{i-1}(w)}{G.outdegree(w)} + (1-\alpha)\frac{1}{|V|} \end{aligned} \tag{3.2}$$

finché $||PR_i - PR_{i-1}||_1 < \epsilon$. Secondo [12] il numero di iterazioni è proporzionale a $\log |V|$. Dunque, la complessità dell'algoritmo è $O((|V| + |E|) \log |V|)$.

3.1.2 *k-core* (*aggCore*)

Definizione 6 (*k-core*)**.** Dato un grafo $G = (V, E)$, un *k-core* di G è un sottografo $G' = (V', E')$ tale che

$$G'.indegree(v) \geq k \quad \forall v \in V' \tag{3.3}$$

Un nodo appartenente ad un *k-core* con k elevato è considerato un nodo centrale nella rete, e quindi un potenziale *influential spreader*. Per il calcolo di questa misura è stato utilizzato l'algoritmo definito in [4] con complessità $O(|V| + |E|)$.

3.1.3 *degree* (*aggDeg*)

Questa misura di centralità assegna ad ogni nodo un valore pari al suo *outdegree*, per cui un nodo con tanti vicini si assume possa avere una certa rilevanza in un processo di diffusione. Può essere calcolata con complessità $O(|V|)$.

3.2 Algoritmi applicati su ogni layer

Alcune delle misure sono state estese alle reti multilayer calcolando la centralità dei nodi in ogni layer separatamente e poi sommando i punteggi ottenuti in ogni layer. In questa categoria rientrano *additive PageRank* [8] e *sum k-core* [2]

3.2.1 *additive PageRank* (*addPR*)

Definizione 7 (*additive PageRank*)**.** Data una *multilayer network* $\mathcal{M} = (\mathcal{G}, \mathcal{E})$ ed un ordinamento dei layer $(X_1, \ldots, X_L)$, l'*additive PageRank* di un nodo $v \in V$ è il valore $addPR(v) = addPR_L(v)$, dove:

$$\begin{aligned} addPR_1(v) &= PR(v) \text{ in } X_1 \\ addPR_l(v) &= \alpha \sum_{w:v \in X_l.adj(w)} \frac{addPR_l(w)}{X_l.outdegree(w)} + (1-\alpha)\frac{addPR_{l-1}(v)}{|V|\langle addPR_{l-1}\rangle} \end{aligned} \tag{3.4}$$

Questa estensione dell'algoritmo di PageRank alle *multilayer network* richiede un ordinamento dei layer. Come fatto in [2], i layer sono stati ordinati per valore dell'autovalore di modulo massimo della rispettiva matrice di adiacenza. Infatti, autovalore più grande significa maggiore capacità di diffusione [17].

Come algoritmo, è stato utilizzato un approccio analogo a quello della versione classica di PageRank descritto in 3.1.1 nella pagina precedente. Poiché è necessario calcolare il PageRank su ogni layer, il costo dell'algoritmo è $O(\sum_i((|V_i| + |E_i|)log(|V_i|)))$, assumendo che i layer siano già ordinati.

3.2.2 *sum k-core* (*sumCore*)

Questa misura è stata ottenuta calcolando il valore di *k-core*di ogni nodo in ogni layer, quindi sommando i valori ottenuti da un nodo in tutti i layer. Pertanto la complessità del calcolo è pari a $O(\sum_i(|V_i| + |E_i|)$.

3.3 Algoritmi applicati sull'intera struttura

Diverse misure definite originariamente per grafi *monoplex* sono state estese a reti *multilayer*. Queste, al contrario di quelle presentate nelle sezioni precedenti, tengono conto della struttura a più livelli della rete ed in particolare delle *inter-connessioni*. Quelle implementate sono *versatility PageRank* [7], *versatility Betweenness Centrality* [7] [15], *multiplex k-core* [1] e le diverse varianti di *Power Community Index*: *minimal-layers PCI* [2], *layer-agnostic PCI* [2], *all-layers PCI* [2], *layer-symmetric PCI* [2].

3.3.1 *versatility PageRank* (*verPR*)

Definizione 8 (*versatility PageRank*)**.** Data una *multilayer network* $\mathcal{M} = (\mathcal{G}, \mathcal{E})$, si definisce *versatility PageRank* di un nodo $v \in V_l$ in un layer l il valore:

$$PR(v,l) = \alpha\left(\sum_{w:v\in G_l.adj(w)} \frac{PR(w,l)}{\mathcal{M}.outdegree(w,l)} + \sum_{j\neq l}\sum_{w:(w,v)\in E_{jl}} \frac{PR(w,j)}{\mathcal{M}.outdegree(w,j)}\right) + (1-\alpha)\frac{1}{N} \quad (3.5)$$

dove $\mathcal{M}.outdegree(w,l)$ è la somma del numero di archi uscenti da w nel layer l e del numero di *inter-connessioni* uscenti da w nel layer l.

La definizione originale in [7] utilizzava un tensore 4-dimensionale per rappresentare la rete. Qui è stata generalizzata, anche prevedendo che un nodo possa non comparire in tutti i layer.

Questa definizione è stata calcolata in modo analogo alla definizione 3.1.1 a pagina 7, dunque la complessità è $O((N+E)\log(N))$.

3.3.2 *versatility Betweenness Centrality* (*verBC*)

Definizione 9 (*versatility Betweenness Centrality*)**.** Data una *multilayer network* $\mathcal{M} = (\mathcal{G}, \mathcal{E})$, la *versatility Betweenness Centrality* di un nodo v è il valore

$$BC(v) = \sum_{s,t\in V} \frac{\sigma_{st}(v)}{\sigma_{st}} \quad (3.6)$$

dove σ_{st} è il numero di percorsi minimi tra il nodo s ed il nodo t indipendentemente dal layer, e $\sigma_{st}(v)$ è il numero di questi che passa per il nodo v in qualche layer.

Questa misura è stata calcolata con l'algoritmo definito in [15], con complessità $O\left(|V|\,(N+E)\right)$.

3.3.3 *multiplex k-core* (*multiCore*)

Definizione 10 (*multiplex k-core*)**.** Data una *multilayer network* $\mathcal{M} = (\mathcal{G}, \mathcal{E})$, il *multiplex k-core* è il più grande sottografo per cui ogni nodo è raggiunto in ogni layer da almeno k archi o *inter-connessioni*.

Questa misura è stata calcolata estendendo l'algoritmo definito in [4] per adattarlo a questa definizione, ottenendo una complessità di $O(N+E)$.

3.3.4 *Power Community Index* (*PCI*)

Definizione 11. Data una *multilayer network* $\mathcal{M} = (\mathcal{G}, \mathcal{E})$, l'indice $mlPCI_n$ di un nodo v in un layer l è il massimo numero k tale che esistono almeno k vicini di v nel layer l in $\mathcal{M}$ con numero di vicini in almeno n layer maggiore o uguale a k.

Da questa definizione si ricavano le seguenti:

Definizione 12 (*minimal-layers PCI*)**.** L'indice $mlPCI$ di un nodo v in un layer l è definito come

$$mlPCI(v,l) = \sum_{n=1}^{L} mlPCI_n(v,l) \tag{3.7}$$

Definizione 13 (*all-layers PCI*)**.** L'indice $alPCI$ di un nodo v in un layer l è definito come

$$alPCI(v,l) = mlPCI_L(v,l) \tag{3.8}$$

Definizione 14 (*layer-symmetric PCI*)**.** L'indice $lsPCI$ di un nodo v in un layer l è il massimo numero k tale che esistono almeno k vicini di v nel layer l in $\mathcal{M}$ con numero di vicini in almeno k layer maggiore o uguale a k.

Definizione 15 (*layer-agnostic PCI*)**.** L'indice $laPCI$ di un nodo v in un layer l è è il massimo numero k tale che esistono almeno k vicini di v nel layer l in $\mathcal{M}$ con numero di vicini maggiore o uguale a k.

Il calcolo di queste definizioni, è stato fatto utilizzando un algoritmo di complessità $O(NLt \log t)$, dove $t = \max\{\mathcal{M}.outdegree(v) \mid v \in V\}$

4 Dataset

Queste misure sono state testate su diverse reti con caratteristiche diverse, seguendo una metodologia simile a [2]:

- *Multiplex networks* estratte da dataset biologici e sociali;
- *Multilayer networks* generate a partire da reti di interazioni in Internet.

4.1 *Multiplex networks*

4.1.1 Descrizione dei dataset

Le reti indicate in tabella 1 a pagina 13 sono state estratte a partire da reti biologiche e sociali fornite da [5]. In particolare i dataset di partenza sono i seguenti [16][6]:

Drosophila *multiplex network* delle interazioni genetiche e proteiche di Drosophila Melanogaster. I layer rappresentano:

1. Direct interaction;
2. Suppressive genetic interaction defined by inequality;
3. Additive genetic interaction defined by inequality;
4. Physical association;
5. Colocalization;
6. Association;
7. Synthetic genetic interaction defined by inequality.

Homo *multiplex network* delle interazioni genetiche e proteiche di Homo Sapiens. I layer rappresentano:

1. Direct interaction;
2. Physical association;
3. Suppressive genetic interaction defined by inequality;
4. Association;
5. Colocalization;
6. Additive genetic interaction defined by inequality;
7. Synthetic genetic interaction defined by inequality.

MA2013 *multiplex network* che rappresenta differenti tipi di relazioni sociali tra utenti di Twitter durante i campionati del mondo di atletica leggera di Mosca 2013. I layer rappresentano:

1. Retweet;
2. Mentions;
3. Replies.

NYCM2014 *multiplex network* che rappresenta differenti tipi di relazioni sociali tra utenti di Twitter durante la Marcia per il Clima svoltasi a New York nel 2014. I layer rappresentano:

1. Retweet;
2. Mentions;
3. Replies.

SacchCere *multiplex network* delle interazioni genetiche e proteiche di Saccharomyces Cerevisiae. I layer rappresentano:

1. Physical association;
2. Suppressive genetic interaction defined by inequality;
3. Direct interaction;
4. Synthetic genetic interaction defined by inequality;
5. Association;
6. Colocalization;
7. Additive genetic interaction defined by inequality.

SacchPomb *multiplex network* delle interazioni genetiche e proteiche di Saccharomyces Pombe. I layer rappresentano:

1. Direct interaction;
2. Colocalization;
3. Physical association;
4. Suppressive genetic interaction defined by inequality;
5. Synthetic genetic interaction defined by inequality;
6. Additive genetic interaction defined by inequality;
7. Association.

4.1.2 Estrazione delle *multiplex networks*

L'obiettivo era quello di ottenere un dataset simile a quello utilizzato da [2], dove da ogni rete viene estratta una sottorete in cui ogni nodo è presente in tutti i layer. Nell'articolo sono indicati per ogni rete i numeri N, E ed $|L|$, rispettivamente di nodi, archi e layer estratti.

Per ogni rete, è stato individuato un sottoinsieme L di $|L|$ layers con numero di nodi con una controparte in ogni layer maggiore o uguale a N. Per tutte le reti, è stato trovato un solo sottoinsieme che rispettasse questo criterio.

Dopo aver costruito una sottorete formata solo dai layer in L, sono stati rimossi i nodi non presenti in tutti i layer e gli archi da e verso essi. Dopodiché sono stati rimossi tutti i nodi da cui non arrivava né partiva più nessun arco. Questa procedura è stata applicata ricorsivamente alla sottorete ottenuta, finché non è stato più possibile rimuovere alcun nodo.

I risultati ottenuti sono stati molto simili a quelli voluti, come si può vedere in tabella 1 nella pagina successiva.

In queste reti sono state aggiunte *inter-connessioni* tra le controparti di ogni nodo in ogni layer.

4.2 Generazione reti multilayer

4.2.1 Descrizione dei dataset

Sono state generate delle *multilayer network* partendo da grafi di ottenuti da diverse applicazioni peer-to-peer, in particolare quelli indicati in tabella 2 a fronte, che si possono trovare in [11].

Tabella 1: Reti multiplex estratte

Nome	Nodi			Archi			Layers	
	iniziali	obiettivo	estratti	iniziali	obiettivo	estratti	iniziali	estratti
Drosophila	8215	1364	1375	43 366	7267	7438	7	2
Homo	18 222	3859	3878	170 899	77 483	78 804	7	3
MA2013	88 804	4370	4377	210 250	33 411	34 589	3	3
NYCM2014	102 439	4150	4262	353 495	45 334	47 840	3	3
SacchCere	6570	3096	3121	28 275	185 849	188 182	7	5
SacchPomb	4092	875	878	63 676	18 214	18 308	7	3

Tabella 2: Dataset per la generazione di reti multilayer

Numero	Nome	Nodi	Archi
1.	wiki-Vote	7115	103 689
2.	cit-HepTh	27 770	352 807
3.	p2p-Gnutella04	10 876	39 994
4.	p2p-Gnutella05	8846	31 839
5.	p2p-Gnutella06	8717	31 525
6.	p2p-Gnutella08	6301	20 777

I grafi utilizzati sono:

wiki-Vote Grafo delle votazioni per le elezioni degli amministratori di Wikipedia svoltesi fino al gennaio 2008. I nodi del grafo rappresentano utenti di Wikipedia, la metà circa sono admin mentre gli altri sono utenti ordinari, e un arco diretto dal nodo i al nodo j indica che l'utente i ha votato per l'utente j;

cit-HepTh Grafo delle citazioni tra articoli sul sito arXiv pubblicati tra gennaio 1993 e aprile 2003. Se un paper i cita un paper j, il grafo contiene un arco diretto da i a j. Non è contenuta alcuna informazione riguardo articoli non presenti sul sito;

p2p-Gnutella04, **p2p-Gnutella05**, **p2p-Gnutella06**, **p2p-Gnutella08** Grafi che rappresentano 'istantanee' della rete di file sharing peer-to-peer Gnutella, nell'agosto 2002. In questi grafi i nodi rappresentano gli host della rete Gnutella e gli archi rappresentano connessioni tra gli host.

4.2.2 Generazione delle *multilayer networks*

Seguendo l'approccio di [2], sono stati generati due tipi di reti:

- Similar Layers Network(SLN), i cui layer sono i grafi [3-6] della tabella 2. In queste reti i layer hanno tutti un numero simile di nodi e archi;
- Different Layers Networks(DLN), i cui layer sono i grafi [1-3] della tabella 2. In queste reti i layer hanno numero di nodi e archi molto differenti.

Per prima cosa, sono state aggiunte *inter-connessioni* tra le controparti di ogni nodo in tutti i layer in cui esso è presente.

Quindi, sono state generate casualmente altre *inter-connessioni* impostando tre parametri:

1. il numero di *inter-connessioni* uscenti da un nodo in un determinato layer;
2. la frequenza con cui un layer viene scelto come layer di destinazione di una connessione;

3. la frequenza con cui un nodo in un certo layer viene scelto come nodo di destinazione.

Ognuno di questi parametri è stato generato secondo una distribuzione che segue la legge di Zipf [18], per cui la frequenza di un certo valore è inversamente proporzionale al valore stesso.

Tale distribuzione genera numeri tra 1 ed un valore massimo m (escluso) con diverse probabilità, secondo il valore di una variabile $s \in [0, 1]$ che regola il grado di 'asimmetria' dei valori generati, in particolare:

- per $s = 0$ la probabilità $\mathcal{P}$ di ogni valore è la stessa;
- per $s = 1$ si ha $\mathcal{P}(0) = 2 \cdot \mathcal{P}(1) = 3 \cdot \mathcal{P}(2) = \cdots = m \cdot \mathcal{P}(m-1)$.

Le variabili che regolano i tre parametri sono chiamate rispettivamente s_{degree}, s_{layer}, s_{node}. La probabilità che un nodo abbia k *inter-connessioni* è $\mathcal{P}_{degree}(k)$. Per ognuna sono stati sperimentati i valori 0.3 e 0.8. Per quanto riguarda il numero di *inter-connessioni* di un nodo in un determinato layer il valore massimo è stato impostato a $d \cdot \log_2 \sum_i V_i$, con $d = 2$. Per gli altri parametri, invece, tutti i layer devono essere selezionabili così come tutti i nodi all'interno di un layer.

L'algoritmo 1 a fronte mostra come sono state generate le reti. Nell'algoritmo gli oggetti della classe ZipfGenerator sono generatori di numeri casuali secondo una distribuzione che segue la legge di Zipf. È stato utilizzato un generatore per ognuno dei tre parametri, in particolare:

1. *degree_generator* genera numeri tra 0 e $max\_interconnectons - 1$ con $s = s_{degree}$ che vengono utilizzati direttamente per definire il numero di *inter-connessioni* uscenti da ogni nodo;
2. *layer_generator* genera numeri tra 0 e $L - 1$, dove L è il numero di layers, con $s = s_{layer}$. Questi numeri vengono utilizzati come indice per scegliere il layer di destinazione all'interno di una permutazione dei layer $perm\_l$;
3. *node_generators* è un vettore di generatori in cui l'i-esimo elemento $node\_generators[i]$ genera numeri tra 0 e $V_i - 1$, dove V_i è il numero di nodi dell'i-esimo layer, con $s = s_{node}$. Questi numeri vengono utilizzati come indice per scegliere il nodo di destinazione all'interno di una permutazione dei nodi del layer i selezionato chiamata $perm\_n[i]$.

Le reti così generate sono denominate con $\text{SLN}_{s_{degree},s_{layer},s_{node}}$ oppure $\text{DLN}_{s_{degree},s_{layer},s_{node}}$

Algoritmo 1: Generazione multilayer networks

```
Input : GRAPH layers[]// grafi usati come layers
        int perm_l[]   // posizione di ogni layer nella permutazione
        int perm_n[][] // posizione di un nodo nella permutazione di ogni layer
        int d
        float s_degree
        float s_layer
        float s_node
int total_nodes = 0
for i = 0 to layers.size() do
    total_nodes = total_nodes + layers[i].nodes().size()
int max_interconnections = d · ⌊log_2 (total_nodes)⌋
// I due parametri della classe ZIPFGENERATOR regolano rispettivamente
// il massimo della distribuzione e il grado di asimmetria
ZIPFGENERATOR degree_generator(max_interconnections, s_degree)
ZIPFGENERATOR layer_generator(layers.size(), s_layer)
ZIPFGENERATOR node_generators[0 ... layers.size() − 1]
for l = 0 to layers.size() − 1 do
    node_generators[l] = ZIPFGENERATOR(layers[l].size(), s_degree)
MULTILAYERNETWORK m
for l = 0 to layers.size() − 1 do
    foreach n ∈ layers[l]do
        // intra-connessioni
        for v ∈ layers[l].adj(n) do
            m.add_edge(n, l, v, l)
        // inter-connessioni tra un nodo e le sue controparti negli altri layer
        for j = 0 to layers.size() − 1 do
            if j ≠ l and n ∈ layers[j].nodes() then
                m.add_edge(n, l, n, j)
        // Generazione inter-connessioni casuali
        int degree = degree_generator.next()
        for i = 1 to degree do
            bool added = False
            while not added do
                int l_dest = l_index[layer_generator.next()]
                while l_dest == l do
                    l_dest = l_index[layer_generator.next()]
                int n_dest = n_index[l_dest][node_generators[l_dest].next()]
                if not m.has_edge(n, l, n_dest, l_dest) then
                    added = True
                    m.add_edge(n, l, n_dest, l_dest)
```

5 Simulazioni

Per valutare se un algoritmo possa essere utilizzato per riconoscere gli *influential spreaders*, sono stati confrontati i valori di centralità che esso assegna ai vari nodi con la frazione di rete infettata da essi simulando un processo di diffusione, detto *spreading process*.

Lo *spreading process* utilizzato segue il modello *SIR*, in cui un nodo può essere in 3 stati:

***Susceptible* (*S*)** ossia vulnerabile ad essere infettato;

***Infectious* (*I*)** ossia infetto. Un nodo in questo stato può contagiare i propri vicini con una certa probabilità;

***Recovered* (*R*)** ossia guarito. Una volta in questo stato, un nodo non può più essere infettato.

5.1 Probabilità di contagio

Un valore da impostare in uno *spreading process* è la probabilità λ che un nodo nello stato I ha di contagiare i propri vicini nello stato S, detta *epidemic probability*. La scelta del valore di λ è molto importante per riconoscere gli *influential spreaders*, in quanto con un valore troppo alto si osserverebbe la diffusione dell'infezione in tutta la rete indipendentemente dal nodo inizialmente infetto, mentre con un valore troppo basso l'epidemia non riuscirebbe ad espandersi nemmeno dai nodi più influenti.

In un grafo *monoplex* G il valore di *critical epidemic point*

$$\lambda_c = \frac{\langle k \rangle}{\langle k^2 \rangle} \tag{5.1}$$

dove k è l'*outdegree* di un nodo, rappresenta un'approssimazione della soglia per cui, indipendentemente dal nodo da cui parte l'epidemia, se $\lambda > \lambda_c$ essa riesce a diffondersi in gran parte della rete, mentre se $\lambda < \lambda_c$ essa rimane confinata in una piccola parte della rete[14].

Seguendo quanto fatto in [2], la *epidemic probability* λ_{ii} da un nodo nel layer i ad uno dello stesso layer è stata impostata al *critical epidemic point* del layer G_i, mentre λ_{ij} tra nodi in layer diversi è stata impostata al *critical epidemic point* del grafo aggregato.

Questi valori rispecchiano l'intuizione secondo cui il contagio si trasmette più facilmente tra nodi sullo stesso layer. Questo succede per esempio nel caso dei social network, in cui diverse piattaforme tendono a mostrare in maniera minore contenuti con collegamenti a piattaforme concorrenti, e nel caso dei ritardi accumulati da una stazione in una rete di trasporti, che possono trasmettersi facilmente alle stazioni vicine nella stessa rete (ad esempio la rete ferroviaria), ma anche, indirettamente e presumibilmente in maniera minore, a stazioni di reti di trasporti differenti (ad esempio la rete degli autobus).

6 Risultati

6.1 Metodo di valutazione

Per ogni rete è stato osservato il numero di nodi infettati da ogni nodo facendo una media di 100 simulazioni e ottenendo così un vettore s di $|V|$ elementi, dove s_i rappresenta il numero di nodi infettati in media in uno *spreading process* in cui i nodi inizialmente infetti sono le controparti del nodo i nei layer in cui esso è presente.

È stato poi calcolato per ogni algoritmo un vettore a dove a_i è il punteggio assegnato dall'algoritmo al nodo i.

Per assegnare ad ogni algoritmo un valore che ne rispecchi la capacità di riconoscere gli *influential spreaders* sono stati confrontati i vettori s e a utilizzando il coefficiente *Kendall's* τ [9], così definito:

$$\tau = \frac{n_c - n_d}{n(n-1)} \tag{6.1}$$

dove:

- n è il numero di elementi dei due vettori, ossia il numero di nodi |V|;
- n_c è il numero di coppie concordanti nei due vettori. Una coppia di elementi indici i, j si dice concordante se $a_i > a_j$ e $s_i > s_j$ oppure se $a_i < a_j$ e $s_i < s_j$;
- n_d è il numero di coppie discordanti nei due vettori. Una coppia di elementi indici i, j si dice discordante se $a_i > a_j$ e $s_i < s_j$ oppure se $a_i < a_j$ e $s_i > s_j$;

Se $a_i = a_j$ oppure $s_i = s_j$, la coppia i, j non viene contata.

Il risultato della formula è compreso tra -1.0 e 1.0.

6.2 Confronto dei risultati

Le tabelle 3, 4 e 5 mostrano i punteggi ottenuti da ogni algoritmo nelle reti DLN, SLN e Multiplex rispettivamente. In ogni tabella sono evidenziati per ogni rete i punteggi ottenuti dai 3 algoritmi migliori.

Si può notare che *mlPCI* e *laPCI* sono i due algoritmi che ottengono punteggi buoni su tutte le reti utilizzate, e possono quindi essere considerati un buon metodo per identificare gli *influential spreaders*. Altri algoritmi come *alPCI* e *aggDeg* sebbene ottengano ottimi risultati su particolari tipi di rete, ottengono scarsi risultati sugli altri; non sono pertanto da considerare affidabili per l'individuazione degli *influential spreaders*.

Algoritmi applicati al grafo trasposto Le diverse versioni degli algoritmi di *PageRank* e *k-core* sono state applicate anche al grafo trasposto.

Tabella 3: Kendall's tau di diversi algoritmi in reti DLN

	$DLN_{0.3,0.3,0.3}$	$DLN_{0.3,0.3,0.8}$	$DLN_{0.3,0.8,0.3}$	$DLN_{0.3,0.8,0.8}$	$DLN_{0.8,0.3,0.3}$	$DLN_{0.8,0.3,0.8}$	$DLN_{0.8,0.8,0.3}$	$DLN_{0.8,0.8,0.8}$
addPR	0.4885	0.4844	0.4888	0.4860	0.4872	0.4874	0.4853	0.4848
aggCore	0.5575	0.5376	0.5552	0.5379	0.5596	0.5391	0.5578	0.5416
aggDeg	0.6620	0.6402	0.6607	0.6401	0.6274	0.6201	0.6297	0.6179
aggPR	0.5316	0.4958	0.5296	0.4972	0.5214	0.4879	0.5195	0.4869
alPCI	0.6945	0.7036	0.7043	0.7109	0.6565	0.6694	0.6776	0.6800
laPCI	0.7145	0.7157	0.7113	0.7158	0.6759	0.6910	0.6783	0.6868
lsPCI	0.6313	0.6285	0.6354	0.6335	0.6840	0.6848	0.6879	0.6895
mlPCI	0.7344	0.7406	0.7305	0.7427	0.6980	0.7114	0.7027	0.7095
multiCore	0.3551	0.3574	0.3554	0.3576	0.3551	0.3549	0.3553	0.3550
sumCore	0.5225	0.5350	0.5259	0.5346	0.5279	0.5350	0.5259	0.5334
verBC	0.6586	0.5605	0.6610	0.5639	0.6275	0.5404	0.6309	0.5441
verPR	0.5275	0.5141	0.5261	0.5136	0.5195	0.5078	0.5162	0.5064

Tabella 4: Kendall's tau di diversi algoritmi in reti SLN

	$SLN_{0.3,0.3,0.3}$	$SLN_{0.3,0.3,0.8}$	$SLN_{0.3,0.8,0.3}$	$SLN_{0.3,0.8,0.8}$	$SLN_{0.8,0.3,0.3}$	$SLN_{0.8,0.3,0.8}$	$SLN_{0.8,0.8,0.3}$	$SLN_{0.8,0.8,0.8}$
addPR	0.4225	0.4293	0.4216	0.4295	0.4198	0.4309	0.4178	0.4287
aggCore	0.4066	0.3684	0.4158	0.3731	0.4010	0.3795	0.4117	0.3841
aggDeg	0.6073	0.5520	0.6009	0.5460	0.6147	0.5490	0.6134	0.5412
aggPR	0.4275	0.3619	0.4313	0.3695	0.4229	0.3713	0.4294	0.3779
alPCI	0.6550	0.6152	0.6379	0.5971	0.6675	0.6300	0.6555	0.6203
laPCI	0.6243	0.5761	0.6195	0.5687	0.6685	0.6208	0.6662	0.6092
lsPCI	0.5901	0.5681	0.5728	0.5479	0.5962	0.5678	0.5897	0.5584
mlPCI	0.6265	0.5810	0.6222	0.5771	0.6660	0.6169	0.6627	0.6057
multiCore	0.3481	0.3560	0.3498	0.3576	0.3506	0.3535	0.3472	0.3513
sumCore	0.4209	0.4288	0.4211	0.4273	0.4164	0.4291	0.4171	0.4267
verBC	0.5589	0.4528	0.5480	0.4506	0.5868	0.4842	0.5739	0.4796
verPR	0.3650	0.3479	0.3697	0.3579	0.3200	0.3297	0.3317	0.3374

Tabella 5: Kendall's tau di diversi algoritmi in reti multiplex

	Drosophila	Homo	MA2013	NYCM2014	SacchCere	SacchPomb
addPR	0.0398	0.3424	0.0059	−0.1313	0.3300	0.3410
aggCore	0.0646	0.4132	0.0572	−0.0453	0.4384	0.1046
aggDeg	0.7355	0.7096	0.5711	0.6150	0.6886	0.7656
aggPR	0.0417	0.3857	0.0164	−0.0771	0.3944	0.2562
alPCI	0.3682	0.1588	0.0493	0.1124	0.0455	0.4502
laPCI	0.6040	0.6859	0.5534	0.6178	0.6980	0.6853
lsPCI	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
mlPCI	0.6947	0.7191	0.5532	0.6028	0.7073	0.7729
multiCore	0.0497	0.1962	0.0428	0.0230	0.1925	−0.0170
sumCore	0.0658	0.4575	0.0616	−0.0487	0.4546	0.1000
verBC	0.5243	0.5653	0.1726	0.2045	0.5666	0.6550
verPR	−0.3505	−0.0606	−0.2707	−0.4107	0.0004	−0.4201

Nel caso di *PageRank*, se nella definizione sul grafo originale un nodo è tanto più importante quanto più importanti sono i nodi che hanno un arco verso di esso, applicando l'algoritmo sul grafo trasposto si ottiene una definizione per cui un nodo è tanto più importante quanto più importanti sono i nodi verso cui ha un arco.

Per quanto riguarda *k-core*, invece, nella definizione sul grafo originale un *k-core* di un grafo è un sottografo massimale in cui ogni nodo ha almeno k archi entranti, applicando l'algoritmo sul grafo trasposto si ottiene una definizione analoga, in cui invece degli archi entranti si considerano gli archi uscenti.

Per entrambi gli algoritmi si è pensato che le definizioni di centralità ottenute applicando l'algoritmo sul grafo trasposto potessero individuare meglio gli *influential spreaders*. Le prestazioni di questi algoritmi, indicati con l'indice ^{T}, sono mostrate nelle tabelle 6, 7 e 8.

In generale, si può osservare come nella maggior parte dei casi questi ottengano risultati migliori se confrontati con la rispettiva versione applicata al grafo originale.

Tabella 6: Kendall's tau di algoritmi applicati al grafo trasposto in reti DLN

	$DLN_{0.3,0.3,0.3}$	$DLN_{0.3,0.3,0.8}$	$DLN_{0.3,0.8,0.3}$	$DLN_{0.3,0.8,0.8}$	$DLN_{0.8,0.3,0.3}$	$DLN_{0.8,0.3,0.8}$	$DLN_{0.8,0.8,0.3}$	$DLN_{0.8,0.8,0.8}$
addPR^T	0.3649	0.4137	0.3601	0.4134	0.3744	0.4136	0.3755	0.4110
aggCore^T	0.6063	0.5604	0.5946	0.5740	0.5718	0.5698	0.5823	0.5809
aggPR^T	0.6161	0.5469	0.6115	0.5461	0.5484	0.4965	0.5496	0.4930
multiCore^T	0.3646	0.3635	0.3647	0.3632	0.3648	0.3633	0.3649	0.3644
sumCore^T	0.4377	0.4813	0.4340	0.4824	0.4468	0.4816	0.4474	0.4804
verPR^T	0.6985	0.6547	0.6934	0.6535	0.6645	0.6554	0.6647	0.6364

Tabella 7: Kendall's tau di algoritmi applicati al grafo trasposto in reti SLN

	$SLN_{0.3,0.3,0.3}$	$SLN_{0.3,0.3,0.8}$	$SLN_{0.3,0.8,0.3}$	$SLN_{0.3,0.8,0.8}$	$SLN_{0.8,0.3,0.3}$	$SLN_{0.8,0.3,0.8}$	$SLN_{0.8,0.8,0.3}$	$SLN_{0.8,0.8,0.8}$
addPR^T	0.4674	0.4831	0.4714	0.4829	0.4675	0.4824	0.4676	0.4849
aggCore^T	0.4608	0.4225	0.4601	0.4300	0.5110	0.4362	0.4811	0.4663
aggPR^T	0.5900	0.5700	0.5822	0.5625	0.5814	0.5413	0.5815	0.5362
multiCore^T	0.4138	0.3830	0.4135	0.4085	0.4137	0.4084	0.4139	0.4068
sumCore^T	0.6339	0.6559	0.6382	0.6565	0.6296	0.6545	0.6316	0.6566
verPR^T	0.6210	0.6245	0.6252	0.6251	0.6323	0.6240	0.6386	0.6207

Tabella 8: Kendall's tau di algoritmi applicati al grafo trasposto in reti multiplex

	Drosophila	Homo	MA2013	NYCM2014	SacchCere	SacchPomb
addPR^T	0.6497	0.5770	0.3193	0.4235	0.5356	0.6701
aggCore^T	0.5686	0.7138	0.4756	0.5475	0.7073	0.7640
aggPR^T	0.6441	0.7033	0.3156	0.4263	0.7202	0.7435
multiCore^T	0.0763	0.2165	0.1491	0.0127	0.2569	0.2110
sumCore^T	0.4377	0.6689	0.4791	0.5380	0.6899	0.7464
verPR^T	0.5280	0.5797	0.2286	0.3878	0.6370	0.6799

Bibliografia

[1] N. Azimi-Tafreshi, J. Gómez-Gardeñes, and S. N. Dorogovtsev. $k-$core percolation on multiplex networks. *Phys. Rev. E*, 90:032816, Sep 2014.

[2] P. Basaras, G. Iosifidis, D. Katsaros, and L. Tassiulas. Identifying influential spreaders in complex multilayer networks: A centrality perspective. *IEEE Transactions on Network Science and Engineering*, 6(1):31–45, Jan 2019.

[3] P. Basaras, D. Katsaros, and L. Tassiulas. Detecting influential spreaders in complex, dynamic networks. *Computer*, 46(4):24–29, April 2013.

[4] Vladimir Batagelj and Matjaz Zaversnik. An o(m) algorithm for cores decomposition of networks. *CoRR*, cs.DS/0310049, 2003.

[5] FBK CoMuNe Lab. Datasets released for reproducibility. `https://comunelab.fbk.eu/data.php`.

[6] Manlio De Domenico, Vincenzo Nicosia, Alexandre Arenas, and Vito Latora. Structural reducibility of multilayer networks. *Nature Communications*, 6(1):6864, 2015.

[7] Manlio De Domenico, Albert Solé-Ribalta, Elisa Omodei, Sergio Gómez, and Alex Arenas. Ranking in interconnected multilayer networks reveals versatile nodes. *Nature Communications*, 6(1):6868, Apr 2015.

[8] Arda Halu, Raúl J. Mondragón, Pietro Panzarasa, and Ginestra Bianconi. Multiplex pagerank. *PloS one*, 8(10):e78293–e78293, Oct 2013.

[9] M. G. Kendall. A new measure of rank correlation. *Biometrika*, 30(1-2):81–93, 06 1938.

[10] Maksim Kitsak, Lazaros K. Gallos, Shlomo Havlin, Fredrik Liljeros, Lev Muchnik, H. Eugene Stanley, and Hernán A. Makse. Identification of influential spreaders in complex networks. *Nature Physics*, 6(11):888–893, Nov 2010.

[11] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

[12] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999.

[13] Sen Pei, Lev Muchnik, José S. Andrade Jr., Zhiming Zheng, and Hernán A. Makse. Searching for superspreaders of information in real-world social media. *Scientific Reports*, 4(1):5547, Jul 2014.

[14] Anna Saumell-Mendiola, M. Ángeles Serrano, and Marián Boguñá. Epidemic spreading on interconnected networks. *Phys. Rev. E*, 86:026106, Aug 2012.

[15] Albert Solé-Ribalta, Manlio De Domenico, Sergio Gómez, and Alex Arenas. Centrality rankings in multiplex networks. In *Proceedings of the 2014 ACM Conference on Web Science*, WebSci '14, page 149–155, New York, NY, USA, 2014. Association for Computing Machinery.

[16] Chris Stark, Bobby-Joe Breitkreutz, Teresa Reguly, Lorrie Boucher, Ashton Breitkreutz, and Mike Tyers. Biogrid: a general repository for interaction datasets. *Nucleic acids research*, 34(Database issue):D535–D539, Jan 2006. 34 (1).

[17] Yang Wang, D. Chakrabarti, Chenxi Wang, and C. Faloutsos. Epidemic spreading in real networks: an eigenvalue viewpoint. In *22nd International Symposium on Reliable Distributed Syst ems, 2003. Proceedings.*, pages 25–34, Oct 2003.

[18] George Kingsley Zipf. *Human behavior and the principle of least effort.* Human behavior and the principle of least effort. Addison-Wesley Press, Oxford, England, 1949.