Università degli Studi di Padova
Dipartimento di Scienze Statistiche
Corso di Laurea in

Statistica, Economia e Finanza

ESTENSIONE DELLA NETWORK ANALYSIS AL CASO MULTIPLEX: FORMALIZZAZIONE, DESCRIZIONE E PREDISPOSIZIONE DI UN PACCHETTO STATISTICO PER L'IMPLEMENTAZIONE DELLE PRINCIPALI MISURE

Relatore: dott.ssa Giovanna Menardi
Dipartimento di Scienze Statistiche

Laureando: Emanuele Degani
Matricola n. 1075669

Anno Accademico 2015/2016

Ai miei genitori

Indice

1 Introduzione **1**

2 Reti Monoplex **5**
2.1 Formalizzazione matematica 6
2.2 Descrittori 7
2.3 Misure di prominenza 13
2.4 Descrizione di una rete monoplex mediante la notazione indiciale 20
2.4.1 Cenni sulla notazione indiciale 20
2.4.2 Applicazione alle reti 22

3 Reti Multiplex **27**
3.1 Reti multilayer e reti multiplex 28
3.2 Formalizzazione matematica 30
3.3 Operazioni algebriche 33
3.4 Descrittori 35
3.5 Misure di prominenza 40

4 Pacchetto di funzioni sviluppate **47**
4.1 Generalità del pacchetto 47
4.2 Creazione della struttura multiplex ed estrazione degli elementi 49
4.3 Semplificazione della struttura multiplex 54
4.4 Descrittori 55
4.4.1 Estensione delle misure per reti monoplex 55
4.4.2 Misure di eterogeneità 57
4.4.3 Misure di transitività per reti multiplex 58
4.5 Indici di centralità per reti multiplex 60

5 Applicazione su dati reali **63**
5.1 Descrizione del dataset 63
5.2 Creazione della struttura multiplex ed estrazione degli elementi 65
5.3 Estensione delle principali misure per reti monoplex . . 68
5.4 Misure di eterogeneità 72
5.5 Misure di transitività per reti multiplex 78
5.6 Indici di multiplex centrality 83

6 Conclusioni e possibili estensioni **89**

A Codice delle funzioni sviluppate **93**

B Codice per le heatmap **109**

Bibliografia **111**

Capitolo 1

Introduzione

Negli ultimi decenni, la *network analysis* (analisi delle strutture a rete) ha attratto un sempre maggiore interesse negli ambiti della sociologia e delle scienze comportamentali: gran parte di questo interesse è certamente attribuibile alla necessità di capire in che modo delle entità sociali entrino in relazione tra di loro, di individuare eventuali regolarità e strutture sociali entro cui le entità si auto-identificano, e quindi capire quali implicazioni comportino a livello globale sull'intera comunità sottoposta all'analisi. I primi concetti di *(social) network analysis* risalgono ad alcuni studi svolti in ambito sociologico da Barnes (1954), con l'obiettivo di studiare una comunità di soggetti ed il modo in cui questi interagiscono tra di loro rispetto ad una determinata relazione osservata; all'interno di questa disciplina hanno poi preso piede i primi lavori orientati a colmare l'esigenza sempre più impellente di individuare degli strumenti in grado sia di descrivere che di fare inferenza su tali interazioni.

Con il passare degli anni, altre discipline si sono avvicinate all'argomento, contribuendo alla formalizzazione di un impianto il più possibile generale e alla creazione di misure descrittive declinabili nel loro significato ai rispettivi ambiti di ricerca quali, per esempio, la sociologia, l'antropologia e le scienze umane, ma anche l'ingegneria, l'informatica e le scienze biologiche. Sebbene queste misure facciano uso di concetti affini, spesso la loro definizione avviene con una terminologia diversa a seconda del contesto entro il quale è stata proposta: questo limita certamente la loro diffusione e il loro utilizzo anche all'interno di altre discipline.

Una prima svolta in tal senso è stata l'applicazione, in questo contesto, della teoria dei grafi: questa ha reso possibile uniformare entro un'unico schema generale e *neutro* (ovvero, non direttamente associabi-

le nella sua terminologia e formalizzazione ad una particolare disciplina) gli studi fino ad allora diffusi nei diversi ambiti di ricerca. In particolare, in ambito statistico/matematico, in cui di norma si assume l'indipendenza delle osservazioni raccolte da diverse unità su un generico set di variabili, la *network analysis* permette di focalizzare l'interesse sulle relazioni di interdipendenza che intervengono tra le entità che compongono la rete fornendo definizioni formali, misure e descrittori e stimando mediante modelli le teorie entro le quali i concetti chiave sono espressi come processi relazionali o risultati strutturali.

L'approccio *classico* e storicamente consolidato nello studio delle reti prevede, nella sua forma più generale, che la realtà possa essere rappresentata da un insieme di entità collegate tra di loro tramite associazioni legate ad un particolare tipo di relazione che si intende esaminare: questa può avere una natura astratta (come, ad esempio, l'amicizia) o più facilmente osservabile (come, ad esempio, i rapporti in un ambiente lavorativo), ma in entrambi i casi rappresenta un singolo ed unico aspetto entro il quale la relazione si può configurare.

Con la sempre maggiore notorietà che la *network analysis* ha acquisito nel corso dei decenni, l'approfondimento degli strumenti inizialmente ideati per studiare le reti e l'introduzione del loro studio in nuovi settori disciplinari, si è sempre più reso necessario introdurre strutture più complicate e sofisticate in grado di tenere conto, contemporaneamente, di più aspetti e relazioni che possano intervenire tra uno stesso insieme di entità. Questa necessità è motivata dall'idea secondo la quale, spesso, studiare un insieme di entità relativamente ad un unico particolare aspetto possa fornire informazioni diverse o solo parzialmente complete rispetto a quelle risultanti da uno studio più approfondito che tenga conto di come le entità si comportano su più aspetti riguardanti una specifica relazione od uno specifico contesto.

Sebbene la distinzione possa apparire meno concreta di quanto effettivamente lo sia, provo a fornire un esempio per chiarire: si consideri una rete pubblica di trasporti composta da stazioni e collegamenti tra queste. Una prima analisi basilare potrebbe essere condotta supponendo che l'aspetto studiato dalla relazione possa essere la capacità di spostarsi da una stazione all'altra, indipendentemente dal mezzo di trasporto utilizzato: in questo caso si potrebbe essere interessati a capire, ad esempio, quali siano le stazioni più frequentate nel corso della giornata in termini di passeggeri, oppure quali tra queste risultino importanti per gli scambi tra un mezzo e l'altro. Sebbene alla prima domanda si possa rispondere utilizzando questo aspetto, la seconda risulta certamente penalizzata da

una semplificazione non necessaria che abbiamo deciso di operare (quella di non tenere conto del mezzo di trasporto con cui ci si sposta da una stazione all'altra). Una struttura più completa e certamente più utile al nostro scopo potrebbe invece essere quella in cui si associano i diversi mezzi di trasporto (autobus, treno, rete suburbana, metropolitana, etc..) ad aspetti diversi, ciascuno con una propria rete, e che tenga in considerazione la possibilità di spostarsi da una rete all'altra tramite determinate stazioni: in tal caso le stazioni più centrali (ad esempio, quella ferroviaria) potrebbero risultare effettivamente degli importanti nodi scambiatori tra più mezzi di trasporti ma, per esempio, altre particolari stazioni potrebbero risultare *importanti* solo entro un certo livello di trasporti.

La necessità di studiare contemporaneamente le relazioni che intervengono entro più aspetti caratterizzanti uno stesso insieme di entità ha portato allo sviluppo di nuove strutture algebriche particolarmente complesse, con notazioni, terminologie, misure e modelli completamente diversi e spesso risultanti dall'ambito entro i quali sono stati inizialmente progettati. Senza ombra di dubbio, questo ha complicato l'interpretazione e, in taluni casi, creato confusione all'interno dei vari ambienti di ricerca, rallentandone la diffusione.

Non meno importante delle considerazioni appena fatte è anche il fatto che esistano pochi strumenti informatici in grado di gestire reti di questo tipo. Per esempio, in R non esiste un pacchetto di funzioni in grado di gestire queste strutture, di produrre le principali misure descrittive, e tantomeno i principali modelli statistici diffusi in tale ambito.

Il lavoro presentato nelle prossime pagine tenta, seppure in un primo approccio esplorativo all'argomento, di affrontare le carenze sopra menzionate, e si pone come obiettivo quello di uniformare le principali misure descrittive per analizzare reti le cui relazioni intervengono sotto diversi aspetti, e di sviluppare nell'ambiente R una serie di funzioni per gestire e descrivere tali strutture.

In particolare, il Capitolo 2 dedica l'attenzione alle reti più *semplici* la cui relazione identifica un solo aspetto, fornendo una prima illustrazione dei concetti più noti sviluppati all'interno della teoria dei grafi. Viene inoltre fatto un breve cenno ai *tensori*, che costituiranno la struttura algebrica entro la quale si svilupperà il Capitolo 3, e all'interno di questa verranno ripercorse le principali misure e i principali indici illustrati.

All'interno del Capitolo 3 vengono definite le strutture a rete che permettono di tenere in considerazione più relazioni relative a diversi aspet-

ti considerate su uno stesso insieme di entità. Viene inoltre illustrata nel dettaglio la formalizzazione proposta da De Domenico *et al.* (2013) ed, entro questa, vengono richiamate ed incluse alcune tra le principali misure sviluppate per tenere conto di tale complessità informativa.

Il Capitolo 4 propone una panoramica generale del pacchetto sviluppato, illustrandone sue funzionalità e, per ciascuna funzione, viene richiamata la misura o l'aspetto che questa intende implementare.

Nel Capitolo 5 viene condotta un'analisi esplorativa su una rete reale composta da 5 diversi livelli, utilizzando le funzioni sviluppate e proposte nel capitolo precedente. Questa analisi permette anche di illustrare il funzionamento delle principali funzioni e l'interpretazione degli output.

Il Capitolo 6 si propone di fornire alcune intuizioni su possibili sviluppi futuri del pacchetto sviluppato.

Infine, l'Appendice A riporta il codice delle funzioni sviluppate e l'Appendice B quello per visualizzare alcuni grafici illustrati nei capitoli.

Capitolo 2

Reti Monoplex

Si parla di rete *monoplex* facendo riferimento, in genere, al più noto e primitivo concetto di rete. Il termine *monoplex*, generalmente omesso, è qui specificato per sottolineare la differenza rispetto alle strutture *multiplex* che verranno presentate nei capitoli successivi: questo concetto fa riferimento alle prime strutture studiate in ambito sociometrico ed entro le quali sono stati sviluppati i più celebri strumenti descrittivi, che in seguito andrò brevemente ad illustrare.

In un contesto sociale, è possibile definire una rete *monoplex* nella sua forma più elementare come una comunità di *attori* che interagiscono tra di loro secondo una specifica *relazione*.

Col passare degli anni, poi, la vasta diffusione in molteplici ambiti di ricerca ha permesso la diffusione di una terminologia diversa che si ispira alla *teoria dei grafi* sottostante: questi termini hanno il pregio di non richiamare l'ambito sociale entro il quale la *network analysis* ha trovato le prime applicazioni e pertanto, col passare degli anni, si sono sedimentati nel vocabolario comune della letteratura in questo contesto.

All'interno della *teoria dei grafi*, quello che precedentemente ho chiamato attore prende il nome di *nodo* (o *vertice*) e la relazione tra due di questi prende il nome di *arco* (o *spigolo*): questi termini sono in qualche modo *neutri nel significato* rispetto a qualunque ambito applicativo e, per tale motivo, sono anche quelli che utilizzerò nel seguito.

In ogni caso, va comunque sottolineato che anche la notazione più utilizzata ha origine nel campo della sociometria (per approfondimenti, si veda Wasserman e Faust, 1994) e può essere estesa senza particolari restrizioni a qualunque ambito: questa utilizza il concetto di matrice come struttura elementare entro la quale codificare l'informazione circa i nodi e gli archi di una rete e, vista la sua facilità, è quella che ha preso più piede nel corso degli anni e che utilizzerò nelle prossime pagine.

2.1 Formalizzazione matematica

Un *grafo* è definibile, nella sua forma più generale, come una tupla $G = (V, E)$ dove $V = \{v_1, v_2, \dots v_N\}$ è l'insieme composto da N nodi ed $E = \{l_1, l_2, \dots l_L\}$ tale che $E \subseteq V \times V$ è l'insieme degli archi, il cui generico elemento l_k rappresenta l'arco esistente tra una coppia di nodi (v_i, v_j).

Una generica relazione stabilisce un arco tra due nodi e quest'ultima è, in genere, la più basilare *unità di misura* che si utilizza all'interno della *network analysis*. A partire dal concetto di arco è possibile definire come *diade* la coppia di nodi e tutti i possibili archi che possono intervenire tra di essi e, allo stesso modo, come *triadi* i sottoinsiemi costituiti da tre nodi e da tutti i possibili archi tra di essi. Queste ultime strutture ricorrono per la maggiore negli studi sul bilanciamento di un grafico, sulla transitività di una tripla di nodi e sulla tendenza a creare dei sottogruppi di nodi che condividono particolari caratteristiche,.

A tal proposito, è bene sottolineare fin dall'inizio una sostanziale differenza tra relazioni *direzionali* e relazioni *non-direzionali*: le prime sono relazioni in cui la coppia associata all'elemento l_k è ordinata, le seconde no. In altre parole, le relazioni direzionali sono caratterizzate da un ordine che ne stabilisce il *verso* dell'associazione (ad esempio, il nodo v_i è in relazione con il nodo v_j, ma è possibile che la relazione inversa non sia osservata - o addirittura, non esista - all'interno del contesto oggetto di studio); le relazioni non-direzionali, invece, definiscono associazioni che esistono in entrambi i sensi di percorrenza (in questo caso, se il nodo v_i è in relazione con il nodo v_j, allora necessariamente esiste anche la relazione inversa, e viceversa).

Nel caso di relazioni direzionali, una coppia di nodi (v_i, v_j) permette la misurazione di due distinte relazioni, $v_i \to v_j$ e $v_j \to v_i$; nel caso di relazioni non-direzionali, invece, una coppia di nodi (v_i, v_j) permette la misurazione di un'unica relazione $v_i \leftrightarrow v_j$. Ne consegue allora che l'insieme E conterrà al più $\binom{N}{2} = \frac{N(N-1)}{2}$ o $N(N-1)$ archi diversi a seconda che la relazione sia, rispettivamente, non-direzionale o direzionale, ovvero che il grafo sottostante la rete sia, rispettivamente, non orientato od orientato.

In un'ottica puramente matematica, le informazioni contenute negli insiemi V ed E sono rappresentabili in una matrice $\mathbf{X}$ di dimensione $N \times N$ chiamata *matrice di adiacenza*, il cui generico elemento x_{ij} è definito come:

$$x_{ij} = \begin{cases} 1 & \text{se } v_i \to v_j \\ 0 & \text{altrimenti} \end{cases} \qquad i, j = 1, 2, \dots N$$

Per quanto appena detto sulle relazioni non-direzionali, una matrice di adiacenza $\mathbf{X}$ avrà necessariamente $x_{ij} = x_{ji}$ $(\forall\ i \neq j)$ e dunque sarà simmetrica: $\mathbf{X} = \mathbf{X}^\top$. In genere, si tende ad escludere i cosiddetti *self-loop*, cioè gli archi associati alle relazioni tra un nodo e se stesso, perchè hanno poco senso: la diagonale di $\mathbf{X}$ è quindi, il più delle volte, indefinita. Si nota molto facilmente che la notazione è adattabile anche al caso di relazioni la cui forza viene misurata assegnando un valore numerico o peso: si parla in questo caso di reti discrete, o pesate.

Una seconda matrice che può essere utilizzata per rappresentare la struttura *monoplex* prende il nome di *matrice d'incidenza* $\mathbf{I}$: questa matrice avrà N righe (tanti quanti sono i nodi), L colonne (tante quante sono gli archi tra i nodi) e generico elemento I_{ij} definito come:

$$I_{ij} = \begin{cases} 1 & \text{se } v_i \text{ è incidente con } l_j \\ 0 & \text{altrimenti} \end{cases} \qquad i = 1, 2, \dots N \quad j = 1, 2, \dots L$$

Siccome il $k - esimo$ arco $l_k = (v_i, v_j)$ è incidente con i due nodi v_i e v_j, ciascuna colonna della matrice d'incidenza $\mathbf{I}$ avrà esattamente due 1 e tutti 0. Tra le due matrici, quella di adiacenza è certamente la più usata, e permette di definire i principali descrittori per reti *monoplex*.

2.2 Descrittori

Una volta formalizzata la struttura algebrica che ci consente di implementare la rete *monoplex* in chiave matematica, è possibile sviluppare alcune misure descrittive in grado di evidenziare particolari caratteristiche della rete in esame. In questo paragrafo mostrerò alcuni tra i più comuni descrittori, e la loro rappresentazione in chiave algebrica utilizzando la formalizzazione appena descritta.

Degree

Definiamo *degree* (grado) di un nodo, $d(v_i)$, il numero di archi incidenti ad esso all'interno della rete in esame. Equivalentemente, il grado di un nodo è definibile come il numero di nodi adiacenti ad esso, ovvero come la cardinalità del suo *vicinato*:

$$d(v_i) = card(V_i) = card(\{v_j \mid \exists\,(v_i, v_j) \vee \exists\,(v_j, v_i)\,, j \neq i\}).$$

Il *degree* assume un valore discreto compreso tra un minimo di 0 quando un nodo è *isolato* (cioè, non è collegato a nessun altro nodo) ad un massimo di $N-1$ se il nodo è collegato a tutti gli altri nodi presenti nella rete.

Per come è stata definita la matrice di adiacenza $\mathbf{X}$, il *degree* di un nodo può essere calcolato sommandone le colonne (o le righe):

$$d(v_i) = \sum_{j=1}^{N} x_{ij} = \sum_{i=1}^{N} x_{ij} = x_{i+} = x_{+i} \qquad i = 1, 2, \ldots N \tag{2.1}$$

Nel caso in cui la rete facesse riferimento a relazioni dirette, è necessario distinguere la nozione di *indegree* (grado interno) $d_I(v_i)$ da quella di *outdegree* (grado esterno) $d_O(v_i)$: il primo definisce il numero di archi che puntano verso un certo nodo, il secondo il numero di archi che prendono origine da un certo nodo; qualora la relazione tra due nodi dovesse essere direzionale, non per forza l'associazione che interviene dal nodo v_i al nodo v_j implica l'esistenza dell'associazione inversa, e viceversa.

Le due misure possono dunque essere definite come, rispettivamente, la somma in colonna e la somma in riga della matrice di adiacenza $\mathbf{X}$:

$$d_I(v_i) = \sum_{j=1}^{N} x_{ij} = x_{i+} \qquad i = 1, 2, \ldots N \tag{2.2}$$

e

$$d_O(v_i) = \sum_{j=1}^{N} x_{ji} = x_{+i} \qquad i = 1, 2, \ldots N \tag{2.3}$$

In un contesto puramente sociometrico, gli *outdegree* possono essere interpretati come misure di espansività, mentre gli *indegree* possono essere interpretati come misure di ricettività, o popolarità; definizioni simili possono essere date anche in altri contesti, con nomi diversi.

Media e varianza dei degree

In molte applicazioni, potrebbe risultare utile e altamente informativo riassumere in un'unica misura i *degree* dei nodi appartenenti alla rete. A tal proposito, definiamo *media dei degree* la statistica che riporta la media dei *degree* di ciascun nodo appartenente alla rete:

$$\bar{d} = \frac{\sum_{i=1}^{N} d(v_i)}{N} = \frac{2\,L}{N}. \tag{2.4}$$

Un'altra misura d'interesse potrebbe essere la variabilità dei *degree* dei nodi: in tal caso, la statistica utilizzata è la *varianza dei degree*, calcolata come:

$$S_D^2 = \frac{\sum_{i=1}^{N}(d(v_i) - \bar{d})^2}{N} \tag{2.5}$$

Ancora una volta, entrambe le misure si modificano nel caso in cui le relazioni siano dirette. In tal caso, distingueremo la *media degli indegree* dalla *media degli outdegree* ma, siccome gli *indegree* (2.2) contano gli archi incidenti dai nodi e i *outdegree* (2.3) contano gli archi incidenti ai nodi, certamente $\sum_{i=1}^{N} d_I(v_i) = \sum_{i=1}^{N} d_O(v_i) = L$ e dunque, definendo le due misure come

$$\bar{d}_I = \frac{\sum_{i=1}^{N} d_I(v_i)}{N} \quad \text{e} \quad \bar{d}_O = \frac{\sum_{i=1}^{N} d_O(v_i)}{N}$$

possiamo vedere che $\bar{d}_I = \bar{d}_O$ e semplificare le due statistiche in

$$\bar{d}_I = \bar{d}_O = \frac{L}{N}\,. \tag{2.6}$$

Allo stesso modo si definiscono la *varianza degli indegree* e la *varianza degli outdegree*:

$$S_{D_I}^2 = \frac{\sum_{i=1}^{N}(d_I(v_i) - \bar{d}_I)^2}{N} \quad \text{e} \quad S_{D_O}^2 = \frac{\sum_{i=1}^{N}(d_O(v_i) - \bar{d}_O)^2}{N} \tag{2.7}$$

Densità

Legato al concetto di nodo vi è quello di *densità*: questa è una statistica che riporta la proporzione tra il numero di archi presenti all'interno della rete ed il numero massimo che questa può contenerne. E' una misura che permette di vedere se una rete è completa o vuota, assume un valore compreso tra 0 e 1, ed è definita come:

$$\Delta = \frac{card(E)}{N(N-1)/2} = \frac{2\,card(E)}{N(N-1)} = \frac{\sum_{i=1}^{N}\sum_{j=1}^{N} x_{ij}}{N(N-1)} \tag{2.8}$$

Chiaramente, qualora le relazioni fossero direzionali, la misura si modifica in:

$$\Delta = \frac{card(E)}{N(N-1)} = \frac{\sum_{i=1}^{N}\sum_{j=1}^{N} x_{ij}}{N(N-1)}. \tag{2.9}$$

Walk

Definiamo *walk* (in italiano, passeggiata) una sequenza di nodi ed archi che inizia e finisce con nodi, in cui ciascuno è incidente con gli archi che lo precedono e seguono. Si definisce poi *lunghezza* di un *walk* il numero di archi presenti in essa: se ciascun arco è incluso più di una volta all'interno del *walk*, questo viene contato più volte.

Il *walk* è il tipo di sequenza più generale che possa essere individuata in una rete, siccome non impone alcuna restrizione su quali nodi e su quali archi debbano essere inclusi.

Utilizzando la matrice di adiacenza di una rete, si possono studiare le opzioni che un nodo ha per raggiungerne un altro, sia nei casi in cui la relazione sia direzionale sia in quelli in cui sia non-direzionale.

E' possibile studiare una *walk* di qualunque lunghezza, semplicemente studiando le potenze della matrice di adiacenza $\mathbf{X}$. In particolar modo, $\mathbf{X}^p = \underbrace{\mathbf{X} \times \mathbf{X} \times \cdots \times \mathbf{X}}_{\text{p volte}}$ avrà nella generica posizione $[\mathbf{X}^p]_{ij}$ il numero di distinti *walk* di lunghezza p che è possibile individuare partento dal nodo v_i e arrivando al nodo v_j. Gli elementi $[\mathbf{X}^p]_{ii}$ sulla diagonale definiscono invece il numero di distinti *closed walk* (in italiano, passeggiate chiuse), cioè il diverso numero di *walks* che iniziano e terminano in v_i.

E' importante notare fin da ora che, se $p = 3$, $[\mathbf{X}^3]_{ii}$ individua anche il numero di distinti *cicli* di lunghezza 3, ovvero di *closed walk* senza nodi od archi ripetuti eccetto il primo e l'ultimo: questa proprietà risulterà utile nel seguito quando si andranno a definire i coefficienti di *clustering*.

Path

Esistono particolari tipi di *walk* che soddisfano determinate proprietà. Uno di questi è sicuramente il *path* (percorso), definito come un *walk* in cui tutti i nodi e tutti gli archi sono diversi tra di loro. Se esiste un *path* tra due nodi v_i e v_j generici, allora questi si dicono *raggiungibili*; se due nodi non sono raggiungibili, dunque, significa non c'è alcun *path* entro il quale un nodo può raggiungere l'altro.

Siccome ciascun *path* è anche un *walk*, possiamo studiare la raggiungibilità di una coppia di nodi considerando le potenze della matrice di adiacenza $\mathbf{X}$. In particolare, se due nodi sono raggiungibili, allora esiste almeno un *path* (e quindi, almeno un *walk*) di lunghezza $N-1$ (o meno) tra di essi.

Un modo per determinare immediatamente se due nodi sono raggiungibili è quello di esaminare l'insieme di tutte le matrici $\mathbf{X}^p$, con $1 \leq p \leq N-1$; se due nodi sono raggiungibili, allora ci sarà certamente una cella diversa da zero in almeno una delle matrici di questo insieme. Un modo molto facile per capirlo è considerare la matrice ottenuta come la somma di tutte le matrici di adiacenza dell'insieme

$$\mathbf{X}^{[\Sigma]} = \mathbf{X} + \mathbf{X}^2 + \mathbf{X}^3 + ... + \mathbf{X}^{N-1} = \sum_{j=1}^{N-1} \mathbf{X}^j$$

la cui generica cella $[\mathbf{X}^{[\Sigma]}]_{ij}$ restituisce il numero di *walks* di qualunque lunghezza $p \leq N-1$ che è possibile identificare da v_i a v_j. Se tale cella avrà un valore uguale da 0, allora certamente v_i e v_j non sono raggiungibili perchè non connessi da nemmeno un *walk* (e dunque, da nemmeno un *path*); ancora una volta, il caso di relazioni direzionali va trattato con particolare attenzione, in relazione a come è definita la matrice di adiacenza.

Geodesiche e distanze

Si definisce *distanza* tra due nodi v_i e v_j, la lunghezza della loro *geodesica*, ovvero del più breve *path* esistente tra di essi. Ovviamente, ancora una volta bisogna subito prestare attenzione al tipo di relazione che caratterizza la rete: se questa è non-direzionale, la geodesica tra due nodi è identica; se è direzionale, invece, la geodesica tra v_i e v_j potrebbe differire da quella tra v_i e v_j. Questo perchè, come già detto, in reti con relazioni direzionali non necessariamente il *path* esistente da un nodo ad un altro implica l'esistenza dell'esatto *path* inverso.

La distanza geodesica $d(v_i, v_j)$ tra due nodi può essere allora genericamente definita come la prima potenza p per cui $[\mathbf{X}^p]_{ij} > 0$. Se non c'è alcun *path* tra due nodi, allora non esiste neanche alcuna geodesica e dunque, per convenzione, si dice che la distanza è *infinita*. Le distanze sono molto importanti in fase di analisi di una rete e sono utilizzate in alcune misure definite per studiare la centralità dei nodi.

Legato al concetto di geodesica vi è quello di *eccentricità* di un nodo: quest'ultima è definita come la massima geodesica tra il nodo ed un qualunque altro. Può assumere un valore minimo di 1 (se un nodo è adiacente a tutti gli altri nodi della rete) ad un massimo di $N-1$, e riassume quanto distante è un nodo da quello più lontano:

$$e(v_i) = \max_{j=1,2,...,N} d(v_i, v_j) \qquad i = 1, 2, \ldots, N \tag{2.10}$$

Coefficiente di clustering

Una delle principali caratteristiche delle reti *monoplex* (in particolare, delle reti collaborative e sociali) è la tendenza dei loro nodi a formare *triangoli*, ovvero cicli composti da 3 nodi: tale concetto (che in gergo tecnico viene definito transitività) è riassunto bene da Battiston *et al.* (2014) nella frase "l'amico del tuo amico è anche un mio amico".

Il concetto di *ciclo* è molto semplice, e fa riferimento ad una *walk* chiusa (cioè, che inizia e finisce dallo stesso nodo) composta da almeno 3 nodi in cui tutti gli archi e tutti i nodi intermedi sono diversi tra di loro, ovvero ad un *path* chiuso.

Una misura molto diffusa che permette di studiare questa proprietà (che in gergo tecnico viene definita *transitività*) è il coefficiente di *clustering* locale, proposto da Watts e Strogatz (1998) e definito come:

$$\begin{aligned} C(v_i) &= \frac{\sum_{j\neq i}^{N}\sum_{m\neq i}^{N}[\mathbf{X}]_{ij}[\mathbf{X}]_{jm}[\mathbf{X}]_{mi}}{\sum_{j\neq i}^{N}\sum_{m\neq i}^{N}[\mathbf{X}]_{ij}[\mathbf{X}]_{mi}} = \\ &= \frac{\sum_{j\neq i}^{N}\sum_{m\neq i}^{N}[\mathbf{X}]_{ij}[\mathbf{X}]_{jm}[\mathbf{X}]_{mi}}{d(v_i)(d(v_i)-1)} \end{aligned} \tag{2.11}$$

Tale coefficiente misura la frazione di triadi centrate in v_i (al denominatore, che si modifica in $\binom{d(v_i)}{2} = \frac{d(v_i)(d(v_i)-1)}{2}$ nel caso di relazioni non direzionali) che si chiudono in triangoli (al numeratore), e assume un valore compreso tra 0 e 1; una sua generalizzazione nel caso di reti *monoplex* con relazioni pesate è stata proposta da Barrat *et al.* (2004). In questo contesto, una statistica molto usata è la media dei coefficienti di clustering della rete *monoplex* in esame, che prende il nome di *network clustering coefficient* (Kemper (2009)), pari a $\bar{C} = \frac{1}{N}\sum_{i=1}^{N} C(v_i)$.

Una seconda misura che tiene in considerazione l'intera rete *monoplex* è quella proposta da Luce e Perry (1949) e nota come coefficiente di *clustering* globale (definita, in alcuni ambiti, anche come *indice di transitività*). Tale misura ha il pregio di poter essere applicata ad un qualunque tipo di rete *monoplex*, è definita come

$$C = \frac{3 \times \text{Numero di triangoli nel grafo}}{\text{Numero di triadi nel grafo}} \tag{2.12}$$

e trova una sua generalizzazione nel caso di reti *monoplex* pesate con l'indice proposto da Opsahl e Panzarasa (2009).

2.3 Misure di prominenza

Uno degli obiettivi principali della *network analysis*, almeno a livello esplorativo, è l'indentificazione dei nodi più importanti all'interno di una rete *monoplex*. Il gergo previsto dalla teoria dei grafi definisce, in tal caso, il concetto di *prominenza* di un nodo, che è a sua volta suddiviso nei concetti di *centralità* e di *prestigio* (quest'ultimo specifico per reti direzionali).

Il concetto di prominenza viene definito a partire da una serie di misure che, in modi ben definiti, specificano quanto importante è un nodo all'interno della rete; tali misure vengono inizialmente definite a livello dei singoli nodi, ma possono essere generalizzate a gruppi di nodi individuabili all'interno della rete sottoposta all'analisi. Anche in questo caso, gran parte delle misure di prominenza che andrò a descrivere sono state inizialmente definite in ambito sociometrico e, pertanto, tendono a rispecchiare all'interno della loro specificazione le caratteristiche e gli obiettivi che caratterizzano tale ambito di ricerca.

Una definizione di prominenza è quella data da Wasserman e Faust (1994) in ambito sociale, secondo la quale un attore viene detto *prominente* se le relazioni ad esso collegate lo rendono particolarmente visibile/popolare agli altri attori che compongono la rete. Per determinare quali tra gli N attori in un particolare gruppo sono prominenti, è dunque necessario esaminare non solo le relazioni *dirette* che prendono origine e terminano da/in un particolare attore, ma anche tutte le relazioni indirette che lo coinvolgono.

Nelle prossime pagine, utilizzerò i termini *centralità* e *prominenza* in maniera ambivalente, siccome farò riferimento in particolare a reti caratterizzate da relazioni non direzionali; le definizioni nel caso di relazioni direzionali sono pressochè analoghe e sviluppate nel dettaglio da Wasserman e Faust (1994).

Indicherò, in particolare, con la lettera C_A un particolare indice di centralità associato ad una certa misura A, che sarà funzione di uno specifico nodo $v_i \in V$. Questa sarà poi generalizzabile in una misura riassuntiva di tutta la rete *monoplex*, che permette di comparare facilmente diverse reti tra di loro, secondo determinati criteri. Tra di questi, cito l'indice proposto da Freeman (1978): definita una certa misura di centralità C_A, e stabilita $C_A(v^*) = \max_{i=1,2,\ldots,N} C_A(v_i)$, allora questo è calcolato come il rapporto tra la somma degli scarti tra $C_A(v^*)$ e $C_A(v_i)$ e la somma massima teoricamente calcolabile tra questi scarti, secondo la

formula:

$$C_A = \frac{\sum_{i=1}^{N}[C_A(v^*) - C_A(v_i)]}{\max \sum_{i=1}^{N}[C_A(v^*) - C_A(v_i)]} \,. \tag{2.13}$$

L'indice di Freeman così definito assume sempre un valore compreso tra 0 (quando tutti i nodi hanno la stessa misura di centralità: $C_A(v_i) = C_A(v_j)$, $i \neq j$) e 1 (quando un particolare nodo domina completamente - in termini di centralità - tutti gli altri).

Degree centrality

La più semplice misura di centralità è interpretata dal concetto di *degree centrality*, secondo il quale un particolare nodo è centrale per la rete se è tra i più attivi all'interno di questa, cioè se entra in un numero ben maggiore di associazioni/archi rispetto agli altri nodi. Chiaramente, la definizione di *attività* è piuttosto vaga, per quanto facilmente intuibile osservando una rappresentazione grafica della rete *monoplex*: un nodo attivo sarà un nodo che, rispetto ad altri, interagisce molto più frequentemente con altri nodi.

Come suggerisce il nome, la *degree centrality* non è altro che il *degree* del nodo stesso:

$$C_D(v_i) = d(v_i) \qquad i = 1, 2, \ldots, N \tag{2.14}$$

Siccome tale misura in qualche modo dipende dalla grandezza della rete in esame e dunque assume un valore massimo di $N-1$, ha senso operare una standardizzazione e definire di conseguenza il coefficiente di *degree centrality* standardizzato

$$C_D^{'}(v_i) = \frac{d(v_i)}{N-1} \qquad i = 1, 2, \ldots, N \tag{2.15}$$

che ha il vantaggio, rispetto a $C_D(v_i)$, di poter essere utilizzato per fare confronti tra nodi appartenenti a reti *monoplex* distinte.

L'interpretazione da dare a tale misura è identica al significato che abbiamo dato al concetto di *degree* di un nodo: in quanto misura di centralità, però, indica quali nodi sono in diretto contatto (adiacenti) ad altri nodi. I nodi caratterizzati da una *degree centrality* elevata saranno quindi in qualche modo interpretabili come nodi *cruciali* per la rete in

esame, perchè connettono tra di loro molti altri nodi; in un contesto puramente sociometrico, in cui i nodi possono essere interpretati come attori di un determinato contesto sociale, il significato da attribuire a questa misura è quella di *ego*.

A livello globale, invece, è possibile definire un indice di centralità basato sul concetto di *degree* facendo riferimento all'indice di Freeman proposto nella (2.13). Tenendo conto che in questo caso si realizza $\max \sum_{i=1}^{N} [C_D(v^*) - C_D(v_i)]$ quando il grafo è *a stella*, cioè quando tutti i nodi sono connessi ad un unico nodo centrale, ed è pari a $(N-1)(N-2)$, allora possiamo ricavare

$$C_D = \frac{\sum_{i=1}^{N} [C_D(v^*) - C_D(v_i)]}{(N-1)(N-2)} \tag{2.16}$$

come l'indice che determina quanto *centralizzati* sono i nodi della rete *multiplex*.

Una seconda misura utilizzabile per definire la centralità globale della rete basata sul concetto di *degree* è la *densità* della rete stessa, definita nella (A.17). Considerando la media dei nodi, infatti, questa può essere riscritta in questo contesto particolare come $\bar{d} = \bar{C}_D = \sum_{i=1}^{N} C_D(v_i)/N$ e, standardizzando per $N-1$, si ottiene esattamente

$$\Delta = \frac{\bar{d}}{N-1} = \frac{\bar{C}_D}{N-1} = \frac{\sum_{i=1}^{N} C_D(v_i)/N}{N-1}. \tag{2.17}$$

Betweenness centrality

Spesso, l'interazione tra due nodi non adiacenti può dipendere da altri nodi che giacciono sul *path* esistente tra i due: questi altri nodi potrebbero avere un qualche tipo di *controllo* sulle interazioni esistenti tra i due nodi non adiacenti.

Il concetto di *betweenness centrality*, inizialmente sviluppato da Freeman (1977) trova vasta applicazione nel campo della biologia, delle scienze sociali e dell'ottimizzazione di reti di trasporti, e si basa sull'intuizione secondo la quale un nodo v_i è centrale se giace sulla geodesica di molte coppie di nodi, ovvero quando assume un ruolo strategico nella rete perchè funge da tramite per molte coppie di nodi.

Definito g_{jk} come il numero di geodesiche che collegano due nodi generici v_j e v_k, allora se tutte le geodesiche possono essere ugualmente

scelte per il *path*, la probabilità che la comunicazione tra due i due nodi non adiacenti passi per una qualunque di queste è semplicemente $1/g_{jk}$. A questo punto, considerato $g_{jk}(v_i)$ come il numero di geodesiche tra v_j e v_k contenenti v_i, è possibile definire la probabilità che un generico attore v_i (sul quale calcoliamo la specifica misura di *betweenness centrality*) possa essere coinvolto in una delle geodesiche tra i due nodi come $g_{jk}(v_i)/g_{jk}$.

L'indice di *betweenness centrality* per il nodo v_i allora è semplicemente la somma di queste probabilità stimate, per tutte le coppie di nodi in $V \setminus \{v_i\}$:

$$C_B(v_i) = \sum_{j<k}^{g} \frac{g_{jk}(v_i)}{g_{jk}} \qquad i = 1, 2, \ldots, N \tag{2.18}$$

Così come le altre misure di centralità, anche in questo caso è possibile operare una standardizzazione considerando che assume un valore massimo pari a $(N-1)(N-2)/2$ (cioè, il numero di coppie di nodi che non includono v_i in un rete a relazioni non direzionali; $(N-1)(N-2)$ nel caso di rete a relazioni direzionali), e dunque definire

$$C_B^{'}(v_i) = \frac{C_B(v_i)}{[(N-1)(N-2)/2]} \qquad i = 1, 2, \ldots, N \tag{2.19}$$

Utilizzando l'indice di Freemann ed eseguendo alcune semplificazioni è possibile definire un indice di *betweenness centrality* aggregato come

$$C_B = \frac{\sum_{i=1}^{N} [C_B^{'}(v^*) - C_B^{'}(v_i)]}{N-1} \tag{2.20}$$

la cui interpretazione è simile agli indici di centralità globali proposti per le altre misure.

Closeness centrality

Le prime idee di *closeness centrality* sono state sviluppate da Sabidussi (1966), e si basano sul concetto di distanza geodesica. L'intuizione che sta alla base delle misure di *closeness centrality* è la seguente: all'aumentare della lunghezza delle geodesiche tra un nodo e tutti gli altri, la centralità del nodo considerato dovrebbe diminuire in quanto, a livello sociometrico, un attore è tanto meno centrale quanto più è distante dagli attori nodi della comunità.

Si nota immediatamente come questo tipo di centralità non dipenda solo dagli archi diretti, ma anche da quelli indiretti, sopratutto quando due nodi non sono adiacenti tra di loro.

Definita la distanza geodesica totale tra v_i e tutti gli altri nodi come $\sum_{j\neq i}^{N} d(v_i, v_j)$, allora l'indice di centralità proposto da Sabidussi (1966) è esprimibile come

$$C_C(v_i) = \left[\sum_{j=1}^{N} d(v_i, v_j)\right]^{-1} \qquad i = 1, 2, \ldots, N \tag{2.21}$$

ed è semplicemente l'inverso della somma delle distanze geodesiche tra un nodo v_i e tutti gli altri nodi in $V \setminus \{v_i\}$; tale indice assume un valore massimo pari ad $(N-1)^{-1}$ quando il nodo è adiacente a tutti gli altri.

A tal proposito, Beauchamp (1965) ha proposto una misura standardizzata dell'indice di Sabidussi, definita come

$$C'_C(v_i) = \frac{N-1}{\left[\sum_{j=1}^{N} d(v_i, v_j)\right]} = (N-1)\, C_C(v_i) \qquad i = 1, 2, \ldots, N \tag{2.22}$$

che permette il confronto tra gli indici calcolato su reti diverse.

Una misura globale di *closeness centrality* è invece ricavabile utilizzando l'indice di Freeman e considerando che il massimo valore assumibile dal numeratore è pari a $[(N-2)(N-1)]/(2N-3)$:

$$C_C = \frac{\sum_{i=1}^{N} [C'_C(v^*) - C'_C(v_i)]}{[(N-2)(N-1)]/(2N-3)} \tag{2.23}$$

Così come l'indice globale di *degree centrality* proposto precedentemente, questa assumerà il suo valore massimo di 1 quando un nodo è adiacente a tutti gli altri $N-1$ nodi (ovvero, quando ha una geodesica di lunghezza 1 con tutti gli altri nodi), e tutti gli altri nodi hanno una geodesica di lunghezza 2 su tutti i rimanenti $(N-2)$ nodi. Tale situazione è ancora una volta rappresentabile con la nozione teorica di grafo *a stella*.

Eigenvector centrality

In generale, data una generica matrice quadrata $\mathbf{A}$, un suo autovalore (*eigenvalue*) è definibile come quella costante λ che risolve l'equazione

$\mathbf{Ax} = \lambda\mathbf{x}$, dove $\mathbf{x}$ è il relativo autovettore (*eigenvector*). Ovviamente, più valori di λ possono soddisfare tale equazione, definendo lo *spettro* di $\mathbf{A}$ come l'insieme $Spec(\mathbf{A}) = \{\lambda \mid \mathbf{Ax} = \lambda\mathbf{x}\}$; se $\mathbf{A}$ è simmetrica e reale, si ha inoltre che $\lambda \in \mathbb{R}$.

Una misura di centralità che fa riferimento al concetto di autovettore associato alla matrice di adiacenza $\mathbf{X}$ della rete *monoplex* in esame è stata sviluppata da Bonacich (1972). In questo contesto, si definisce centrale un nodo v_i che presenta un valore elevato nella relativa posizione dell'autovettore associato al più elevato autovalore della matrice di adiacenza $\mathbf{X}$. L'intuizione che sta dietro lo sviluppo di questa misura può essere considerata in chiave sociale tenendo in considerazione come, in genere, un attore è tanto più importante quanto più è collegato ad altri attori altrettanto importanti all'interno della rete.

Se definiamo la *eigenvector centrality* di un generico nodo v_i come $C_E(v_i)$, questa è proporzionale (secondo una certa costante λ fissata, da determinare) alla media degli indici di *eigenvector centrality* associati agli altri nodi del suo vicinato V_i,

$$C_E(v_i) = \frac{1}{\lambda}\sum_{j \in V_i} C_E(v_j) = \frac{1}{\lambda}\sum_{j=1}^{N} [\mathbf{X}]_{ij}\, C_E(v_j) \quad i = 1, 2, \ldots, N \quad (2.24)$$

Definito allora il vettore degli indici di *eigenvector centrality* come $\mathbf{C}_E = (C_E(v_1), C_E(v_2), \ldots, C_E(v_N))$, l'equazione di cui sopra può essere riscritta in forma matriciale come $\mathbf{XC}_E = \lambda\mathbf{C}_E$, e risolta come un generico problema di ricerca di autovalori di una matrice. Siccome per definizione gli autovettori hanno norma unitaria ($\|\mathbf{C}_E\| = 1$), allora i singoli pesi $C_E(v_i)$ risultano essere una misura già standardizzata, che assume valore compreso tra 0 ed 1.

Ovviamente, $\mathbf{C}_E$ risulta essere l'autovettore associato al più elevato autovalore in $Spec(\mathbf{X})$: in questo modo, se $\mathbf{X}$ è irriducibile risulta essere valido il *Teorema di Perron-Frobenius* e dunque è garantito che $\mathbf{C}_E$ assumerà valori tutti positivi (in altre parole, ciascun singolo $C_E(v_i) > 0$).

La misura di *eigenvector centrality* definita in questo modo assegna a ciascun nodo una misura di centralità che dipende sia dal numero che dalla *qualità* delle associazioni entro cui è coinvolto. Una sua famosa applicazione riguarda l'algoritmo *Page-Rank* sviluppato da Larry Page, tramite il quale il motore di ricerca Google conta il numero di link e la loro qualità per determinare quanto importante è un determinato sito all'interno del contesto specificato dalla chiave di ricerca; maggiori

informazioni sul funzionamento di tale algoritmo sono riportate in Brin e Page (1998).

Katz centrality

Il concetto di *Katz centrality* (Katz, 1953), a differenza delle principali misure di centralità proposte, prende in considerazione il concetto di *walk* al posto di quello di *path*: in questo senso, può essere considerata come una sorta di generalizzazine (ed estensione) della *degree centrality*, e presenta il vantaggio di poter essere calcolata anche nel caso di relazioni dirette.

Una misura di *Katz centrality* calcola l'influenza relativa di un generico nodo v_i all'interno della rete *monoplex*, calcolando il numero di nodi nel vicinato V_i e, in aggiunta rispetto alla *degree centrality*, di tutti gli altri nodi della rete che connettono v_i con i nodi nel vicinato V_i. Il contributo di questi secondi nodi (detti nodi *distanti*), però, viene penalizzato da un *fattore di attenuazione* $\alpha \in [0, 1]$ fissato, elevato di una potenza d pari al grado di lontananza (ovvero, al numero di archi presenti nel *path* che congiunge v_i con il nodo *distante*).

Matematicamente, la *Katz centrality* di un generico nodo v_i è definita come

$$C_K(v_i) = \sum_{k=1}^{\infty} \sum_{j=1}^{N} \alpha^k [\mathbf{X}^k]_{ji} \quad i = 1, 2, \ldots, N \tag{2.25}$$

e sfrutta il fatto che $[\mathbf{X}^k]_{ji}$ equivale al numero di *walks* di lunghezza k individuabili tra i nodi v_i e v_j.

Un'altra forma entro la quale può essere espressa $C_K(v_i)$, e che facilita la comprensione del perchè questa può essere intesa come una generalizzazione della *eigenvector centrality*, è la seguente:

$$C_K(v_i) = \alpha \sum_{j=1}^{N} [\mathbf{X}]_{ij} \left(C_K(v_j) + 1\right) \quad i = 1, 2, \ldots, N \tag{2.26}$$

Un'ultima considerazione va fatta sul valore del fattore di attenuazione α: secondo Junker e Schreiber (2008) questo dovrà essere scelto tale per cui $\alpha < [|\max \lambda_j \in Spec(\mathbf{X})|]^{-1}$; inoltre, hanno definito la possibilità di esprimere il vettore degli indici di *Katz centrality* $\mathbf{C}_K = (C_K(v_1), C_K(v_2), \ldots, C_K(v_N))$ come

$$\mathbf{C}_K = ((\mathbf{I} - \alpha \mathbf{X}^\top)^{-1} - \mathbf{I})\mathbf{1} \tag{2.27}$$

dove $\mathbf{1}$ è il vettore composto da N 1.

2.4 Descrizione di una rete monoplex mediante la notazione indiciale

Per gli sviluppi che seguiranno nel Capitolo 3, è conveniente ripercorrere i principali concetti finora esposti utilizzando una notazione alternativa, che fa uso dei tensori e della notazione di Einstein (Einstein (1916)).

2.4.1 Cenni sulla notazione indiciale

Nella sua definizione più generale, si definisce *applicazione multilineare* su n spazi vettoriali $V_1, V_2, \ldots, V_n$ e W la funzione del tipo $f : V_1 \times V_2 \times \cdots \times V_n \longmapsto W$ che associa ad n vettori $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$, appartenenti rispettivamente agli spazi $V_1, V_2, \ldots, V_n$, un vettore $f(\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n)$ che soddisfa l'equazione

$$f(\mathbf{v}_1, \ldots, \mathbf{v}_{i-1}, \lambda\mathbf{v}_i + \mu\mathbf{v}_i^{'}, \ldots, \mathbf{v}_n) = \lambda f(\mathbf{v}_1, \ldots, \mathbf{v}_n) + \mu f(\mathbf{v}_1, \ldots, \mathbf{v}_n)$$

per ogni coppia di $\lambda, \mu \in K$, per ogni sua $i - esima$ componente.

Se lo spazio W equivale al campo K, si parla più nello specifico di *forma multilineare* ed entro questa specificazione può essere definito il concetto di tensore. In particolare, definito V uno spazio vettoriale di dimensione n su un campo K generico e V^* lo spazio vettoriale *duale* di dimensione n formato da tutti i *funzionali lineari* del tipo $f : V \longmapsto K$ (che, se $V = \mathbb{R}^N$ corrisponderanno a dei vettori riga che agiranno sui vettori colonna di V tramite prodotto scalare), è definibile *tensore* del $(h + k) - esimo$ ordine quella particolare forma multilineare

$$T : \underbrace{V \times V \times \ldots V}_{\text{h volte}} \times \underbrace{V^* \times V^* \times \cdots \times V^*}_{\text{k volte}} \longmapsto K$$

che associa ad h vettori e k covettori (i vettori riga corrispondenti ai funzionali lineari definiti nello spazio duale V^*) uno scalare in K.

Definita $\{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n\}$ la base canonica di V, è possibile definire la *base canonica duale* $\{\mathbf{e}^1, \mathbf{e}^2, \ldots, \mathbf{e}^n\}$ per V^* definita come $\mathbf{e}^i(e_j) = \delta^i_j$, ovvero il cui $i - esimo$ vettore riga avrà la $j - esima$ componente pari ad 1 solo se $i = j$, e 0 in tutti gli altri casi.

A questo punto, un generico tensore generato da h vettori e k covettori è determinato dagli n^{h+k} valori

$$\mathbf{T}^{j_1,\ldots,j_h}_{i_1,\ldots,i_k} = T(\mathbf{e}_{j_1}, \ldots, \mathbf{e}_{j_h}, \mathbf{e}^{i_1}, \ldots, \mathbf{e}^{i_k}) \qquad i_k, j_h = 1, 2, \ldots, n, \ \forall k, h$$

ciascuno dei quali è espresso rispetto alle coordinate della base canonica di V (e, di conseguenza, della duale canonica di V^*). In particolare, l'insieme dei generici $\mathbf{e}_{j_h}$ (analogamente, $\mathbf{e}^{i_k}$) appartenenti ad un singolo spazio vettoriale V identifica gli n vettori della base canonica (analogamente della base duale canonica), $\forall h, k$.

Facendo uso del *prodotto di Kronecker* (che corrisponde al prodotto esterno nel caso vettoriale), $\mathbf{e}^{i_1} \otimes \cdots \otimes \mathbf{e}^{i_k} \otimes \mathbf{e}_{j_1} \otimes \cdots \otimes \mathbf{e}_{j_h}$ indica il tensore *canonico* del $(h+k) - esimo$ ordine che vale 1 nella posizione $(\mathbf{e}_{j_1}, \ldots, \mathbf{e}_{j_h}, \mathbf{e}^{i_1}, \ldots, \mathbf{e}^{i_k})$ (i_k, j_h fissati, $\forall k, h$) e 0 in tutte le altre posizioni. In altre parole, il tensore canonico avrà coordinata 1 nella combinazione $(j_1, \ldots, j_h, i_1, \ldots, i_k)$ e 0 in tutte le altre combinazioni (sempre con i_k, j_h fissati, $\forall k, h$).

Allora, il generico tensore $\mathbf{T}$ può essere espresso, richiamando la notazione di Einstein, come combinazione lineare degli n^{h+k} tensori canonici del $(h+k) - esimo$ ordine appena definiti, secondo la relazione:

$$\begin{aligned} \mathbf{T} &= \mathbf{T}^{j_1,\ldots,j_h}_{i_1,\ldots,i_k}\, \mathbf{e}^{i_1} \otimes \cdots \otimes \mathbf{e}^{i_k} \otimes \mathbf{e}_{j_1} \otimes \cdots \otimes \mathbf{e}_{j_h} = \\ &= \sum_{k,h} \sum_{i_k=1}^{n} \sum_{j_h=1}^{n} \mathbf{T}^{j_1,\ldots,j_h}_{i_1,\ldots,i_k}\, \mathbf{e}^{i_1} \otimes \cdots \otimes \mathbf{e}^{i_k} \otimes \mathbf{e}_{j_1} \otimes \cdots \otimes \mathbf{e}_{j_h} \end{aligned} \tag{2.28}$$

con, anche in questo caso, $i_k, j_h = 1, 2, \ldots, n$, $\forall h, k$.

Per semplificare la notazione e compattare gli indici, utilizzeremo nel seguito la notazione proposta da Ricci e Levi-Civita (1900) secondo la quale un generico vettore riga $\mathbf{a} \in \mathbb{R}^N$ è dato da un *vettore covariante* $\mathbf{a}_\alpha$ e il corrispondente *vettore controvariante* $\mathbf{a}^\alpha$ è un vettore colonna appartenente allo spazio Euclideo. Onde evitare confusione, le lettere latine $i, j, \ldots$ indicheranno, ad esempio, l'$i - esimo$ vettore o il $(ij) - esimo$ tensore, mentre le lettere greche $\alpha, \beta, \ldots$ verranno utilizzate per indicare le componenti di un particolare vettore o tensore. La notazione di Ricci e Levi-Civita (1900) permette dunque di fare riferimento direttamente alla componente di un generico vettore (covariante o controvariante) e risulta particolarmente funzionale se utilizzata assieme alla notazione di Einstein per contrarre i tensori.

Un esempio molto semplice di tensore è la matrice quadrata: definito uno spazio vettoriale V di dimensione n, questa è esprimibile come un tensore del secondo ordine composto da 1 vettore ed 1 covettore, definita dalla forma bilineare

$$M : V \times V^* \longmapsto K \, .$$

Se consideriamo una generica matrice reale quadrata di dimensione 3×3 (ovvero, $dim(V) = 3$), possiamo considerare la base canonica

di $V = \mathbb{R}^3$ come $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ e la base duale di V^* come $\{\mathbf{e}^1, \mathbf{e}^2, \mathbf{e}^3\}$: ciascuno dei suoi 3^{1+1} valori espresso rispetto alle coordinate della base canonica è

$$\mathbf{M}_{i_1}^{j_1} = \mathbf{M}_i^j = M(\mathbf{e}^{i_1}, \mathbf{e}_{j_1}) = M(\mathbf{e}^i, \mathbf{e}_j) \qquad i, j = 1, 2, 3$$

Ora, utilizzando il prodotto di Kronecker, è possibile costruire il tensore canonico del secondo ordine (matrice) $\mathbf{e}^i \otimes \mathbf{e}_j$ che avrà un 1 nella combinazione (i, j) e 0 in tutte le altre combinazioni (i, j fissati), in analogia con quanto fatto. La matrice $\mathbf{M}$ potrà dunque essere scritta, utilizzando ancora una volta la notazione di Einstein, come

$$\begin{aligned}\mathbf{M} = \mathbf{M}_i^j\, \mathbf{e}^i \otimes \mathbf{e}_j = \sum_{i=1}^{3}\sum_{j=1}^{3} \mathbf{M}_i^j\, \mathbf{e}^i \otimes \mathbf{e}_j = \\ = \mathbf{M}_1^1 \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \mathbf{M}_1^2 \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \cdots + \mathbf{M}_3^3 \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}\end{aligned}$$

dove ho scritto esplicitamente i termini della sommatoria per mostrare in che modo è costruita; questo è possibile farlo perchè stiamo trattando una matrice: per tensori di ordine superiore non è possibile scrivere esplicitamente la sua scomposizione in maniera esplicita.

2.4.2 Applicazione alle reti

In questo parafrago si fa riferimento a De Domenico *et al.* (2013) cui si rimanda per approfondimenti.

Si consideri la base canonica dello spazio $\mathbb{R}^N$, $\xi = \{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_N\}$, dove $\mathbf{e}_i = (0, 0, \ldots, 1, \ldots, 0)^\top$ assume valore 1 nella posizione i, e 0 altrimenti. Per rappresentare una rete, ogni nodo v_i, $i = 1, \ldots, N$, $N \in \mathbb{N}$, è associato ad uno stato, rappresentato da un vettore canonico $\mathbf{e}_i$ della base ξ.

Siccome l'obiettivo è quello di definire una struttura in grado di mettere in relazione i nodi v_i tra di loro nella maniera più generale possibile, la scelta più naturale ricade nell'utilizzo del prodotto tensoriale (altresì noto come *prodotto di Kronecker*) e, dunque, nella definizione dello spazio creato utilizzando tale prodotto: $\mathbb{R}^N \otimes \mathbb{R}^N = \mathbb{R}^{N \times N}$.

A questo punto, è possibile definire i tensori canonici del secondo ordine $\mathbf{E}_{ij} = \mathbf{e}_i \otimes \mathbf{e}_j^\top$ (dove $i, j = 1, 2, \ldots, N$) come particolari elementi dello spazio $\mathbb{R}^{N \times N}$, e se chiamiamo w_{ij} l'intensità della relazione tra due

nodi v_i e v_j, possiamo definire il *tensore relazionale* come

$$\mathbf{W} = \sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}\,\mathbf{E}_{ij} = \sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}\,\mathbf{e}_i \otimes \mathbf{e}_j^{\top}. \tag{2.29}$$

Ovviamente, una struttura di questo genere permette di gestire relazioni non-direzionali (se w_{ij} è binario e simmetrico), direzionali (se w_{ij} è binario, e non vale necessariamente che $w_{ij} = w_{ji}$) ma anche pesate (se $w_{ij} \geq 0$).

Nel contesto delle reti *monoplex* descritte in precedenza, $\mathbf{W}$ corrisponde ad una matrice pesata di dimensione $N \times N$ che rappresenta il grafo sotteso ad una rete composta da N nodi; in altre parole, $\mathbf{W}$ equivale alla matrice di adiacenza $\mathbf{X}$ e nel contesto che andremo a trattare, tale matrice prenderà il nome di *tensore d'adiacenza*.

Il tensore d'adiacenza $\mathbf{W}$ può essere ulteriormente rappresentato come una combinazione lineare di tensori definiti nelle rispettive basi canoniche, ed utilizzando la notazione di Ricci e Levi-Civita (1900), è possibile definire la generica posizione $(\alpha\beta) - esima$ del tensore di adiacenza come:

$$\mathbf{W}_{\beta}^{\alpha} = \sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}\,\mathbf{e}^{\alpha}(i)\,\mathbf{e}_{\beta}(j) = \sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}\,\mathbf{E}_{\beta}^{\alpha}(ij). \tag{2.30}$$

Si richiamano nel seguito alcuni concetti precedentemente esposti utilizzando la notazione appena introdotta.

Degree

Definito il vettore $\mathbf{1}^{\alpha} = (1, 1, \ldots, 1)^{\top} \in \mathbb{R}^N$, e $\mathbf{U}_{\alpha}^{\beta} = \mathbf{1}^{\beta}\,\mathbf{1}_{\alpha}$ la matrice composta da tutti 1, è possibile calcolare il vettore dei *degree* come

$$\mathbf{k}_{\beta} = \mathbf{W}_{\beta}^{\alpha}\mathbf{1}_{\alpha} \tag{2.31}$$

cioè sommando tutte le colonne del tensore di adiacenza definito nella (2.30). Il *degree* del generico nodo v_i definito nella (A.12) allora può essere ricavato proiettando il vettore dei *degree* sull'$i - esimo$ vettore canonico: $k(i) = \mathbf{k}_{\beta}\,\mathbf{e}^{\beta}(i)$.

Qualora la relazione fosse direzionale, allo stesso modo è possibile definire il vettore degli *indegree* come $\mathbf{k}_{\beta} = \mathbf{W}_{\beta}^{\alpha}\mathbf{1}_{\alpha}$ e il vettore degli *outdegree* come $\mathbf{k}^{\alpha} = \mathbf{W}_{\beta}^{\alpha}\mathbf{1}^{\beta}$, e da questi ricavare l'*indegree* e l'*outdegree* del generico nodo v_i come, rispettivamente, $k_I(i) = \mathbf{k}_{\beta}\mathbf{e}^{\beta}(i)$ e $k_O(i) = \mathbf{k}^{\alpha}\mathbf{e}_{\alpha}(i)$.

Media e varianza dei degree

In maniera analoga a quanto definito nella (2.4) e nella (2.5), è possibile definire la media dei *degree* come

$$\langle \mathbf{k} \rangle = (\mathbf{U}^{\rho}_{\rho})^{-1} \mathbf{k}_{\beta} \mathbf{1}^{\beta} \tag{2.32}$$

e, definito il momento secondo $\langle \mathbf{k^2} \rangle = (\mathbf{U}^{\rho}_{\rho})^{-1} \mathbf{k}_{\beta} \mathbf{k}^{\beta}$, la varianza dei *degree* come

$$var(\mathbf{k}) = (\mathbf{U}^{\rho}_{\rho})^{-1} \mathbf{k}^{\beta}_{\beta} - (\mathbf{U}^{\rho}_{\rho})^{-2} \mathbf{k}_{\alpha} \mathbf{k}_{\beta} \mathbf{U}^{\alpha\beta}. \tag{2.33}$$

Con alcuni passaggi, è possibile anche definire i rispettivi concetti di media e varianza dei *degree* nel caso di relazione dirette definiti dalle (2.6) e (2.7), ed il concetto di densità definito nella (A.17). Va notato che $\mathbf{U}^{\rho}_{\rho} = N$ e corrisponde all'operazione di contrazione, della quale darò un breve cenno nel prossimo paragrafo.

Coefficienti di clustering

Il significato dei coefficienti di clustering è già stato accennato nel Capitolo 2, nel quale sono stati illustrati alcuni coefficienti che misurano differenti definizioni di *clustering*.

Nel contesto notazionale proposto dalla (2.30), è possibile ricavare il numero di diversi cicli di lunghezza m che iniziano e terminano in uno stesso nodo v_i considerando l'espressione

$$\mathbf{W}^{\alpha}_{\xi_1} \mathbf{W}^{\xi_1}_{\xi_2} \mathbf{W}^{\xi_2}_{\xi_3} \dots \mathbf{W}^{\xi_{m-2}}_{\xi_{m-1}} \mathbf{W}^{\xi_{m-1}}_{\beta} \mathbf{e}_{\alpha}(i) \mathbf{e}^{\beta}(i) \tag{2.34}$$

che equivale all'$i-esima$ posizione sulla diagonale di $\mathbf{X}^m$, utilizzando la notazione proposta nel Capitolo 2.

A questo punto, possiamo definire il coefficiente di *clustering* locale, in analogia con la (2.11), come

$$\begin{aligned} c(\mathbf{W}^{\alpha}_{\beta}, i) &= \frac{\mathbf{W}^{\alpha}_{\rho} \mathbf{W}^{\rho}_{\sigma} \mathbf{W}^{\sigma}_{\beta}\, \mathbf{e}_{\alpha}(i) \mathbf{e}^{\beta}(i)}{\mathbf{W}^{\alpha}_{\rho} (\mathbf{U}^{\rho}_{\sigma} - \boldsymbol{\delta}^{\rho}_{\sigma}) \mathbf{W}^{\sigma}_{\beta}\, \mathbf{e}_{\alpha}(i) \mathbf{e}^{\beta}(i)} = \\ &= \frac{\mathbf{W}^{\alpha}_{\rho} \mathbf{W}^{\rho}_{\sigma} \mathbf{W}^{\sigma}_{\beta}\, \mathbf{e}_{\alpha}(i) \mathbf{e}^{\beta}(i)}{(\mathbf{W}^{\alpha}_{\rho} \mathbf{U}^{\rho}_{\sigma} \mathbf{W}^{\sigma}_{\beta} - \mathbf{W}^{\alpha}_{\rho} \boldsymbol{\delta}^{\rho}_{\sigma} \mathbf{W}^{\sigma}_{\beta})\, \mathbf{e}_{\alpha}(i) \mathbf{e}^{\beta}(i)} \end{aligned} \tag{2.35}$$

dove $\mathbf{U}^{\rho}_{\sigma} - \boldsymbol{\delta}^{\rho}_{\sigma}$ è il tensore di adiacenza corrispondente ad una rete *monoplex* completa (privata dei *self-loop*).

Allo stesso modo, è possibile definire il coefficiente di *clustering* globale, in analogia con la (2.12), come

$$c(\mathbf{W}_{\beta}^{\alpha}) = \frac{\mathbf{W}_{\rho}^{\alpha}\mathbf{W}_{\sigma}^{\rho}\mathbf{W}_{\alpha}^{\sigma}}{\mathbf{W}_{\rho}^{\alpha}(\mathbf{U}_{\sigma}^{\rho} - \boldsymbol{\delta}_{\sigma}^{\rho})\mathbf{W}_{\alpha}^{\sigma}} = \\ = \frac{\mathbf{W}_{\rho}^{\alpha}\mathbf{W}_{\sigma}^{\rho}\mathbf{W}_{\alpha}^{\sigma}}{\mathbf{W}_{\rho}^{\alpha}\mathbf{U}_{\sigma}^{\rho}\mathbf{W}_{\alpha}^{\sigma} - \mathbf{W}_{\rho}^{\alpha}\boldsymbol{\delta}_{\sigma}^{\rho}\mathbf{W}_{\alpha}^{\sigma}} \tag{2.36}$$

dove al numeratore si considera il numero totale di cicli di lunghezza 3, e al denominatore il numero totale di cicli di lunghezza 3 in cui il secondo *passo* avviene in una rete *monoplex* completa (sviluppando i conti, è possibile dimostrare che il denominatore equivale alla somma dei *degree* moltiplicata per tale somma diminuita di 1, in analogia con la definizione data nella (2.11)).

Degree centrality

Per come è stato definito nella (2.14), il generico indice di *degree centrality* è equivalente alla forma proposta poco fa.

Eigenvector centrality

Il vettore degli indici di *eigenvector centrality* definito nella (2.24) risulta essere una soluzione della generica equazione tensoriale

$$\mathbf{W}_{\beta}^{\alpha}\mathbf{v}_{\alpha} = \lambda_{(1)}\mathbf{v}_{\beta} \tag{2.37}$$

e il generico indice di *eigenvector centrality* è ottenibile come $\mathbf{v}_{\alpha}\mathbf{e}^{\alpha}(i)$.

Katz centrality

Richiamando la formula (2.27) proposta da Junker e Schreiber (2008), questa può essere riscritta nella notazione tensoriale come

$$\mathbf{v}_{\beta} = (\boldsymbol{\delta}_{\beta}^{\alpha} - a\mathbf{W}_{\beta}^{\alpha})^{-1}\mathbf{1}_{\alpha} \tag{2.38}$$

dove $\boldsymbol{\delta}_{\beta}^{\alpha}$ rappresenta l'equivalente della matrice identità $\mathbf{I}$ sotto forma di *Delta di Kronecker*.

Capitolo 3

Reti Multiplex

I rami tradizionali della ricerca in questo ambito e i primi studi a riguardo, hanno sempre assunto che i nodi caratterizzanti la struttura a rete fossero connessi l'uno con l'altro attraverso un singolo tipo di associazione/relazione, che incapsuli in qualche modo tutte le possibili connessioni tra di essi. Va detto, però, come questa assunzione sia sempre in qualche modo una facile semplificazione della vasta complessità che caratterizza il fenomeno oggetto di studio e anzi, spesso il non tener conto dei possibili diversi aspetti potrebbe addirittura condurre a risultati sbagliati o ingannevoli.

Ad esempio: ignorare la dipendenza temporale delle relazioni può portare ad una perdita di informazione circa l'ordine temporale entro cui i contatti umani determinano la trasmissione di malattie, come mostrato in uno studio a riguardo condotto da Holme e Saramäki (2012); in particolari contesti di modellazione di reti di trasporto o comunicazione, ignorare la presenza di molteplici tipi di associazioni rende difficile tenere in considerazione la simultanea presenza e rilevanza delle varie modalità entro cui può realizzarsi la relazione tra una coppia di nodi; inoltre, non tenere conto dell'esistenza di molteplici relazioni tra nodi potrebbe alterare la topologia e le dinamiche caratterizzanti un particolare sistema, portando a sovrastimare o sottostimare proprietà cruciali dei nodi, quali per esempio la centralità rispetto a particolari criteri (De Domenico *et al.*, 2013).

A sostegno di tutto ciò, già decenni fa alcuni sociologi (tra cui Wasserman e Faust, 1994) sostenevano quanto fosse cruciale studiare sistemi sociali complessi costruendo reti sociali in grado di tener conto delle diverse tipologie di associazioni tra uno stesso insieme di individui.

Per questa ragione, si è reso sempre più essenziale oltrepassare i concetti di *monoplex networks*, cercando di sviluppare e definire strutture più complicate e più realistiche.

3.1 Reti multilayer e reti multiplex

Prima di scendere nel dettaglio della descrizione e della formalizzazione, è cruciale porre l'attenzione sulla differenza che esiste tra il termine *multilayer networks* e il caso specifico dei *multiplex networks* che andrò a trattare nelle pagine seguenti.

Kivelä *et al.* (2014) definiscono una rete *multilayer* come la più generale struttura utilizzabile per rappresentare qualunque tipo di rete: alla base di tale struttura, vi è il concetto elementare di grafo, definito dalla tupla $G = (V, E)$ e corrispondente ad una rete *monoplex*.

In aggiunta, si introducono i concetti di *livello* e di *aspetto*. Ad esempio, in una rete in cui il primo aspetto è il tipo di interazione tra due nodi e il secondo aspetto è il tempo, avremo la necessità di un insieme di livelli che definisce tutti i possibili tipi di interazione, ed un secondo insieme di livelli che include la dimensione temporale (ad esempio, determinati intervalli di tempo). Permetteremo dunque a ciascun nodo di appartenere a qualunque sottoinsieme di tali livelli, e considereremo la possibilità che possa esistere un qualunque tipo di associazione tra una coppia di nodi.

Per evitare confusione, si usa definire *livello elementare* uno qualunque tra i possibili elementi di questi insiemi di livelli, mentre il termine *livello* farà riferimento ad una combinazione di livelli elementari associati a più aspetti. Sempre in riferimento all'esempio appena accennato, sia un generico tipo di interazione tra nodi (per esempio, uno scambio di mail) che un particolare intervallo di tempo (per esempio, una determinata settimana) costituiranno dei livelli elementari; una combinazione di questi due aspetti, definirà invece un livello (ovvero, ad esempio, il numero di mail scambiate all'interno della settimana).

Formalmente, un *multilayer network* è rappresentabile come una quadrupla $M = (V_M, E_M, V, \mathbf{L})$. I primi due elementi di M costruiscono la tupla $G_M = (V_M, E_M)$ che permette di interpretare la rete *multilayer* come un grafo ai cui nodi si associa il nome del livello corrispondente. V è l'insieme di tutti i nodi osservati sulla rete, indipendentemente dal livello, e la sequenza $\mathbf{L} = \{L_a\}_{a=1}^{d}$ descrive gli insiemi di livelli elementari,

dove il generico L_a è l'insieme dei livelli elementari che fanno riferimento all'aspetto a.

Utilizzando la sequenza degli insiemi dei livelli elementari appena definita, possiamo costruire un insieme di livelli considerando tutte le possibili combinazioni di livelli elementari e assemblandole utilizzando il prodotto cartesiano $L_1 \times \cdots \times L_d$.

Per indicare se uno specifico nodo è presente in uno specifico livello oppure no, consideriamo l'insieme $V \times L_1 \times \cdots \times L_d$ contenente tutte le possibili combinazioni, e da questo ci restringiamo ad un suo sottoinsieme $V_M \subseteq V \times L_1 \times \cdots \times L_d$ che contiene solo le combinazioni in cui un nodo è effettivamente presente nel corrispondente livello: quindi, la tupla $(v, \alpha_1, \ldots, \alpha_d)$ rappresenta il nodo v sul livello $(\alpha_1, \ldots, \alpha_d)$ ottenuto come prodotto cartesiano dei livelli elementari $\alpha_1, \ldots, \alpha_d$, ciascuno dei quali fa riferimento ad un particolare aspetto. Di conseguenza, l'*edge set* $E_M \subseteq V_M \times V_M$ è l'insieme delle coppie di tutte le possibili combinazioni di nodi e livelli elementari. Ovviamente, se $d = 0$, M si riduce al caso dei *monoplex networks* e in tal caso, $V_M = V$.

Questa osservazione permette di generalizzare facilmente alcuni concetti già definiti per le reti *monoplex*: ad esempio, è possibile definire un *multilayer network* pesato considerando una funzione $w : E_M \longrightarrow \mathbb{R}$; assumiamo inoltre, anche in questo caso, che non siano ammessi i *self-loop* tra i nodi.

Dalla definizione generale, poi, è possibile restringere il campo d'interesse ai *multiplex networks* (e altrettante simili tipologie di reti), definendo questi ultimi come una sequenza di grafi $\{G_\alpha\}_{\alpha=1}^{b} = \{(V_\alpha, E_\alpha)\}_{\alpha=1}^{b}$ dove $E_\alpha \subset V_\alpha \times V_\alpha$ è l'insieme degli *edges* e α indicizza i grafi. In genere, gli insiemi di nodi sono gli stessi in tutti i livelli (ovvero, $V_\alpha = V_\beta \, , \forall \alpha \neq \beta$) o, al più, questi ultimi condividono determinati nodi (cioè $\bigcap_{\alpha=1}^{b} V_\alpha \neq \emptyset$).

In altre parole, è possibile distinguere il concetto di rete *multilayer* da quello di rete *multiplex* sottolineando la seguente importante distinzione: nei sistemi *multilayer*, ciascun nodo ha una controparte in ciascun livello, cosicchè rimane sempre possibile associare un vettore di stati a ciascun nodo e tenere conto delle relazioni che intervengono tra un nodo appartenente ad un certo livello ed un altro nodo appartenente ad un secondo livello considerato nella rete; nei sistemi *multiplex*, invece, rimane ancora vero che ciascun nodo possiede la propria controparte in ciascun livello: in questo, però, le relazioni ammesse tra due livelli devono riguardare necessariamente uno stesso nodo.

In una rete *multiplex*, sono ammesse tutte le possibili relazioni intra-

livello (*intralayer connections*) (ovvero, come nel caso *monoplex*, tutte le possibili relazioni tra i nodi definiti entro uno stesso livello), mentre sono ammesse le sole relazioni inter-livello (*interlayer connections*) tra le stesse controparti di un nodo definite in ciascun livello. Le connessioni tra un nodo e la sua controparte possono avere diversi pesi per diverse coppie di livelli, e allo stesso modo le relazioni inter-livello possono differire a seconda delle distinte entità della rete.

Va poi sottolineato come, in molte applicazioni reali quali, ad esempio, le reti di trasporti, uno stesso nodo possa appartenere a più livelli (ad esempio, la rete della metropolitana e quella degli autobus) simultaneamente. In questi casi, una rete che non tiene conto delle reazioni inter-livello tra livelli non riuscirebbe a catturare l'intera struttura della rete in esame, perchè mancherebbe l'informazione circa il *costo* (in termini di tempo, o economico) per spostarsi da una rete all'altra (ovvero, tra uno stesso nodo da un livello all'altro).

3.2 Formalizzazione matematica

Per rappresentare e descrivere le reti *multilayer* (e il caso particolare delle reti *multiplex*), farò riferimento alla formalizzazione matematica introdotta da De Domenico *et al.* (2013), diretta estensione del caso *monoplex* presentato nel paragrafo 2.4. Tale rappresentazione sfrutta degli strumenti algebrici che prendono il nome di *tensori* e che, intuitivamente, possono essere interpretati come *matrici di matrici*. Rispetto alla formalizzazione definita per le reti *monoplex*, nel generalizzarla adeguatamente per il contesto *multilayer* (e *multiplex*) dobbiamo in qualche modo tenere conto della possibilità che esistano più tipologie di relazioni tra una generica coppia di nodi (v_i, v_j).

Utilizziamo il termine *intralayer adiacency tensor* (in italiano, tensore di adiacenza intra-livello) per definire il tensore del secondo ordine $\mathbf{W}^{\alpha}_{\beta}(\widetilde{k})$ che indica le relazioni intercorrenti tra i nodi di uno stesso livello $\widetilde{k}$, con $\alpha, \beta = 1, 2, \dots N$, in analogia con quanto definito nella (2.30). Quando non si specificano i valori di α e β, si intende considerare l'intero *intralayer adiacency tensor*: gli indici vengono comunque lasciati perchè, come vedremo, consentono di utilizzare la notazione di Einstein che permette di raggruppare i simboli di sommatoria, semplificando la notazione che andremo ad utilizzare.

Per considerare invece tutta l'informazione relativa alle relazioni che intervengono tra due nodi appartenenti a livelli diversi, definiamo poi

l' *interlayer adjacency tensor* (in italiano, tensore di adiacenza inter-livello) $\mathbf{C}^{\alpha}_{\beta}(\widetilde{h}\widetilde{k})$ come il tensore del secondo ordine che rappresenta l'insieme di tutti i possibili archi tra i due livelli $\widetilde{h}$ e $\widetilde{k}$. Si nota immediatamente che $\mathbf{C}^{\alpha}_{\beta}(\widetilde{k}\widetilde{k}) = \mathbf{W}^{\alpha}_{\beta}(\widetilde{k})$, ovvero che il tensore di adiacenza inter-livello corrispondente al caso in cui la coppia di livelli rappresenta lo stesso livello $\widetilde{k}$ è equivalente al tensore di adiacenza intra-livello del livello stesso.

Similmente a quanto fatto nella definizione del tensore di adiacenza per le reti *monoplex*, (equazione 2.30), introduciamo il vettore $\mathbf{e}^{\widetilde{\gamma}}(\widetilde{k})$ (dove $\widetilde{\gamma}, \widetilde{k} = 1, 2, \ldots, L$) della base canonica nello spazio $\mathbb{R}^L$: anche in questo caso la lettera greca indica la componente del vettore, mentre la lettera latina indica il $k-esimo$ vettore canonico; il simbolo di tilde ci permette di distinguere questi indici da quelli corrispondenti ai nodi.

E' possibile a questo punto definire i tensori canonici del secondo ordine

$$\mathbf{E}^{\widetilde{\gamma}}_{\widetilde{\delta}}(\widetilde{h}\widetilde{k}) = \mathbf{e}^{\widetilde{\gamma}}(\widetilde{h})\, \mathbf{e}_{\widetilde{\delta}}(\widetilde{k}) \tag{3.1}$$

che rappresentano le basi canoniche dello spazio $\mathbb{R}^{L\times L}$ e, da questi, definire il *multilayer adjacency tensor* utilizzando il prodotto tensoriale tra il tensore di adiacenza inter-livello $\mathbf{C}^{\alpha}_{\beta}(\widetilde{h}\widetilde{k})$ e i tensori canonici $\mathbf{E}^{\widetilde{\gamma}}_{\widetilde{\delta}}(\widetilde{h}\widetilde{k})$, ottenendo il seguente tensore del quarto ordine:

$$\mathbf{M}^{\alpha\,\widetilde{\gamma}}_{\beta\,\widetilde{\delta}} = \sum_{\widetilde{h}=1}^{L}\sum_{\widetilde{k}=1}^{L} \mathbf{C}^{\alpha}_{\beta}(\widetilde{h}\widetilde{k})\, \mathbf{E}^{\widetilde{\gamma}}_{\widetilde{\delta}}(\widetilde{h}\widetilde{k}) \tag{3.2}$$

Il *multilayer adjacency tensor* appena definito può essere semplicemente inteso (abusandone nella sua definizione strettamente formale in campo algebrico) come una matrice di ordine superiore al secondo, in cui ciascuna sua cella è indicizzata da 4 indici anzichè 2.

Per quanto definito sul significato del generico tensore di adiacenza inter-livello $\mathbf{C}^{\alpha}_{\beta}(\widetilde{h}\widetilde{k})$ quando $\widetilde{h} = \widetilde{k}$, questo può essere ulteriormente riscritto come

$$\mathbf{C}^{\alpha}_{\beta}(\widetilde{h}\widetilde{k}) = \sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}(\widetilde{h}\widetilde{k})\, \mathbf{E}^{\alpha}_{\beta}(ij) \tag{3.3}$$

dove $w_{ij}(\widetilde{h}\widetilde{k}) \in \mathbb{R}$ indica l'intensità della relazione tra il nodo v_i nel livello $\widetilde{h}$ e il nodo v_j nel livello $\widetilde{k}$.

Allora, definendo il tensore del quarto ordine appartenente alla base canonica nello spazio $\mathbb{R}^{N\times N\times L\times L}$

$$\begin{aligned}\boldsymbol{\xi}^{\alpha\widetilde{\gamma}}_{\beta\widetilde{\delta}}(ij\widetilde{h}\widetilde{k}) &= \mathbf{E}^{\alpha}_{\beta}(ij)\,\mathbf{E}^{\widetilde{\gamma}}_{\widetilde{\delta}}(\widetilde{h}\widetilde{k}) = \\ &= \mathbf{e}^{\alpha}(i)\mathbf{e}_{\beta}(j)\,\mathbf{e}^{\widetilde{\gamma}}(\widetilde{h})\mathbf{e}_{\widetilde{\delta}}(\widetilde{k})\end{aligned} \tag{3.4}$$

e sostituendo le espressioni ottenute per $\mathbf{C}^{\alpha}_{\beta}(\widetilde{h}\widetilde{k})$ (3.3) e $\boldsymbol{\xi}^{\alpha\widetilde{\gamma}}_{\beta\widetilde{\delta}}(ij\widetilde{h}\widetilde{k})$ (3.4) all'interno della (3.2), si ottiene l'espressione generale del *multilayer adjacency tensor* come:

$$\begin{aligned}\mathbf{M}^{\alpha\,\widetilde{\gamma}}_{\beta\,\widetilde{\delta}} &= \sum_{\widetilde{h}=1}^{L}\sum_{\widetilde{k}=1}^{L}\left[\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}(\widetilde{h}\widetilde{k})\,\mathbf{E}^{\alpha}_{\beta}(ij)\right]\mathbf{E}^{\widetilde{\gamma}}_{\widetilde{\delta}}(\widetilde{h}\widetilde{k}) = \\ &= \sum_{\widetilde{h}=1}^{L}\sum_{\widetilde{k}=1}^{L}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}(\widetilde{h}\widetilde{k})\,\boldsymbol{\xi}^{\alpha\widetilde{\gamma}}_{\beta\widetilde{\delta}}(ij\widetilde{h}\widetilde{k}) = \\ &= \sum_{\widetilde{h},\widetilde{k}=1}^{L}\sum_{i,j=1}^{N} w_{ij}(\widetilde{h}\widetilde{k})\,\boldsymbol{\xi}^{\alpha\widetilde{\gamma}}_{\beta\widetilde{\delta}}(ij\widetilde{h}\widetilde{k}).\end{aligned} \tag{3.5}$$

Il *multilayer adjacency tensor* $\mathbf{M}^{\alpha\,\widetilde{\gamma}}_{\beta\,\widetilde{\delta}}$ è un oggetto molto flessibile, che può essere utilizzato per rappresentare a livello algebrico una vasta gamma di relazioni (più o meno complicate) tra nodi, nell'ottica di quanto definito da Kivelä *et al.* (2014) e precedentemente esposto sulle reti *multilayer*.

Utilizzando la definizione del tensore di adiacenza *multilayer* data dalla (3.3), restringeremo la sua definizione al caso di un *multiplex adjacency tensor* (in italiano, tensore di adiacenza *multiplex*) imponendo che tutti i tensori di adiacenza inter-livello associati, $\mathbf{C}^{\alpha}_{\beta}(\widetilde{h}\widetilde{k})$, siano diagonali: questo limita l'esistenza delle relazioni tra coppie di nodi diversi al solo caso in cui questi siano l'uno la controparte dell'altro in entrambi i livelli $\widetilde{h}$ e $\widetilde{k}$.

Uno dei punti di forza del *multilayer adjacency tensor* $\mathbf{M}^{\alpha\,\widetilde{\gamma}}_{\beta\,\widetilde{\delta}}$ definito dalla (3.5) è la possibilità di poterlo rappresentare come uno speciale tensore del secondo ordine: questo è possibile tramite un processo definito di *matricizzazione* (noto anche con il termine di *unfolding* o *flattening*) che permette di rappresentare gli elementi di $\mathbf{M}^{\alpha\,\widetilde{\gamma}}_{\beta\,\widetilde{\delta}}$ definiti nello spazio $\mathbb{R}^{N\times N\times L\times L}$ come una matrice di dimensione $N^2\times L^2$, oppure $NL\times NL$.

L'operazione di matricizzazione di un *multilayer adjacency tensor* può essere particolarmente utile, sopratutto nelle implementazioni di

tale struttura all'interno di algoritmi computazionali: alcune particolari scomposizioni che permettono l'implementazione di tale operazione, e relative applicazioni, vengono riportate nel lavoro svolto da Kolda e Bader (2009).

3.3 Operazioni algebriche

Prima di descrivere alcune generalizzazioni dei descrittori e delle misure di prominenza illustrati nel Capitolo 2, è necessario richiamare alcune operazioni algebriche che possono essere utilizzare per estrarre informazioni utili dal *multilayer adjacency tensor* $\mathbf{M}_{\beta\,\widetilde{\delta}}^{\alpha\,\widetilde{\gamma}}$.

Contrazione

L'operazione di contrazione riduce l'ordine di un tensore di 2 dimensioni, e nella notazione *di Einstein* viene definita ponendo uguali gli indici covarianti e controvarianti del tensore.

Questa permette, ad esempio, di ottenere il numero di nodi della struttura *multiplex* come $N = \boldsymbol{\delta}_{\alpha}^{\alpha} = \boldsymbol{\delta}_{1}^{1} + \boldsymbol{\delta}_{2}^{2} + \cdots + \boldsymbol{\delta}_{N}^{N}$ oppure, nel caso di relazioni non pesate, il numero di archi calcolando $\mathbf{W}_{\beta}^{\alpha}\mathbf{U}_{\alpha}^{\beta}$.

Estrazione di livelli singoli

In alcune applicazioni, potrebbe risultare utile *estrarre* uno specifico livello dalla rete *multilayer* (ad esempio, il livello $\widetilde{r} - esimo$).

Utilizzando l'algebra tensoriale, questa operazione è equivalente al proiettare il tensore di adiacenza *multilayer* $\mathbf{M}_{\beta\,\widetilde{\delta}}^{\alpha\,\widetilde{\gamma}}$ sul tensore canonico del secondo ordine $\mathbf{E}_{\widetilde{\delta}}^{\widetilde{\gamma}}(\widetilde{r}\widetilde{r})$ definito nella (3.1) corrispondente al particolare livello d'interesse.

Il generico tensore canonico del secondo ordine è generato dai vettori canonici che compongono una base ortonormale di $\mathbb{R}^{L\times L}$, perciò il prodotto $\mathbf{E}_{\widetilde{\delta}}^{\widetilde{\gamma}}(\widetilde{h}\widetilde{k})\,\mathbf{E}_{\widetilde{\gamma}}^{\widetilde{\delta}}(\widetilde{r}\widetilde{r})$ sarà definito come segue:

$$\mathbf{E}_{\widetilde{\delta}}^{\widetilde{\gamma}}(\widetilde{h}\widetilde{k})\,\mathbf{E}_{\widetilde{\gamma}}^{\widetilde{\delta}}(\widetilde{r}\widetilde{r}) = \begin{cases} 1 & \text{se } \widetilde{h} = \widetilde{k} = \widetilde{r} \\ 0 & \text{altrimenti} \end{cases}. \tag{3.6}$$

Allora, richiamando la (3.2), avremo che

$$\begin{aligned}\mathbf{M}_{\beta\,\widetilde{\delta}}^{\alpha\,\widetilde{\gamma}}\,\mathbf{E}_{\widetilde{\gamma}}^{\widetilde{\delta}}(\widetilde{r}\widetilde{r}) &= \left[\sum_{\widetilde{h}=1}^{L}\sum_{\widetilde{k}=1}^{L}\mathbf{C}_{\beta}^{\alpha}(\widetilde{h}\widetilde{k})\,\mathbf{E}_{\widetilde{\delta}}^{\widetilde{\gamma}}(\widetilde{h}\widetilde{k})\right]\mathbf{E}_{\widetilde{\gamma}}^{\widetilde{\delta}}(\widetilde{r}\widetilde{r}) = \\ &= \sum_{\widetilde{h}=1}^{L}\sum_{\widetilde{k}=1}^{L}\mathbf{C}_{\beta}^{\alpha}(\widetilde{h}\widetilde{k})\,\mathbf{E}_{\widetilde{\delta}}^{\widetilde{\gamma}}(\widetilde{h}\widetilde{k})\,\mathbf{E}_{\widetilde{\gamma}}^{\widetilde{\delta}}(\widetilde{r}\widetilde{r}) = \\ &= \mathbf{C}_{\beta}^{\alpha}(\widetilde{r}\widetilde{r}) = \mathbf{W}_{\beta}^{\alpha}(\widetilde{r})\end{aligned} \tag{3.7}$$

che è proprio il tensore di adiacenza corrispondente al livello $\widetilde{r}-esimo$.

Projected monoplex network

In alcuni casi, si rende necessario costruire una rete *monoplex* aggregando l'intera *rete multiplex*: ad esempio, nello studio di reti temporali, in genere una delle prime operazioni che si svolge riguarda l'aggregazione dell'intera rete in base all'aspetto temporale.

La proiezione avviene contraendo il *multilayer adjacency tensor* con il tensore del secondo ordine $\mathbf{U}_{\widetilde{\gamma}}^{\widetilde{\delta}}$, e permette di ottenere il *projected monoplex network* come

$$\mathbf{P}_{\beta}^{\alpha} = \mathbf{M}_{\beta\,\widetilde{\delta}}^{\alpha\,\widetilde{\gamma}}\mathbf{U}_{\widetilde{\gamma}}^{\widetilde{\delta}} = \sum_{\widetilde{h}=1}^{L}\sum_{\widetilde{k}=1}^{L}\mathbf{C}_{\beta}^{\alpha}(\widetilde{h}\widetilde{k}) \tag{3.8}$$

Overlay monoplex network

Una struttura simile al *projected monoplex network* definito nella (3.8) è l'*overlay monoplex network*: quest'ultima, a differenza della precedente, ignora i contributi ai pesi delle relazioni inter-livello (ovvero, ignora i tensori di adiacenza inter-livello).

L'*overlay network* (noto anche col nome di *aggregated network*) si ottiene tramite un'operazione di contrazione del tensore di adiacenza *multilayer* nel modo seguente:

$$\mathbf{O}_{\beta}^{\alpha} = \mathbf{M}_{\beta\widetilde{\gamma}}^{\alpha\widetilde{\gamma}} = \sum_{\widetilde{r}=1}^{L}\mathbf{W}_{\beta}^{\alpha}(\widetilde{r}) \tag{3.9}$$

3.4 Descrittori

La generalizzazione al caso *multiplex* dei descrittori e degli indici di centralità definiti per le reti *monoplex* non è immediata, e per ciascuna di tali misure sono stati proposti, nel corso degli anni, diversi indici che andrò ad illustrare nel seguito.

Una prima doverosa premessa a tale riguardo, valida soprattutto nel caso delle reti *multiplex*, è la seguente: generalizzare i principali descrittori ed indici di centralità applicando le misure proposte nel Capitolo 2 all'*overlay monoplex network* (3.9) ottenuto aggregando *verticalmente* i livelli che lo compongono, risulta spesso un primo approccio semplice e facilmente attuabile; allo stesso modo, però, va riconosciuto come questo approccio possa condurre a risultati diversi dalla realtà quando le relazioni intra-livello assumono pesi e valori diversi: uno studio in tale senso è quello proposto da Battiston *et al.* (2014).

Molti dei descrittori definiti sui nodi delle reti *multiplex* inoltre, risultano essere delle semplici applicazioni delle misure proposte nel Capitolo 2 *livello per livello* o, in forma matriciale, secondo quanto proposto dalla (2.31). Altre opzioni saranno discusse nel seguito in questo capitolo.

Gradi e distribuzioni intra-layer

Potrebbe risultare utile, in questo caso, studiare la distribuzione dei vettori dei gradi in tutti i livelli considerati nel *multiplex network*: potrebbero infatti esserci nodi che hanno un grado elevato solo in certi livelli, oppure altri che mantengono un *degree* costante in tutti livelli considerati. A tal proposito, Battiston *et al.* (2014) hanno proposto due quantità che permettono di quantificare l'omogeneità della distribuzione dei *degree* per un generico nodo v_i all'interno dei vari livelli.

La prima quantità è una misura di entropia, ed è definita come

$$\begin{aligned} H(v_i) &= -\sum_{\widetilde{r}=1}^{L} \frac{\mathbf{W}_{\beta}^{\alpha}(\widetilde{r})\mathbf{1}_{\alpha}\mathbf{e}^{\beta}(i)}{\mathbf{O}_{\beta}^{\alpha}\mathbf{1}_{\alpha}\mathbf{e}^{\beta}(i)} \, ln\left(\frac{\mathbf{W}_{\beta}^{\alpha}(\widetilde{r})\mathbf{1}_{\alpha}\mathbf{e}^{\beta}(i)}{\mathbf{O}_{\beta}^{\alpha}\mathbf{1}_{\alpha}\mathbf{e}^{\beta}(i)}\right) = \\ &= -\sum_{\widetilde{r}=1}^{L} \frac{k(i)^{[\widetilde{r}]}}{o(i)} \, \ln\left(\frac{k(i)^{[\widetilde{r}]}}{o(i)}\right) \qquad \forall i = 1, 2, \ldots, N \end{aligned} \tag{3.10}$$

dove $k(i)^{[\widetilde{r}]}$ è il grado del nodo v_i sul livello $\widetilde{r} - esimo$, e $o(i)$ è il grado del nodo v_i calcolato sull'*overlay monoplex network* definito in (3.9). Questa quantità assume un valore minimo di 0 quando tutti gli archi

relativi al nodo v_i giacciono su un unico livello, mentre assume il suo valore massimo quando tali archi sono uniformemente distribuiti su tutti gli L livelli della rete *multiplex*.

Una seconda quantità, simile all'entropia, è il *coefficiente di partecipazione multiplo*, definito come

$$\begin{aligned} P(v_i) &= \frac{L}{L-1}\left[1 - \sum_{\widetilde{r}=1}^{L}\left(\frac{\mathbf{W}_{\beta}^{\alpha}(\widetilde{r})\mathbf{1}_{\alpha}\mathbf{e}^{\beta}(i)}{\mathbf{O}_{\beta}^{\alpha}\mathbf{1}_{\alpha}\mathbf{e}^{\beta}(i)}\right)^2\right] = \\ &= \frac{L}{L-1}\left[1 - \sum_{\widetilde{r}=1}^{L}\left(\frac{k(i)^{[\widetilde{r}]}}{o(i)}\right)^2\right] \qquad \forall i = 1, 2, \ldots, N. \end{aligned} \tag{3.11}$$

Esso assume valori compresi tra 0 e 1, rispettivamente quando tutti gli archi che fanno riferimento a v_i giacciono su un unico stesso livello e quando v_i ha esattamente lo stesso numero di archi in ciascuno degli L livelli della rete *multiplex*. Sulla base del valore assunto da $P(v_i)$, possiamo distinguere il generico nodo come *focused* (se $0 \leq P(v_i) \leq 1/3$), *mixed* (se $1/3 < P(v_i) \leq 2/3$) o *truly multiplex* (se $P(v_i) > 2/3$).

Media e varianza dei degree

I concetti di media e varianza dei *degree* definiti per le reti *monoplex* dalla (2.4) e dalla (2.5) possono essere estesi al caso *multiplex* considerando, rispettivamente, un vettore contenente le medie dei *degree* ed un vettore contenente le varianze dei *degree* calcolate separatamente su ciascun livello.

Una seconda opzione, invece, è quella di calcolare la media dei *degree* dell'*overlay monoplex network* definito dalla (3.9): questa soluzione risulta spesso opportuna quando le relazioni inter-livello sono binarie o, in generale, tutte uguali.

Densità

Così come per la media e la varianza dei *gradi*, il concetto di densità può essere esteso alle reti *multiplex* considerando un vettore contenente le densità calcolate separatamente su ciascuno dei livelli.

Walk

Il concetto di *walk* all'interno del contesto *multiplex* risulta da una generalizzazione dello stesso concetto già proposto per le reti *monoplex*. In

aggiunta al primo caso, però, le reti *multiplex* permettono la possibilità di spostarsi da un livello all'altro se è presente la relazione inter-livello tra uno stesso nodo (assumerò nelle prossime pagine che questa non esista quando il nodo è isolato in almeno uno dei due livelli); non esiste una convenzione ampiamente riconosciuta, ma il più delle volte si tende ad assumere che la stessa relazione inter-livello (e dunque, l'arco sotteso) costituisca a tutti gli effetti un passo della passeggiata, a maggior ragione quando tale relazione è pesata anzichè binaria.

Path

Nel caso dei *multiplex networks*, un *path* può essere definito come una sequenza ordinata di nodi ed archi distinti, che inizia da un nodo v_i appartenente al livello $\widetilde{h} - esimo$ e termina nel nodo v_j appartenente al livello $\widetilde{k} - esimo$.

Anche in questo caso, però, è necessario porre un'ulteriore restrizione rispetto alla definizione data nel caso *monoplex*: ciascun arco appartenente al *path* deve esistere anche tra una coppia di stessi nodi appartenenti a due differenti livelli (quelle che, in gergo, prendono il nome di relazioni intra-livello).

Definito $\mathcal{P}_{n_i\widetilde{h} \to n_j\widetilde{k}}$ l'insieme di tutti i possibili *path* tra un nodo n_i appartenente al livello $\widetilde{h} - esimo$ e un nodo n_j appartenente al livello $\widetilde{k} - esimo$, allora un *path* sarà uno qualunque tra i suoi elementi, $p_{n_i\widetilde{h} \to n_j\widetilde{k}}$.

Geodesiche e distanze

Per ciascun *path*, è possibile associare una funzione di distanza $d(p_{n_i\widetilde{h} \to n_j\widetilde{k}})$ che conta il numero di archi attraversati dal *path* misurato, come per le reti *monoplex*; anche in questo caso, vale l'accorgimento adottato per i *walk* secondo il quale l'arco sotteso alla relazione inter-livello tra due nodi costituisca a tutti gli effetti un passo del *path*.

A questo punto, si definisce l'insieme delle *geodesiche* tra due nodi (ovvero, dei *path* con distanza minima) come l'insieme di tutti i *path* che minimizzano la funzione di distanza tra di essi:

$$P^*_{n_i \to n_j} = \underset{\substack{p_{n_i\widetilde{h} \to n_j\widetilde{k}} \in \mathcal{P}_{n_i\widetilde{h} \to n_j\widetilde{k}} \\ \widetilde{h},\widetilde{k}=1,2,\ldots,L}}{\operatorname{argmin}} d(p_{n_i\widetilde{h} \to n_j\widetilde{k}}) \,. \tag{3.12}$$

In altre parole, una *geodesica* tra due generici nodi v_i e v_j in una rete *multiplex* è il *path* a distanza minima che prende origine dal nodo

v_i in un qualunque livello e termina nel nodo v_j in un qualunque livello. Questa definizione è coerente con la definizione che abbiamo dato di *multiplex network*, dato che lo stesso nodo rappresenta la stessa entità in ciascun livello della rete considerata.

Coefficienti di clustering

Nelle reti *multiplex*, non è immediato definire i coefficienti di *clustering* utilizzando i triangoli come misure di transitività in analogia a quanto fatto nella (2.11). In queste strutture, infatti, un *ciclo* di 3 nodi centrato nel nodo v_i (triangolo) potrebbe non esistere su uno stesso livello, ma la transitività potrebbe ancora emergere tenendo in considerazione più livelli assieme.

Definiamo allora un *2-triangolo* come un triangolo formato da 1 arco appartenente ad un livello $\widetilde{r}$ e dagli altri 2 archi appartenenti ad un secondo livello $\widetilde{t}$; un *3-triangolo* come un triangolo formato da 3 archi appartenenti a 3 livelli diversi; una *1-triade* centrata in v_i come una triade in cui entrambi gli archi giacciono sullo stesso livello; una *2-triade* centrata in v_i come una triade in cui i due archi giacciono su 2 livelli diversi. A questo punto, Battiston *et al.* (2014) propongono due indici di *clustering* locale per reti *multiplex*.

Il primo indice è definito, per ciascun nodo v_i, come rapporto tra il numero di 2-triangoli che contengono v_i ed il numero di 1-triadi centrate in v_i

$$\begin{aligned} C_1(v_i) &= \frac{\sum_{\widetilde{r}=1}^{L} \sum_{\widetilde{t}=1,\, \widetilde{t}\neq\widetilde{r}}^{L} \mathbf{W}_{\rho}^{\alpha}(\widetilde{r}) \mathbf{W}_{\sigma}^{\rho}(\widetilde{t}) \mathbf{W}_{\beta}^{\sigma}(\widetilde{r})\, \mathbf{e}_{\alpha}(i) \mathbf{e}^{\beta}(i)}{(L-1) \sum_{\widetilde{r}=1}^{L} \mathbf{W}_{\rho}^{\alpha}(\widetilde{r}) (\mathbf{U}_{\sigma}^{\rho} - \boldsymbol{\delta}_{\sigma}^{\rho}) \mathbf{W}_{\beta}^{\sigma}(\widetilde{r})\, \mathbf{e}_{\alpha}(i) \mathbf{e}^{\beta}(i)} = \\ &= \frac{\sum_{\widetilde{r}=1}^{L} \sum_{\widetilde{t}=1}^{L} \mathbf{W}_{\rho}^{\alpha}(\widetilde{r}) \mathbf{W}_{\sigma}^{\rho}(\widetilde{t}) \mathbf{W}_{\beta}^{\sigma}(\widetilde{r})\, \mathbf{e}_{\alpha}(i) \mathbf{e}^{\beta}(i)}{(L-1) \sum_{\widetilde{r}=1}^{L} (k(i)^{[\widetilde{r}]} (k(i)^{[\widetilde{r}]} - 1)} \end{aligned} \tag{3.13}$$

normalizzato per $(L-1)$ per permettere di fare confronti con altre reti *multiplex*.

Il secondo indice, invece, è definito per ciascun nodo v_i come rapporto tra il numero di 3-triangoli contenenti v_i ed il numero di 2-triadi centrate in v_i

$$C_2(v_i) = \frac{\sum_{\widetilde{r}=1}^{L} \sum_{\widetilde{t}=1,\, \widetilde{t}\neq\widetilde{r}}^{L} \sum_{\widetilde{q}=1,\, \widetilde{q}\neq\widetilde{t}\neq\widetilde{r}}^{L} \mathbf{W}_{\rho}^{\alpha}(\widetilde{r}) \mathbf{W}_{\sigma}^{\rho}(\widetilde{t}) \mathbf{W}_{\beta}^{\sigma}(\widetilde{q})\, \mathbf{e}_{\alpha}(i) \mathbf{e}^{\beta}(i)}{(L-2) \sum_{\widetilde{r}=1}^{L} \sum_{\widetilde{t}=1,\, \widetilde{t}\neq\widetilde{r}}^{L} \mathbf{W}_{\rho}^{\alpha}(\widetilde{r}) (\mathbf{U}_{\sigma}^{\rho} - \boldsymbol{\delta}_{\sigma}^{\rho}) \mathbf{W}_{\beta}^{\sigma}(\widetilde{t})\, \mathbf{e}_{\alpha}(i) \mathbf{e}^{\beta}(i)} \tag{3.14}$$

dove, in questo caso, la normalizzazione avviene dividendo per $(L-2)$; a differenza del precedente, $C_2(v_i)$ può essere utilizzato solo per reti *multiplex* con $L \geq 3$.

Allo stesso modo, è possibile generalizzare la definizione di coefficiente di clustering globale definito nella (2.12) e ripreso nella (2.36) definendo due indici di *clustering* globale per *multiplex networks*.

Il primo è definito come rapporto tra il numero di 2-triangoli ed $(L-1)$ volte il numero di 1-triadi contenute nella rete *multilayer*:

$$C_1 = \frac{\sum_{\widetilde{r}=1}^{L} \sum_{\widetilde{t}=1,\, \widetilde{t}\neq\widetilde{r}}^{L} \mathbf{W}_{\rho}^{\alpha}(\widetilde{r}) \mathbf{W}_{\sigma}^{\rho}(\widetilde{t}) \mathbf{W}_{\alpha}^{\sigma}(\widetilde{r})}{(L-1) \sum_{\widetilde{r}=1}^{L} \mathbf{W}_{\rho}^{\alpha}(\widetilde{r}) (\mathbf{U}_{\sigma}^{\rho} - \boldsymbol{\delta}_{\sigma}^{\rho}) \mathbf{W}_{\alpha}^{\sigma}(\widetilde{r})} \tag{3.15}$$

dove si utilizza l'operazione tensoriale di contrazione per calcolare la traccia del tensore risultante.

Il secondo indice, in analogia con quanto definito dalla (3.14), è invece definito come il rapporto tra il numero di 3-triangoli ed $(L-2)$ volte il numero di 2-triadi contenute nel *multilayer network*:

$$C_2 = \frac{\sum_{\widetilde{r}=1}^{L} \sum_{\widetilde{t}=1,\, \widetilde{t}\neq\widetilde{r}}^{L} \sum_{\widetilde{q}=1,\, \widetilde{q}\neq\widetilde{t}\neq\widetilde{r}}^{L} \mathbf{W}_{\rho}^{\alpha}(\widetilde{r}) \mathbf{W}_{\sigma}^{\rho}(\widetilde{t}) \mathbf{W}_{\alpha}^{\sigma}(\widetilde{q})}{(L-2) \sum_{\widetilde{r}=1}^{L} \sum_{\widetilde{t}=1,\, \widetilde{t}\neq\widetilde{r}}^{L} \mathbf{W}_{\rho}^{\alpha}(\widetilde{r}) (\mathbf{U}_{\sigma}^{\rho} - \boldsymbol{\delta}_{\sigma}^{\rho}) \mathbf{W}_{\alpha}^{\sigma}(\widetilde{t})}. \tag{3.16}$$

Una ulteriore generalizzazione del coefficiente di *clustering* globale, invece, è quella proposta da De Domenico *et al.* (2013). In analogia con la (2.12), calcoliamo il quadrato del *multilayer adjacency tensor* $\mathbf{M}_{\beta\widetilde{\delta}}^{\alpha\widetilde{\gamma}}$ costruendo il tensore dell'ottavo ordine $\mathbf{M}_{\beta\widetilde{\delta}}^{\alpha\widetilde{\gamma}} \mathbf{M}_{\sigma\widetilde{\eta}}^{\rho\widetilde{\epsilon}}$ e contraendo β con ρ e $\widetilde{\delta}$ con $\widetilde{\epsilon}$.

Tale coefficiente può essere definito, allora, come

$$c(\mathbf{M}_{\beta\widetilde{\delta}}^{\alpha\widetilde{\gamma}}) = \mathcal{N}^{-1} \frac{\mathbf{M}_{\beta\widetilde{\delta}}^{\alpha\widetilde{\gamma}} \mathbf{M}_{\epsilon\widetilde{\eta}}^{\beta\widetilde{\delta}} \mathbf{M}_{\alpha\widetilde{\gamma}}^{\epsilon\widetilde{\eta}}}{\mathbf{M}_{\beta\widetilde{\delta}}^{\alpha\widetilde{\gamma}} (\mathbf{U}_{\epsilon\widetilde{\eta}}^{\beta\widetilde{\delta}} - \boldsymbol{\delta}_{\epsilon\widetilde{\eta}}^{\beta\widetilde{\delta}}) \mathbf{M}_{\alpha\widetilde{\gamma}}^{\epsilon\widetilde{\eta}}} \tag{3.17}$$

dove $(\mathbf{U}_{\epsilon\widetilde{\eta}}^{\beta\widetilde{\delta}} - \boldsymbol{\delta}_{\epsilon\widetilde{\eta}}^{\beta\widetilde{\delta}})$ è il tensore d'adiacenza corrispondente ad un *multilayer network* completo (senza *self-loop*) e $\mathcal{N}$ è un fattore di normalizzazione, scelto in genere come $\mathcal{N} = \max_{\alpha,\beta,\widetilde{\gamma},\widetilde{\delta}} \mathbf{M}_{\beta\widetilde{\delta}}^{\alpha\widetilde{\gamma}}$.

La contrazione eseguita nella (3.17) permette di contare il numero di diversi triangoli, inclusi quelli in cui il *walk* sottostante attraversa qualunque combinazione di relazioni *interlayer* o *intralayer*. Dunque,

in un qualche modo possiamo considerare l'indice di *clustering* globale appena definito come una sorta di generalizzazione dei due indici proposti nella (3.13) e (3.14): anche in questo caso, infatti, si tiene conto dei triangoli (cicli di lunghezza 3) che si realizzano considerando le relazioni inter-livello tra i nodi che rappresentano la stessa controparte in ciascun livello.

Un secondo approccio proposto sempre da De Domenico *et al.* (2013), certamente più facile e concettualmente più immediato, è quello di calcolare il coefficiente di *clustering* globale sull'*overlay monoplex network* $\mathbf{O}^{\alpha}_{\beta}$ definito nella (3.9). In tal caso, otteniamo

$$\begin{aligned} c(\mathbf{O}^{\alpha}_{\beta}) &= \mathcal{M}^{-1} \frac{\mathbf{M}^{\alpha\widetilde{\gamma}}_{\beta\widetilde{\gamma}}\, \mathbf{M}^{\beta\widetilde{\delta}}_{\epsilon\widetilde{\delta}}\, \mathbf{M}^{\epsilon\widetilde{\eta}}_{\alpha\widetilde{\eta}}}{\mathbf{M}^{\alpha\widetilde{\gamma}}_{\beta\widetilde{\gamma}}\, (\mathbf{U}^{\beta\widetilde{\delta}}_{\epsilon\widetilde{\delta}} - \boldsymbol{\delta}^{\beta\widetilde{\delta}}_{\epsilon\widetilde{\delta}})\, \mathbf{M}^{\epsilon\widetilde{\eta}}_{\alpha\widetilde{\eta}}} = \\ &= \mathcal{M}^{-1} \frac{\mathbf{O}^{\alpha}_{\beta}\, \mathbf{O}^{\beta}_{\epsilon}\, \mathbf{O}^{\epsilon}_{\alpha}}{\mathbf{O}^{\alpha}_{\beta}\, (\mathbf{U}^{\beta\widetilde{\delta}}_{\epsilon\widetilde{\delta}} - \boldsymbol{\delta}^{\beta\widetilde{\delta}}_{\epsilon\widetilde{\delta}})\, \mathbf{O}^{\epsilon}_{\alpha}} \end{aligned} \tag{3.18}$$

dove, ancora una volta, $(\mathbf{U}^{\beta\widetilde{\delta}}_{\epsilon\widetilde{\delta}} - \boldsymbol{\delta}^{\beta\widetilde{\delta}}_{\epsilon\widetilde{\delta}})$ è il tensore d'adiacenza corrispondente ad un *overlay monoplex network* completo (senza *self-loop*) e $\mathcal{M}$ è un fattore di normalizzazione, scelto in genere pari a $\mathcal{M} = \big(\max_{\alpha,\beta} \mathbf{M}^{\alpha\widetilde{\gamma}}_{\beta\widetilde{\delta}}\big)/L$.

La (3.17) e la (3.18) assumono lo stesso valore quando c'è un solo livello (rete *monoplex*) oppure, più in generale, quando non ci sono relazioni inter-livello e tutti i tensori di adiacenza intra-livello sono esattamente gli stessi.

3.5 Misure di prominenza

Come è già stato notato nel Capitolo 2, risulta spesso utile poter assegnare una misura globale di *importanza* per ciascun nodo, aggregando l'informazione ottenuta da ciascuno dei livelli entro i quali partecipa. In particolar modo, potrebbe risultare necessario studiare le caratteristiche di *prominenza* di un particolare nodo v_i all'interno della rete *multiplex*, che in un'ottica più generale si riflettono nelle note misure di centralità.

Abbiamo già notato nello scorso paragrafo come la semplice aggregazione della rete *multiplex* utilizzando l'*overlay monoplex network* (3.9) conduca spesso, a risultati ben diversi dalla realtà: tale soluzione potrebbe essere più appropriata, ad esempio, in qualunque tipo di rete *multiplex* in cui non esistono relazioni inter-livello.

Risulta pertanto necessario essere in grado di definire delle generalizzazioni delle misure di centralità più note, che siano in grado di mantenere le loro caratteristiche e il loro significato, ma che allo stesso tempo possano tenere conto della complessità informativa gestita dalle reti *multiplex*.

Degree centrality

Una generalizzazione della *degree centrality* definita nella (2.14) e richiamata nella (2.31) è fornita da De Domenico *et al.* (2013).

Definiamo allora il *multidegree centrality vector* come il vettore

$$\begin{aligned}\mathbf{K}^{\alpha} &= \left[\mathbf{M}^{\alpha\widetilde{\gamma}}_{\beta\widetilde{\delta}}\mathbf{U}^{\widetilde{\delta}}_{\widetilde{\gamma}}\right]\mathbf{1}^{\beta} = \mathbf{P}^{\alpha}_{\beta}\,\mathbf{1}^{\beta} = \\ &= \left[\sum_{\widetilde{h}=1}^{L}\sum_{\widetilde{k}=1}^{L}\mathbf{C}^{\alpha}_{\beta}(\widetilde{h}\widetilde{k})\right]\mathbf{1}^{\beta} = \\ &= \sum_{\widetilde{h}=1}^{L}\sum_{\widetilde{k}=1}^{L}\mathbf{C}^{\alpha}_{\beta}(\widetilde{h}\widetilde{k})\,\mathbf{1}^{\beta} = \\ &= \sum_{\widetilde{h}=1}^{L}\sum_{\widetilde{k}=1}^{L}\mathbf{k}^{\alpha}(\widetilde{h}\widetilde{k})\end{aligned} \tag{3.19}$$

dove $\mathbf{k}^{\alpha}(\widetilde{h}\widetilde{k})$ è il vettore dei gradi definito nella (2.31) calcolato sul tensore di adiacenza inter-livello $\mathbf{C}^{\alpha}_{\beta}(\widetilde{h}\widetilde{k})$; ovviamente, nel caso specifico delle reti *multiplex* che stiamo trattando, il *multidegree centrality vector* può essere calcolato tenendo conto che tutti i tensori di adiacenza inter-livello $\mathbf{C}^{\alpha}_{\beta}(\widetilde{h}\widetilde{k})$ sono diagonali per definizione, e che quindi $\mathbf{C}^{\alpha}_{\beta}(\widetilde{h}\widetilde{k})\,\mathbf{1}^{\beta} = diag(\mathbf{C}^{\alpha}_{\beta}(\widetilde{h}\widetilde{k}))$ ogniqualvolta $\widetilde{h} \neq \widetilde{k}$. Va anche notato che $\mathbf{K}^{\alpha}$ differisce dal *degree centrality vector* che si potrebbe ottenere semplicemente applicando la (2.31) all'*overlay monoplex network* definito nella (3.9).

Betweenness centrality

Considerata la definizione di *path* data nella (3.12), è possibile definire la *multiplex betweenness centrality* di un nodo v_i sul livello $\widetilde{r} - esimo$ facendo riferimento a quanto proposto da Solé-Ribalta *et al.* (2014),

come:

$$g(v_i, \widetilde{r}) = \sum_{\substack{s=1 \\ s \neq t \neq i}}^{N} \sum_{\substack{t=1 \\ t \neq s \neq i}}^{N} \frac{\sigma_{s,t}(v_i, \widetilde{r})}{\sigma_{s,t}} \qquad \begin{matrix} i=1,2,\ldots,N \\ \widetilde{r}=1,2,\ldots,L \end{matrix} \tag{3.20}$$

dove $\sigma_{s,t}(v_i, \widetilde{r})$ è il numero di volte che il nodo v_i del livello $\widetilde{r} - esimo$ appartiene ad una geodesica tra v_s e v_t, e $\sigma_{s,t} = card(P^*_{v_s \to v_t})$ è il numero complessivo di geodesiche tra v_s e v_t. In altre parole, $g(v_i, \widetilde{r})$ rappresenta la somma, per ciascuna possibile coppia di nodi v_s e v_t diversi da v_i, delle frazioni di volte in cui il nodo v_i appartenente al livello $\widetilde{r} - esimo$ giace su una geodesica tra v_s e v_t.

Una misura globale di *multiplex betweenness centrality* per il generico nodo v_i è proposta ancora da Solé-Ribalta *et al.* (2014) come:

$$g(v_i) = \sum_{\widetilde{r}=1}^{L} g(v_i, \widetilde{r}) \qquad i = 1, 2, \ldots, L \tag{3.21}$$

Si nota immediatamente che, per come è stato definito, l'indice di *multiplex betweenness centrality* (3.20) tende fisiologicamente ad assegnare una maggiore centralità a tutti quei nodi che fungono da *ponte* tra nodi connessi su livelli diversi, perchè necessariamente un gran numero di *path* passerà proprio per questi.

Closeness centrality

Utilizzando un approccio simile a quanto appena descritto per la *multiplex betweenness centrality*, e richiamando la definizione data nel caso *monoplex* (2.21), è possibile definire l'indice di *multiplex closeness centrality* di un generico nodo v_i come la media dell'inverso delle distanze di tutte le geodesiche che iniziano da un qualunque nodo v_j e terminano in v_i.

Eigenvector centrality

La generalizzazione della misura di *eigenvector centrality* al caso *multiplex* non è immediata per via della diversa struttura algebrica che stiamo utilizzando rispetto al caso *monoplex*, e in quanto tale sono stati proposti numerosi indici in grado di tenere conto della complessità informativa gestita dalla rete *multiplex* in questione.

Una prima misura estremamente semplicistica, proposta da Solá *et al.* (2013), consiste nel calcolare L diversi vettori di *eigenvector cen-*

trality $\mathbf{C}_{E,\widetilde{r}}$ e poi, tramite una qualche specifica trasformazione lineare, aggregarli in un unico vettore $\overline{\mathbf{C}}_E$.

Una misura più formale, proposta da De Domenico *et al.* (2013), fa invece riferimento al seguente problema:

$$\mathbf{M}_{\beta\widetilde{\delta}}^{\alpha\widetilde{\gamma}}\,\mathbf{\Theta}_{\alpha\widetilde{\gamma}} = \lambda_{(1)}\mathbf{\Theta}_{\beta\widetilde{\delta}} \tag{3.22}$$

dove $\mathbf{\Theta}_{\alpha\widetilde{\gamma}}$ è l'*autotensore* che racchiude la misura di *eigenvector centrality* per ciascun nodo in ciascun livello, la cui soluzione può essere individuata iterativamente (De Domenico *et al.*, 2013). Una prima generalizzazione formale dell'indice di *eigenvector centrality* (2.24) risulta dunque dall'espressione:

$$\mathbf{\Theta}_{\beta\widetilde{\delta}} = \lambda_{(1)}^{-1}\,\mathbf{M}_{\beta\widetilde{\delta}}^{\alpha\widetilde{\gamma}}\mathbf{\Theta}_{\alpha\widetilde{\gamma}}\,. \tag{3.23}$$

Un modo alternativo e più semplice suggerito da De Domenico *et al.* (2013) per risolvere il problema definito dalla (3.22) si basa sul lavoro svolto da Kolda e Bader (2009) in termini di procedure di *scomposizione* di tensori di qualunque ordine. Senza scendere troppo nel dettaglio della trattazione sull'argomento, Kolda e Bader (2009) hanno definito col termine *matricizzazione* (spesso più nota coi termini *flattening* o *unfolding*) quella particolare operazione che permette di trasformare un qualunque tensore in una matrice (o vettore) con particolari dimensioni, cioè che definisce una mappa biettiva, lineare ed univocamente definita $f : \mathbb{R}^{N\times N\times L\times L} \to \mathbb{R}^{NL\times NL}$.

Nel caso del tensore di adiacenza definito dalla (3.5) con la particolare restrizione che impone il tensore di adiacenza inter-livello $\mathbf{C}_{\beta}^{\alpha}(\widetilde{h}\widetilde{k})$ diagonale, questo potrà dunque essere trasformato in un tensore quadrato del secondo ordine (matrice) di dimensione $NL \times NL$. L'operazione di *unfolding* applicata a tale contesto produce $L!$ diverse matrici di dimensione $NL \times NL$, tante quante sono le permutazioni dei blocchi di dimensione $N\times N$ sulla sua diagonale, definite *supra-adjacency matrices*.

La generica *supra-adjacency matrix* può essere espressa nella seguente forma a blocchi

$$\widetilde{\mathbf{M}}_{\delta}^{\gamma} = \begin{bmatrix} \mathbf{W}_{\beta}^{\alpha}(1) & \mathbf{C}_{\beta}^{\alpha}(12) & \dots & \dots & \mathbf{C}_{\beta}^{\alpha}(1L) \\ \mathbf{C}_{\beta}^{\alpha}(21) & \mathbf{W}_{\beta}^{\alpha}(2) & \dots & \dots & \mathbf{C}_{\beta}^{\alpha}(2L) \\ \vdots & \vdots & \ddots & & \vdots \\ \vdots & \vdots & & \ddots & \vdots \\ \mathbf{C}_{\beta}^{\alpha}(L1) & \mathbf{C}_{\beta}^{\alpha}(L2) & \dots & \dots & \mathbf{W}_{\beta}^{\alpha}(L) \end{bmatrix} \tag{3.24}$$

dove, con un leggero abuso di notazione, $\mathbf{W}_{\beta}^{\alpha}(\widetilde{r}) = \mathbf{C}_{\beta}^{\alpha}(\widetilde{rr})$ corrisponde al tensore di adiacenza intra-livello del livello $\widetilde{r} - esimo$ e la generica $\mathbf{C}_{\beta}^{\alpha}(\widetilde{hk})$ corrisponde al tensore di adiacenza inter-livello definito dalla (3.3); in questo caso, gli indici e i pedici α e β vengono *rimpiazzati* dagli indici e pedici $\gamma, \delta = 1, 2, \ldots, NL$.

A questo punto, possiamo risolvere il problema

$$\widetilde{\mathbf{M}}_{\delta}^{\gamma}\, \widetilde{\boldsymbol{\theta}}_{\gamma} = \widetilde{\lambda}_{(1)}\, \widetilde{\boldsymbol{\theta}}_{\delta} \tag{3.25}$$

la cui soluzione $\widetilde{\boldsymbol{\theta}}_{\gamma}$ è un vettore di lunghezza NL interpretabile come il *supra-eigenvector* associato al più grande *supra-eigenvalue* della *supra-adjacency matrix* $\mathbf{M}_{\delta}^{\gamma}$, ottenuto dall'applicazione dell'operazione di *unfolding* sull'autotensore $\boldsymbol{\Theta}_{\alpha\widetilde{\gamma}}$. Siccome è dimostrabile che la *supra-adjacency matrix* conserva le stesse proprietà spettrali del *multiplex adjacency tensor*, gli NL elementi del *supra-vector* $\widetilde{\boldsymbol{\theta}}_{\gamma}$ verranno interpretati, a seconda della loro posizione, come gli indici di *multiplex eigenvector centrality* del generico nodo v_i sul livello $\widetilde{r} - esimo$.

Rimane comunque importante sottolineare la necessità della validità delle ipotesi del *Teorema di Perron-Frobenius* per la matrice $\widetilde{\mathbf{M}}_{\delta}^{\gamma}$ e, in particolare, la sua *irriducibilità*; se tale ipotesi è soddisfatta, il teorema garantisce (tra le altre cose) la positività del *supra-eigenvector* associato e quindi degli indici di *multiplex eigenvector centrality*.

Un aspetto non meno importante da tenere in considerazione è che, in generale, la centralità di un generico nodo v_i all'interno di uno specifico livello potrebbe dipendere non solo dai nodi nel vicinato V_i che giacciono sullo stesso livello, ma anche da tutti i nodi che giacciono nel vicinato che fa riferimento allo stesso nodo, su altri livelli.

Per tenere conto di tale informazione, Solá *et al.* (2013) definiscono l'indice di *multiplex heterogeneous eigenvector centrality* del nodo v_i sul generico livello $\widetilde{r} - esimo$ come la generica posizione dell'*autovettore di Perron* associato alla matrice $\mathbb{A}$ che andrò a definire nelle prossime righe.

Nel seguito, definiamo *matrice di influenza* $\widetilde{\mathbf{W}}_{\widetilde{\delta}}^{\widetilde{\gamma}}$ di dimensione $L \times L$ quella particolare matrice positiva il cui generico elemento (che, per semplicità notazionale, chiameremo $w_{\widetilde{\gamma}\widetilde{\delta}}$) misura l'influenza del livello $\widetilde{\gamma} - esimo$ sul livello $\widetilde{\delta} - esimo$; ipotizzando che ciascuna relazione inter-livello tra due qualunque livelli della rete *multiplex* sia uguale per ciascun nodo, allora tale elemento rappresenta anche il valore che si posiziona sulla diagonale del tensore di adiacenza inter-livello $\mathbf{C}_{\beta}^{\alpha}(\widetilde{hk})$ definito nella (3.3). Così, ad esempio, la $\mathbf{C}_{\beta}^{\alpha}(12)$ rappresenterà il tensore

di adiacenza inter-livello associato alla relazione inter-livello che intercorre tra il livello 1 ed il livello 2 della rete *multiplex*, e avrà il generico elemento sulla diagonale, $\mathbf{C}_{\beta}^{\alpha}(12)\mathbf{e}_{\alpha}(i)\mathbf{e}^{\beta}(i)$, pari a w_{12}.

A questo punto, è possibile definire la matrice quadrata in $\mathbb{R}^{NL\times NL}$

$$\mathbb{A} = \begin{bmatrix} w_{11}\mathbf{W}_{\beta}^{\alpha\top}(1) & w_{21}\mathbf{W}_{\beta}^{\alpha\top}(2) & \dots & w_{L1}\mathbf{W}_{\beta}^{\alpha\top}(L) \\ w_{12}\mathbf{W}_{\beta}^{\alpha\top}(1) & w_{22}\mathbf{W}_{\beta}^{\alpha\top}(2) & \dots & w_{L2}\mathbf{W}_{\beta}^{\alpha\top}(L) \\ \vdots & \vdots & \ddots & \vdots \\ w_{1L}\mathbf{W}_{\beta}^{\alpha\top}(1) & w_{2L}\mathbf{W}_{\beta}^{\alpha\top}(2) & \dots & w_{LL}\mathbf{W}_{\beta}^{\alpha\top}(L) \end{bmatrix} \tag{3.26}$$

dove $\mathbf{W}_{\beta}^{\alpha\top}(\widetilde{r})$ corrisponde alla matrice di adiacenza dell'$\widetilde{r}-esimo$ livello, trasposta. E' possibile definire $\mathbb{A}$ in vari modi, a seconda del tipo di prodotto di utilizzato e, quindi, delle matrici delle quali disponiamo.

Ad esempio, utilizzando il *prodotto di Hadamard*,

$$\begin{aligned} \mathbb{A} &= \begin{bmatrix} w_{11}\mathbf{U}_{\beta}^{\alpha} & \dots & w_{L1}\mathbf{U}_{\beta}^{\alpha} \\ \vdots & \ddots & \vdots \\ w_{1L}\mathbf{U}_{\beta}^{\alpha} & \dots & w_{LL}\mathbf{U}_{\beta}^{\alpha} \end{bmatrix} \circ \begin{bmatrix} \mathbf{W}_{\beta}^{\alpha\top}(1) & \dots & \mathbf{W}_{\beta}^{\alpha\top}(L) \\ \vdots & \ddots & \vdots \\ \mathbf{W}_{\beta}^{\alpha\top}(1) & \dots & \mathbf{W}_{\beta}^{\alpha\top}(L) \end{bmatrix} = \\ &= \left(\widetilde{\mathbf{W}}_{\widetilde{\delta}}^{\widetilde{\gamma}} \otimes \mathbf{U}_{\beta}^{\alpha}\right) \circ \begin{bmatrix} \mathbf{W}_{\beta}^{\alpha\top}(1) & \dots & \mathbf{W}_{\beta}^{\alpha\top}(L) \\ \vdots & \ddots & \vdots \\ \mathbf{W}_{\beta}^{\alpha\top}(1) & \dots & \mathbf{W}_{\beta}^{\alpha\top}(L) \end{bmatrix} \end{aligned} \tag{3.27}$$

oppure, facendo riferimento al *prodotto di Kathri-Rao*, come

$$\mathbb{A} = \widetilde{\mathbf{W}}_{\widetilde{\delta}}^{\widetilde{\gamma}} \odot \begin{bmatrix} \mathbf{W}_{\beta}^{\alpha\top}(1) & \mathbf{W}_{\beta}^{\alpha\top}(2) & \dots & \mathbf{W}_{\beta}^{\alpha\top}(L) \end{bmatrix} \tag{3.28}$$

Indipendentemente da ciò, però, è possibile ricavare l'*autovettore di Perron* associato alla matrice $\mathbb{A}$ che avrà lunghezza NL. Come il *supra-eigenvector* che risolve la (3.25), così tale autovettore potrà essere *spezzato* in L sotto-vettori di lunghezza N, ciascuno dei quali prenderà il nome di *multiplex heterogeneous eigenvector centrality vector* e la cui generica cella rappresenterà la misura di *multiplex eigenvector centrality* associata al nodo n_i sul livello $\widetilde{r}-esimo$.

Il nome *autovettore di Perron* deriva dall'omonimo *Teorema di Perron-Frobenius* che garantisce (tra le altre cose) la positività dell'autovettore associato al più grande autovalore di $\mathbb{A}$ matrice irriducibile. Per eventuali approfondimenti sulla sua esistenza ed unicità, rimando al lavoro svolto da Romance *et al.* (2015) e da Solá *et al.* (2013).

Capitolo 4

Pacchetto di funzioni sviluppate

In questo capitolo viene presentato un pacchetto di funzioni sviluppate per gestire le reti *multiplex* in ambiente *R* (R Development Core Team, 2011), e per ricavare i principali indici di *clustering* e le principali misure di centralità descritte nel Capitolo 3.

4.1 Generalità del pacchetto

In *R* esistono già alcuni pacchetti di funzioni che permettono (tra le altre cose) di implementare le reti *monoplex*, calcolare i principali indici di centralità e le principali misure di *clustering*, rappresentare graficamente i grafi sottostanti e ricavare i descrittori più noti in letteratura. Tra di questi, i più noti ed utilizzati sono certamente *igraph* (Csardi e Nepusz, 2006), del quale farò utilizzo in alcune funzioni, e *sna* (Butts, 2014). Ad ora, però, non è ancora presente alcun pacchetto di funzioni in grado di gestire le reti *multiplex*, implementarne una struttura coerente con la formalizzazione teorica sottostante e facilmente utilizzabile, calcolare i descrittori e le misure di centralità proposte in precedenza e, soprattutto, in grado di essere il più possibile generale ed applicabile.

Uno strumento particolarmente utile, del quale farò un breve utilizzo nel Capitolo 5 per produrre alcuni grafici, è `muxViz`: questa è un'interfaccia grafica gratuita ed *open-source* proposta da De Domenico *et al.* (2014), che permette di interfacciare R ed il software *Octave* per calcolare alcune delle misure proposte in ambito *multiplex*. Sebbene questo rimanga di fatto l'unico strumento attualmente in grado di fornire una rappresentazione grafica delle reti *multiplex* (e non solo) e permet-

Tabella 4.1: Tabella riassuntiva delle funzioni proposte nel pacchetto

Funzionalità	*Nome funzione*
Creazione della struttura	`create.multiplex`
	`add.interlayer.multiplex`
Estrazione degli elementi	`interlayer.multiplex`
	`nodes.multiplex`
	`layers.multiplex`
	`adjacency.multiplex`
	`type.multiplex`
	`graph.multiplex`
Principali matrici ottenibili tramite semplificazioni della struttura multiplex	`aggregated.topological.multiplex`
	`aggregated.overlapping.multiplex`
	`supra.adjacency.multiplex`
Estensione dei descrittori proposti nel pacchetto igraph	`degree.multiplex`
	`degree.distribution.multiplex`
	`total.degree.multiplex`
	`mean.degree.multiplex`
	`variance.degree.multiplex`
	`density.multiplex`
Misure di uniformità/eterogeneità sulla distribuzione dei gradi tra i livelli	`entropy.degree.multiplex`
	`participation.degree.multiplex`
Indici di clustering per reti multiplex	`local.clustering.multiplex`
	`c1.local.multiplex`
	`c2.local.multiplex`
	`C1.global.multiplex`
	`C2.global.multiplex`
	`global.overlay.multiplex`
Indici di centralità per reti multiplex	`degree.centrality.multiplex`
	`supra.eigenvector.centrality.multiplex`
	`heter.eigenvector.centrality.multiplex`

ta la gestione di reti con dimensioni elevate attraverso un'interfaccia *user-friendly*, soffre di una ridotta possibilità di essere personalizzato dall'utente. Inoltre, le modalità di inserimento dei dati risultano essere spesso diverse da come questi vengono generalmente organizzati e, per tale motivo, potrebbe essere utile sviluppare degli strumenti alternativi utilizzabili direttamente da *console*.

Quel che mi propongo di illustrare nelle prossime pagine è il pacchetto di funzioni che ho sviluppato: queste si pongono l'obiettivo di essere il più possibile generali, con un *output* facilmente restringibile a particolari gruppi di nodi/livelli. Le funzionalità di questo pacchetto possono essere riassunte nella Tabella 4.1, in cui *classifico* le funzioni in 7 grandi gruppi a seconda dello scopo per le quali sono state progettate. Nel seguito descrivo come utilizzare le funzioni sviluppate, le loro principali funzionalità e per le più *complesse* illustro alcuni passaggi.

4.2 Creazione della struttura multiplex ed estrazione degli elementi

La funzione principale, sulla quale si basano tutte le altre funzioni sviluppate, prende il nome di `create.multipex` (A.1):

```
create.multiplex(nodes, layersNames = FALSE, layer1, type1, ...)
```

Questa funzione permette di creare un oggetto `multiplex`, che è la struttura sulla quale si basano tutte le altre funzioni che ho sviluppato.

La funzione prende in input i seguenti argomenti:

- `nodes`: fa riferimento ad una qualunque struttura di tipo `data.frame`, che ha come prima colonna l'identificativo (numero progressivo) del nodo v_i e come seconda colonna l'etichetta relativa al nome;

- `layersNames`: fa riferimento ad un vettore di stringhe che identificano i nomi dei livelli che andrò ad inserire. Nel caso in cui venga lasciata di default, verranno assegnati automaticamente i nomi *Layer(i)* dalla funzione stessa;

- `layer1`: rappresenta il primo livello della rete *multiplex*, e fa riferimento ad una qualunque struttura del tipo `data.frame` oppure `matrix`, composta da 3 colonne. Per ciascuna riga (che rappresenta un arco), la prima colonna rappresenta l'identificativo del nodo di partenza, la seconda colonna rappresenta l'identificativo del nodo di arrivo, la terza colonna rappresenta il *peso* del legame (appartenente a $0, 1$ se la relazione non è pesata);

- `type1`: è una stringa che fa riferimento al tipo di relazione (direzionale o non direzionale) prevista per il livello inserito nell'argomento `layer1`;

- `...` : è l'argomento che permette di inserire qualunque altra struttura simile a quella per `layer1`, e qualunque tipo di stringa simile a `type1`. In particolare, è possibile specificare il tipo di relazione appena dopo aver inserito un nuovo livello o, non esprimendosi, lasciare che quest'ultima sia assunta come non-direzionale. La gestione di questo argomento ha una sezione dedicata all'interno della funzione, che illustrerò più nello specifico.

Una caratteristica che rende questa funzione estremamente utile ed adattabile a qualunque situazione è il fatto che vengano richiesti pochi *input* essenziali: l'unica restrizione è che questi siano formattati secondo una precisa struttura, descritta poco fa, e che vengano inseriti al posto giusto. In particolare, l'argomento "..." consente di inserire un numero indefinito di livelli, eventualmente accompagnati da una stringa che specifica il tipo di relazione; è infatti plausibile assumere che nel caso di livelli con relazioni non-direzionali, la struttura rappresentante il livello contenga solo metà degli archi, dando per scontata l'altra metà (gli archi nel *verso* opposto): questo rischia di diventare cruciale nel caso di relazioni direzionali, in cui invece il *verso* dello arco ha un'effettiva rilevanza.

Con riferimento al codice sviluppato all'interno della funzione, dalla riga 19 alla riga 50 avviene la gestione dell'argomento "...", che permette massima flessibilità nella gestione dell'input. Qualunque oggetto inserito al suo interno viene strutturato in un'unico oggetto `list` e, per ciascun elemento, viene svolto un controllo di ammissibilità alla struttura prevista per i livelli (righe 22-25). Se l'oggetto supera tutti i controlli, questo viene inserito nella lista dei livelli, inizializzata dal `layer1` in riga 5.

Le righe 30-33 aggiornano il vettore con le tipologie di relazione per l'ultimo `layer` inserito all'interno dell'ultima cella dell'argomento "...": in quanto ultimo oggetto inserito, significa che non viene specificata la tipologia della relazione e quindi questa è assunta essere non-direzionale.

Le righe 54-59 controllano se l'argomento `layersNames` è stato popolato con un vettore di stringhe, e in caso contrario assegnano un nome predefinito tramite la funzione `sprintf`.

Le righe 61-78 costruiscono le matrici di adiacenza di ciascun livello. Le righe 67-69 sommano la matrice di adiacenza trasposta se la relazione è non-direzionale, in modo tale da garantirne la simmetria: se non si facesse questo passaggio, la matrice di adiacenza riferita a livelli con relazioni non-direzionali risulterebbe essere triangolare inferiore/superiore perchè popolata dai soli archi inseriti nella struttura `layer` (che identificano per natura uno dei due versi di percorrenza, anche se la relazione è non-direzionale). Infine, le righe 71-74 eseguono un controllo su eventuali *self-loop* sulla diagonale di ciascuna matrice di adiacenza: se ne vengono identificati, questi vengono rimossi (riga 72) e viene segnalato in console (riga 73).

Una volta svolte tutte le elaborazioni degli input nei formati richiesti, i risultati vengono organizzati in una struttura `list` che costituisce l'oggetto `multiplex` creato, ovvero l'output della funzione.

In particolare, gli oggetti inclusi all'interno della lista che costituisce l'output sono i seguenti:

- `nodes`: è l'oggetto `data.frame` preso in input, contenente tutte le informazioni circa i nodi della rete *multiplex*;
- `layers`: è una lista con tutti i `data.frame` collezionati dall'input della funzione. Ciascun elemento di tale lista ha un nome che rappresenta l'etichetta del livello d'appartenenza;
- `adjacency`: è un oggetto `list` con tutte le matrici di adiacenza costruite, una per ogni livello;
- `type`: è un vettore di stringhe, ciascuna delle quali fa riferimento al tipo di relazione (direzionale, `"directed"` o non-direzionale, `"undirected"`);
- `layersNames`: è un vettore di stringhe, ciascuna delle quali rappresenta l'etichetta del livello di riferimento. Qualora dovesse essere fornito l'argomento `layersNames` in input, l'output è esattamente tale vettore.

Una seconda funzione altrettanto utile è la `add.interlayer.multiplex` (A.2):

```
add.interlayer.multiplex(multiplex)
```

che può essere utilizzata per aggiungere in coda alla struttura `multiplex` creata con la funzione `create.multiplex` una matrice composta a blocchi da tutte le matrici di adiacenza inter-livello $\mathbf{C}^{\alpha}_{\beta}(\widetilde{h}\widetilde{k})$, $\widetilde{h} \neq \widetilde{k}$. Ognuna di queste è, per sua natura, diagonale: l'$i-esimo$ posto sulla diagonale avrà un 1 se v_i appartiene ad almeno un arco sia nel livello $\widetilde{h}-esimo$ che in quello $\widetilde{k}-esimo$, e dunque è effettivamente possibile passare dal livello $\widetilde{h}-esimo$ a quello $\widetilde{k}-esimo$ (e viceversa) tramite il nodo v_i perchè quest'ultimo non è isolato in nessuno dei due livelli; uno 0 in caso contrario.

Questa caratteristica è il risultato di una semplificazione che ho deciso di fare; sto infatti assumendo che le relazioni inter-livello abbiano tutte uno stesso peso comune pari ad 1, e che queste siano presenti solo quando il nodo è connesso in tutti e due i livelli. Si pensi, ad esempio, ad una rete *multiplex* che coinvolge più livelli/reti di trasporto di una

metropoli: sebbene tutte le stazioni siano presenti su tutti i livelli (l'ipotesi che sta alla base delle reti *multiplex*), può accadere che alcune di queste non siano affatto collegate alla rete di trasporto rappresentata nel generico livello: a questo punto, definire una relazione tra una stessa stazione effettivamente presente nella rete rappresentata da un certo livello e isolata nella rete rappresentante il secondo livello, potrebbe portare a risultati completamente diversi dalla realtà.

Qualora si volesse tenere conto di pesi diversi per le relazioni interlivello, invece, va certamente generalizzata la struttura che ho proposto: in ogni caso, il più delle volte risulta difficile assegnare un peso a ciascuna di queste, e quindi si preferisce mantenere una relazione binaria. Riprendendo ancora il caso della rete integrata di trasporti: passare da un livello ad un altro (ovvero, da una rete di trasporti all'altra) potrebbe avere un costo diverso a seconda del mezzo di trasporto (unitario, se è presente un biglietto integrato, o superiore in altri casi) e questo potrebbe addirittura cambiare da nodo a nodo (si pensi ad esempio alle stazioni più periferiche, che in genere richiedono tariffe superiori alle stazioni centrali).

La riga 6 definisce tutte le possibili coppie ottenute come combinazioni dei livelli della struttura `multiplex` e le righe 9-26 calcolano, per ciascuna coppia di livelli ottenuta, la rispettiva matrice di adiacenza inter-livello. In particolar modo, le righe 13-14 permettono di ottenere i vettori con gli identificativi dei nodi presenti in almeno un arco del livello, mentre la riga 15 interseca i due vettori ottenuti e restituisce un vettore con gli identificativi di tutti i nodi che appartengono ad almeno un arco in entrambi i livelli considerati. Le righe 17-22 costruiscono la matrice di adiacenza inter-livello relativa alla coppia di livelli selezionata dal ciclo `for`, e le righe 24-25 posizionano questa matrice all'interno della *interlayer supra-adjacency matrix*. La funzione permette di aggiungere in coda all'oggetto `multiplex` proprio quest'ultima matrice.

Associata a quest'ultima funzione vi è la (A.3):

```
interlayer.multiplex(multiplex, layer1, layer2)
```

che permette di estrarre la matrice di adiacenza inter-livello relativa ai due livelli indicizzati con gli argomenti `level1` e `level2`.

La riga 2 di questa funzione restituisce un messaggio di errore se i due argomenti sono uguali, perchè per definizione $\mathbf{C}^{\alpha}_{\beta}(\widetilde{h}\widetilde{h}) = \mathbf{W}^{\alpha}_{\beta}(\widetilde{h})$ e quest'ultima può essere ricavata con la funzione (A.6) che andrò ad illustrare in seguito.

Funzioni analoghe alla precedente, che permettono di estrarre particolari oggetti dalla struttura `multiplex` definita con la funzione `create.multiplex` (A.1), sono le funzioni (A.4):

```
nodes.multiplex(multiplex,
                index = 1:nrow(multiplex$nodes),
                label = FALSE)
```

che permette di estrarre l'oggetto `nodes`; la funzione (A.5):

```
layers.multiplex(multiplex,
                 index = 1:length(multiplex$layers),
                 label = FALSE)
```

che permette di estrarre l'oggetto `layers`; la funzione (A.6):

```
adjacency.multiplex(multiplex, index = 1:length(multiplex$adjacency))
```

che permette di estrarre la lista delle matrici di adiacenza (l'oggetto `adjacency`), e infine la funzione (A.7):

```
type.multiplex(multiplex, index = 1:length(multiplex$type))
```

che permette di estrarre l'oggetto `type`.

Ciascuna di queste funzioni consente di restringere l'output solo a determinati indici dei nodi/livelli di interesse; in particolare, per le funzioni (A.4) e (A.5) l'opzione `label = TRUE` restituisce un vettore di stringhe contenente le etichette dei nodi/livelli selezionati. Queste funzioni permettono di richiamare facilmente i principali oggetti appartenenti alla struttura `multiplex` all'interno di altre funzioni, evitando complicazioni sul codice.

La funzione (A.8):

```
graph.multiplex(multiplex)
```

associa a ciascun livello della struttura `multiplex` un oggetto di classe `igraph` costruito a partire dalla relativa matrice di adiacenza; questo permette di utilizzare gli strumenti contenuti nell'omonimo pacchetto per realizzare le funzioni che permettono di estendere al contesto *multiplex* i principali descrittori definiti per reti *monoplex*. In particolare, l'opzione `mode` all'interno della funzione `graph.adjacency` (che appartiene al pacchetto *igraph*) permette di specificare il tipo di relazione prevista per il livello in considerazione.

4.3 Semplificazione della struttura multiplex

Una funzione che permette di semplificare la rete *multiplex* al più banale caso *monoplex* è la (A.9):

```
aggregated.topological.multiplex(multiplex,
           indexNode = 1:length(nodes.multiplex(multiplex)),
           indexLayer = 1:length(layers.multiplex(multiplex)),
           verbose = FALSE)
```

Questa funzione permette di ricavare la *aggregated topological adjacency matrix* definita in Battiston *et al.* (2014), nel suo generico elemento come:

$$\mathbf{A}_{\beta}^{\alpha} = \begin{cases} 1 & \text{se } \exists\, \widetilde{r} : \mathbf{W}_{\beta}^{\alpha}(\widetilde{r}) = 1 \\ 0 & \text{altrimenti} \end{cases} \qquad \alpha, \beta = 1, 2, \ldots, N \tag{4.1}$$

Questa matrice permette di vedere se esiste un arco tra due nodi in almeno uno dei livelli considerati nella struttura `multiplex` (eventualmente selezionati con l'argomento `indexLayer`).

Nonostante questa matrice costituisca, a tutti gli effetti, una banale semplificazione della rete *multiplex* costruita con la `create.multiplex` (A.1), spesso potrebbe risultare un primo strumento per descrivere l'intero *multiplex*. E' chiaro, poi, che più livelli possiede un *multiplex network*, più tale matrice tenderà ad essere satura di 1.

Selezionando un solo indice per l'argomento `indexLayer`, la funzione permette di *dicotomizzare* la matrice di adiacenza $\mathbf{W}_{\beta}^{\alpha}(\widetilde{r})$ associata a tale livello: ciò potrebbe risultare particolarmente utile qualora si volessero implementare particolari misure di *clustering* su livelli caratterizzati da relazioni pesate (per approfondimenti, si può fare riferimento alle misure proposte da Barrat *et al.* (2004)).

Una seconda funzione altrettanto utile nell'aggregare le matrici di adiacenza che contengono le relazioni intra-livello della rete *multiplex* è la (A.10):

```
aggregated.overlapping.multiplex(multiplex,
           indexNode = 1:length(nodes.multiplex(multiplex)),
           indexLayer = 1:length(layers.multiplex(multiplex)),
           verbose = FALSE)
```

Questa consente di ottenere la *overlapped adjacency matrix* riferita all' *overlay monoplex network* definito dalla (3.9). La funzione permette di selezionare i livelli dell'oggetto `multiplex` entro cui costruirla tramite l'argomento `indexLayer`, e restituisce la matrice ottenuta come *somma verticale* delle matrici di adiacenza di tali livelli.

Una terza funzione che lavora sulle matrici di adiacenza è la (A.11):

```
supra.adjacency.multiplex(multiplex)
```

e permette di ricavare la *supra-adjacency matrix* definita dalla (3.24) a partire da una struttura `multiplex` costruita con la funzione (A.1).

Questa funzione utilizza la *interlayer supra-adjacency matrix* definita con la funzione (A.2) (riga 6) e popola la sua diagonale con tutte le matrici di adiacenza dei livelli della struttura `multiplex` (righe 8-10 del codice).

4.4 Descrittori

4.4.1 Estensione delle misure per reti monoplex

Un primo approccio che è possibile seguire per generalizzare al caso *multiplex* le principale misure descrittive definite per le *reti monoplex* è quello di calcolarle *parallelamente* livello per livello, raccogliendone i risultati all'interno di un vettore che avrà lunghezza pari al numero dei livelli dell'oggetto `multiplex` creato dalla (A.1). Questi risultati possono poi essere aggregati secondo qualche specifico criterio per realizzare un indice locale per ciascun nodo v_i, o addirittura un indice globale per l'intera rete presa in considerazione.

Seguendo tale approccio, è possibile applicare una serie di funzioni definite all'interno del pacchetto *igraph* sui grafi dei livelli dell'oggetto `multiplex`: questi vengono ricavati utilizzanto la funzione (A.8) precedentemente definita.

La funzione (A.12):

```
degree.multiplex(multiplex,
                 indexNode = 1:length(nodes.multiplex(multiplex)),
                 modeDirected = FALSE)
```

restituisce una lista con i vettori dei *degree* dei nodi selezionati con l'argomento `indexNode`, per ciascun livello della struttura `multiplex`.

L'argomento `modeDirected = TRUE` permette di restituire una lista associata a ciascun oggetto della lista restituita in output che fa riferimento ad un livello con relazione direzionale, contenente le misure di *indegree*, *outdegree* e *total degree*; se questa opzione non dovesse essere specificata, la funzione restituisce di default il *total degree* (pari alla somma di *indegree* e *outdegree*).

Una funzione simile alla precedente è la (A.13):

```
degree.distribution.multiplex(multiplex)
```

che restituisce una lista contenente, per ciascun livello, la relativa distribuzione dei *degree*. Questa funzione ha una doppia utilità: per ciascun livello, permette di osservare qual'è il *degree* massimo e di capire qual'è la percentuale di nodi che possiede un determinato *degree* all'interno del livello relativo.

Una funzione che utilizza la (A.12) è la (A.14):

```
total.degree.multiplex(multiplex,
                indexNode = 1:length(nodes.multiplex(multiplex)),
                indexLayer = 1:length(layers.multiplex(multiplex)),
                verbose = FALSE)
```

questa restituisce un vettore contenente la somma dei *degree* per ciascun nodo selezionato con l'argomento `indexNode`, su ciascun livello selezionato con l'argomento `indexLayer`; qualora uno o più livelli dovessero avere una relazione direzionale, si considera la misura di *total degree* come quella più opportuna su cui eseguire la somma. L'opzione `verbose = TRUE` restituisce un messaggio in *console* con i nomi dei livelli sui quali è stata eseguita la somma.

Alcune funzioni che permettono di calcolare i più comuni descrittori in maniera *parallela* (ovvero, livello per livello separatamente) sui livelli della struttura `multiplex` sono la (A.15):

```
mean.degree.multiplex(multiplex,
                indexNodeMean = 1:length(nodes.multiplex(multiplex)),
                verbose = FALSE)
```

che permette di calcolare la media dei *degree*, la (A.16):

```
variance.degree.multiplex(multiplex,
                indexNodeVar= 1:length(nodes.multiplex(multiplex)),
                verbose = FALSE)
```

che permette di calcolare la varianza dei *degree*, e la (A.17):

```
density.multiplex(multiplex)
```

che consente di calcolare la densità per ciascun livello della rete *multiplex*. In particolare, le due funzioni (A.15) e (A.16) utilizzano l'argomento `indexNodeMean` (`indexNodeVar`) per selezionare gli indici dei nodi su cui eseguire il calcolo della media (varianza) dei rispettivi *degree*.

4.4.2 Misure di eterogeneità

Alcune misure per determinare quanto uniformemente sono distribuiti i *degree* dei nodi all'interno della rete *multiplex* vengono invece implementate dalle funzioni (A.18) e (A.19).

In particolare, la (A.18):

```
entropy.degree.multiplex(multiplex,
        indexNode = 1:length(nodes.multiplex(multiplex)),
        indexOverlappingLayer = 1:length(layers.multiplex(multiplex),
        display = FALSE)
```

fa riferimento alla misura di entropia descritta dalla (3.10) per studiare l'omogeneità/eterogeneità della distribuzione dei *degree* dei nodi sui livelli del *multiplex*, e restituisce un vettore con tale misura per ciascun nodo (eventualmente selezionato con l'argomento `indexNode`). Se `display = TRUE`, la funzione restituisce un grafico con i risultati ottenuti, che permette un rapido confronto con eventuali altre strutture `multiplex`. Tale funzione utilizza anche una piccola correzione al numeratore dell'argomento del logaritmo, per evitare di restituire valori *NaN* se il *degree* è nullo.

La funzione (A.19):

```
participation.degree.multiplex(multiplex,
       indexNode = 1:length(nodes.multiplex(multiplex)),
       indexOverlappingLayer = 1:length(layers.multiplex(multiplex)),
       display = FALSE)
```

restituisce, per ciascun nodo selezionato con l'argomento `indexNode`, il coefficiente di partecipazione multiplo $P(n_i)$ descritto dalla (3.11). La sua struttura è itendica alla `entropy.degree.multiplex` (A.18).

4.4.3 Misure di transitività per reti multiplex

La funzione (A.20):

```
local.clustering.multiplex(multiplex,
                  indexNode = 1:length(nodes.multiplex(multiplex)))
```

restituisce una lista con i coefficienti di *clustering* definiti nella (2.11) per ciascun nodo scelto con l'argomento `indexNode`, per ciascun livello incluso nell'oggetto `multiplex`. Questa funzione utilizza la funzione `transitivity` del pacchetto *igraph* per calcolare tali indici (riga 7), il cui argomento `vids` combinato all'argomento `type = "local"` permette di selezionare i nodi entro i quali calcolare gli indici.

Alcune funzioni che permettono di implementare delle misure di *clustering* locale e globale in grado di tenere conto della possibilità per i nodi di formare dei triangoli considerando i rispettivi vicinati su più livelli contemporaneamente, sono invece implementate dalle funzioni (A.21) e (A.22) (misure locali) e dalle (A.23) e (A.24) (misure globali).

In particolare, la funzione (A.21):

```
c1.local.multiplex(multiplex,
         indexNode = 1:length(nodes.multiplex(multiplex)),
         indexLayer = 1:length(layers.multiplex(multiplex)))
```

permette di calcolare l'indice di *clustering* locale definito dalla (3.13), e restituisce un vettore con il valore assunto di ciascun nodo (eventualmente selezionato con l'argomento `indexNode`).

La riga 4 restituisce un messaggio di errore e termina la funzione, se vengono selezionati meno di due livelli con l'argomento `indexLayer` entro i quali calcolare l'indice; la riga 8 verifica poi che il pacchetto *gtools* (Warnes *et al.*, 2015) sia stato caricato: questo permette di definire tutte le possibili combinazioni di lunghezza `r = 2` degli indici dei livelli selezionati con l'argomento `indexLayer`, tramite la funzione `permutations`; in riga 9 quest'ultima viene utilizzata per produrre una matrice dove, per ciascuna colonna, viene riportata la generica combinazione di lunghezza 2.

Le righe 11-14 invece costruiscono il numeratore dell'indice, sommando tutti i triangoli ottenibili combinando 2 livelli diversi (cioè, tutte le coppie assegnate alla variabile `couples` in riga 9); le righe 16-19, invece, costruiscono il denominatore dell'indice.

Una seconda funzione è la (A.22):

```
c2.local.multiplex(multiplex,
        indexNode = 1:length(nodes.multiplex(multiplex)),
        indexLayer = 1:length(layers.multiplex(multiplex)))
```

che è identica alla `c1.local.multiplex`, ma come si deduce dal nome calcola l'indice di *clustering* locale definito dalla (3.14).

La funzione (A.23):

```
C1.global.multiplex(multiplex,
         indexLayer = 1:length(layers.multiplex(multiplex)))
```

restituisce l'indice di *clustering* globale, generalizzazione dell'indice di *clustering* locale (3.13), definito dalla (3.15).

Questa funzione è simile alla `c1.local.multiplex` nel suo funzionamento, ma restituisce il rapporto delle somme delle diagonali delle due matrici anzichè il rapporto delle singole celle (ovvero, anzichè un vettore di indici restituisce un singolo indice globale).

Analogamente, la funzione (A.24):

```
C2.global.multiplex(multiplex,
         indexLayer = 1:length(layers.multiplex(multiplex)))
```

restituisce l'indice definito dalla (3.16). In entrambi i casi, l'argomento `indexLayer` permette di selezionare i livelli della struttura `multiplex` entro i quali calcolare l'indice.

Un'ultima misura globale di *clustering* è implementata dalla funzione (A.25):

```
global.overlay.clustering.multiplex(multiplex,
               indexLayer = 1:length(layers.multiplex(multiplex)),
               verbose = FALSE)
```

che fa riferimento alla misura descritta dalla (3.17) e calcolata sull'*overlay multiplex network*. L'opzione `verbose = TRUE` restituisce l'informazione sul valore del coefficiente di standardizzazione utilizzato nel calcolo dell'indice (questo è calcolato con le righe 7-14).

4.5 Indici di centralità per reti multiplex

Una funzione che implementa il concetto di *multiplex degree centrality* definita dalla (3.17), invece, è la (A.26):

```
degree.centrality.multiplex (multiplex,
                  indexNode = 1:length(nodes.multiplex(multiplex)))
```

questa restituisce un vettore con l'indice calcolato su ciascuno dei nodi della struttura `multiplex` (eventualmente selezionati con l'argomento `indexNode`).

In questo caso, la proiezione operata sul *multiplex adjacency tensor* $\mathbf{M}_{\beta\tilde{\delta}}^{\alpha\tilde{\gamma}}$ che permette di ottenere il *projected monoplex network* $\mathbf{P}_{\beta}^{\alpha}$ (3.8) necessita delle matrici di adiacenza intra-livello: il doppio ciclo `for` (righe 7-16) restituisce la matrice `interlayersMatrix` pari alla somma di tutte le matrici di adiacenza inter-livello. Quest'ultima viene poi sommata (riga 18) alla *overlayMatrix* per ottenere la matrice di adiacenza riferita al *projected monoplex network*.

Infine, due funzioni che implementano due misure di *multiplex eigenvector centrality* sono la (A.27) e la (A.28). In particolare, la (A.27):

```
supra.eigenvector.centrality.multiplex(multiplex,
          indexNode = 1:length(nodes.multiplex(multiplex)),
          rowStand = TRUE,
          testIrreducibility = FALSE,
          maxPower = 100)
```

permette di implementare la misura proposta dalla (3.25) e ne restituisce l'indice derivante, per ciascun nodo in ciascun livello della struttura `multiplex`.

Le righe 6-27 permettono di testare l'irreducibilità della matrice di supra-adiacenza implementando la sottofunzione `irreducible`, che applicata ad una qualunque matrice X restituisce `TRUE` se questa è irriducibile (ovvero se $\exists\, m <$ `maxPower` tale per cui $[X^m]_{ij} > 0$, dove l'argomento `maxPower` è fissato arbitrariamente e in base alle dimensioni della rete *multiplex*) e `FALSE` se almeno una cella di X^{maxPower} è uguale a 0; in questo secondo caso, l'ipotesi di irriducibilità non è necessariamente garantita e, dunque, potrebbero non valere le ipotesi del *Teorema di Perron-Frobenius*: le righe 24-26 restituiscono un messaggio d'allerta qualora la *supra-adjacency matrix* non dovesse risultare irriducibile.

La riga 29 calcola il *supra-eigenvector* (l'autovettore di lunghezza NL associato alla *supra-adjacency matrix* della struttura *multiplex* calcolata in riga 2), mentre la riga 30 *spezza* tale vettore in L sottovettori di lunghezza N ciascuno, collezionandoli nella matrice `outMatrix`; le righe 34-36 permettono poi, se `rowStand = TRUE`, di *standardizzare* ciascuna riga della `outMatrix`, e questo potrebbe risultare utile quando si vogliono fare confronti tra nodi appartenenti ad uno stesso livello.

Infine, le righe 38-43 permettono di superare un problema che si presenta quando si calcolano gli autovettori di una matrice in R: se almeno uno degli autovalori appartiene al campo dei complessi $\mathbb{C}$, tutti gli altri vengono riportati utilizzando la notazione $a + ib$ anche se appartengono ad $\mathbb{R}$. In tal caso, se le ipotesi del *Teorema di Perron-Frobenius* sono soddisfatte, sappiamo che certamente l'autovettore associato all'autovalore (in modulo) più elevato appartiene ad $\mathbb{R}^+$ e dunque considerarne la sua parte reale tramite la funzione `Re()` non ne altera la natura; inoltre, R potrebbe restituire l'autovalore con i segni cambiati (negativi): in questo caso, è possibile moltiplicare l'autovettore per lo scalare (-1) senza alterarne la forma.

La funzione (A.28), invece:

```
heter.eigenvector.centrality.multiplex(multiplex,
      W = matrix(1, length(layers.multiplex(multiplex)),
                    length(layers.multiplex(multiplex))),
      indexNode = 1:length(nodes.multiplex(multiplex)),
      rowStand = TRUE,
      testIrreducibility = FALSE,
      maxPower = 100)
```

permette di calcolare l'indice di *multiplex heterogeneous eigenvector centrality* del nodo v_i sul generico livello $\widetilde{r} - esimo$.

Oltre agli argomenti, la funzione prende in input la matrice `W`, che corrisponde alla matrice di influenza $\widetilde{\mathbf{W}}_{\widetilde{\delta}}^{\widetilde{\gamma}}$.

La riga 6 esegue la funzione `do.call` con argomento `cbind` sulla lista delle matrici di adiacenza trasposte dei *layers* della struttura *multiplex*, e produce la seguente struttura:

$$\left[\mathbf{W}_{\beta}^{\alpha\top}(1) \;\; \mathbf{W}_{\beta}^{\alpha\top}(2) \;\; \ldots\ldots \;\; \mathbf{W}_{\beta}^{\alpha\top}(L-1) \;\; \mathbf{W}_{\beta}^{\alpha\top}(L)\right].$$

La riga 7 esegue la funzione `do.call` con argomento `rbind` sulla lista ottenuta dalle L repliche della struttura appena creata, e permette di ottenere la seguente matrice:

$$\begin{bmatrix} \mathbf{W}_{\beta}^{\alpha\top}(1) & \dots & \mathbf{W}_{\beta}^{\alpha\top}(L) \\ \vdots & \ddots & \vdots \\ \mathbf{W}_{\beta}^{\alpha\top}(1) & \dots & \mathbf{W}_{\beta}^{\alpha\top}(L) \end{bmatrix}.$$

La riga 8 permette, infine, di creare la matrice $\mathbb{A}$ definita dalla (3.26) eseguendo il prodotto di Hadamard (che in R può essere fatto utilizzando l'operatore $*$) tra la matrice appena ottenuta e la matrice ottenuta dal prodotto di Kronecker tra la matrice $\widetilde{\mathbf{W}}_{\tilde{\delta}}^{\tilde{\gamma}}$ e la matrice $\mathbf{U}_{\beta}^{\alpha}$. Così come la (A.27), questa funzione restituisce una matrice con tante righe quanti sono i *layers* della struttura `multiplex` presa in input, e tante colonne quanti sono i nodi.

Capitolo 5

Applicazione su dati reali

In questo capitolo, viene proposta un'applicazione su dati reali delle funzioni sviluppate illustrate nel Capitolo 4.

5.1 Descrizione del dataset

Il dataset che utilizzerò (Magnani *et al.*, 2013) descrive una rete sociale *multiplex* di 61 colleghi del Dipartimento di Informatica dell'Università di Aarhus, in Danimarca; sono state raccolte le relazioni su 5 diversi livelli di *rapporti* reciproci tra coppie di colleghi, di seguito descritte:

- *facebook*: se i due colleghi hanno stretto amicizia su Facebook;
- *leisure*: se i due colleghi trascorrono assieme il loro tempo libero;
- *work*: se i due colleghi lavorano assieme;
- *coauthor*: se i due colleghi sono autori dello stesso lavoro pubblicato;
- *lunch*: se i due colleghi pranzano spesso assieme.

Le relazioni per ciascuno di questi livelli sono non-direzionali e non-pesate: in altre parole, tutte le matrici di adiacenza sono simmetriche e binarie. Vista la natura delle relazioni descritte, l'ipotesi è che esista una relazione inter-livello tra due nodi rappresentanti la stessa entità che giacciono su livelli diversi, solo se questi appartengono ad almeno un arco nei rispettivi livelli: in altre parole, non sono ammesse relazioni inter-livello tra due nodi isolati.

Una prima analisi grafica può essere condotta utilizzando l'ambiente `muxViz`. Tralasciando la fase di preparazione dei dati nel formato richiesto, una prima illustrazione del dataset proposta è riportata nella Figura 5.1:

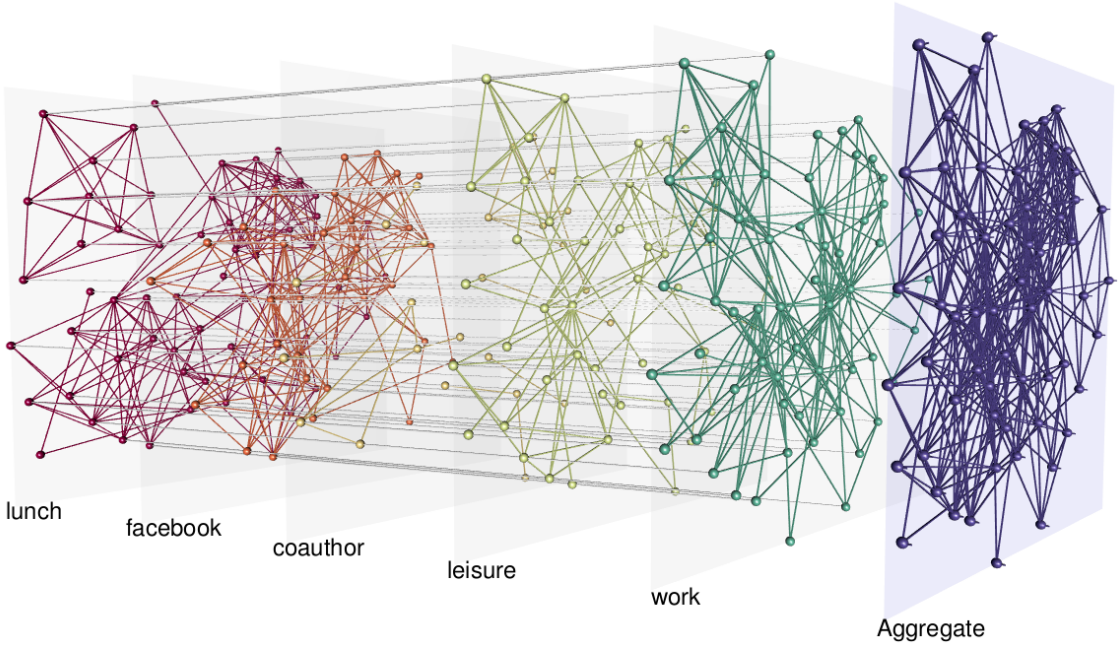

Figura 5.1: Rappresentazione dei livelli che costituiscono la rete *multiplex* sottostante. Per ciascuno di essi, vengono mostrati i nodi isolati, gli archi appartenenti a ciascun livello e quelli rappresentanti le relazioni inter-livello con gli altri livelli. *Aggregate* rappresenta l'*aggregated topological monoplex network* ottenuto unendo tutte le matrici di adiacenza dei 5 livelli.

Le informazioni relative ai nodi del *multiplex network* sono organizzate in una struttura `data.frame` con una colonna che indica il numero progressivo dell'identificativo (`nodeID`) e una seconda colonna che indica la rispettiva etichetta (`nodeLabel`), nella seguente forma:

```
> nodi

  nodeID nodeLabel
1      1      U102
2      2      U139
3      3       U33
4      4      U106
5      5      U107
6      6      U118
     ...       ...
```

Le informazioni relative ai livelli, invece, sono organizzate in una struttura `data.frame` composta da 3 colonne: la prima indica l'identificativo del nodo di partenza (`ID1`), la seconda colonna l'identificativo del nodo di arrivo (`ID2`), la terza il peso (`Weight`) della relazione (a seconda che questa sia pesata o binaria), nella forma:

```
> L1

  ID1 ID2 Weight
1   1   2      1
2   1   3      1
3  10  11      1
4  10  14      1
5  10  15      1
6  10  16      1
  ... ...    ...
```

dove ciascuna riga rappresenta un arco appartenente al livello.

Vi è infine un vettore contenente i nomi dei livelli della rete:

```
> labelLivelli

[1] "lunch"    "facebook" "coauthor" "leisure"  "work"
```

5.2 Creazione della struttura multiplex ed estrazione degli elementi

Utilizzando la funzione `create.multiplex` (A.1), è possibile costruire la struttura `multiplex` che rappresenta la rete in esame tramite il seguente comando:

```
M <- create.multiplex(nodes = nodi, layersNames = labelLivelli, layer1 = L1,
                      type1 = "undirected", L2, L3, L4, L5)
> str(M)

List of 5
$ nodes            :'data.frame':        61 obs. of  2 variables:
..$ nodeID   : int [1:61] 1 2 3 4 5 6 7 8 9 10 ...
..$ nodeLabel: chr [1:61] "U102" "U139" "U33" "U106" ...
$ layers           :List of 5
..$ lunch   : int [1:193, 1:3] 1 1 10 10 10 10 11 11 11 12 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : chr [1:193] "1" "2" "3" "4" ...
.. .. ..$ : chr [1:3] "ID1" "ID2" "Weight"
..$ facebook: int [1:124, 1:3] 12 12 13 13 13 13 13 13 15 15 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : chr [1:124] "194" "195" "196" "197" ...
.. .. ..$ : chr [1:3] "ID1" "ID2" "Weight"
..$ coauthor: int [1:21, 1:3] 10 12 18 23 23 23 26 26 26 26 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : chr [1:21] "318" "319" "320" "321" ...
.. .. ..$ : chr [1:3] "ID1" "ID2" "Weight"
..$ leisure : int [1:88, 1:3] 10 12 12 15 15 15 17 17 17 17 ...
.. ..- attr(*, "dimnames")=List of 2
```

```
.. .. ..$ : chr [1:88] "339" "340" "341" "342" ...
.. .. ..$ : chr [1:3] "ID1" "ID2" "Weight"
..$ work    : int [1:194, 1:3] 10 10 10 10 11 11 11 11 11 11 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : chr [1:194] "427" "428" "429" "430" ...
.. .. ..$ : chr [1:3] "ID1" "ID2" "Weight"
$ adjacency        :List of 5
..$ lunch   : num [1:61, 1:61] 0 1 1 0 0 0 0 0 0 0 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : chr [1:61] "1" "2" "3" "4" ...
.. .. ..$ : chr [1:61] "U102" "U139" "U33" "U106" ...
..$ facebook: num [1:61, 1:61] 0 0 0 0 0 0 0 0 0 0 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : chr [1:61] "1" "2" "3" "4" ...
.. .. ..$ : chr [1:61] "U102" "U139" "U33" "U106" ...
..$ coauthor: num [1:61, 1:61] 0 0 0 0 0 0 0 0 0 0 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : chr [1:61] "1" "2" "3" "4" ...
.. .. ..$ : chr [1:61] "U102" "U139" "U33" "U106" ...
..$ leisure : num [1:61, 1:61] 0 0 0 0 0 0 0 0 0 0 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : chr [1:61] "1" "2" "3" "4" ...
.. .. ..$ : chr [1:61] "U102" "U139" "U33" "U106" ...
..$ work    : num [1:61, 1:61] 0 0 0 0 0 0 0 0 0 0 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : chr [1:61] "1" "2" "3" "4" ...
.. .. ..$ : chr [1:61] "U102" "U139" "U33" "U106" ...
$ type             : chr [1:5] "undirected" "undirected" ...
$ layersNames      : chr [1:5] "lunch" "facebook" "coauthor" ...
```

Come si vede, la funzione richiede gli argomenti `nodes`, `layerNames` (facoltativo), `layer1` e `type1`; dopodichè, permette di inserire un qualunque numero di oggetti simili a `L1` per ciascuno degli altri livelli della rete *multiplex*, eventualmente specificando il tipo di relazione. Nel nostro caso, la rete *multiplex* è composta da 5 livelli, ciascuno dei quali è caratterizzato da relazioni non-direzionali, dunque non è obbligatorio specificare il tipo di relazione: quest'ultima è infatti assunta non-direzionale di *default* dalla funzione.

Analizzando la struttura del risultato della funzione, osserviamo che questa è una lista composta da 5 oggetti: `nodes` (`data.frame nodi` inserito come primo argomento); `layers` (lista dei 5 oggetti `data.frame` `L1`, `L2`, ..., `L5` riferiti ai livelli della rete *multiplex*); `adjacency` (lista delle 5 matrici di adiacenza associate ai livelli); `type` (vettore con le stringhe che rappresentano il tipo di relazione per ciascun livello) e `layersNames` (vettore con le stringhe che rappresentano le etichette dei livelli).

Utilizzando la funzione `add.interlayer.multiplex` (A.2) è possibile costruire la *supra-adjacency interlayer matrix*

```
M <- add.interlayer.multiplex(M)
```

che viene aggiunta tramite l'oggetto `interlayersMatrix` (matrice 305×305) nell'ultima cella della lista dell'oggetto `multiplex`:

```
> str(M)

List of 6
$ nodes            :'data.frame':        61 obs. of  2 variables:
..$ nodeID   : int [1:61] 1 2 3 4 5 6 7 8 9 10 ...
...
...
$ interlayersMatrix: num [1:305, 1:305] 0 0 0 0 0 0 0 0 0 0 ...
```

Utilizzando la funzione `interlayer.multiplex` (A.3) è poi possibile estrarre una qualunque matrice di adiacenza inter-livello $\mathbf{C}_{\beta}^{\alpha}(\widetilde{hk})$ (nel seguito, si riporta solo uno stralcio della matrice di dimensione 61×61 risultante):

```
> interlayer.multiplex(M, level1 = 1, level2 = 2)

     U102 U139 U33 U106 U107 U118 U123 U1 U21 ...
U102    0    0   0    0    0    0    0  0   0
U139    0    0   0    0    0    0    0  0   0
U33     0    0   0    0    0    0    0  0   0
U106    0    0   0    1    0    0    0  0   0
U107    0    0   0    0    1    0    0  0   0
U118    0    0   0    0    0    0    0  0   0
U123    0    0   0    0    0    0    1  0   0
U1      0    0   0    0    0    0    0  1   0
U21     0    0   0    0    0    0    0  0   1
...
```

Questa matrice permette di osservare, ad esempio, che non esiste una interrelazione per il nodo *U102* tra il livello 1 ed il livello 2 (argomenti `level1` e `level2` della funzione), ma che invece esiste per il nodo `U1`; verosimilmente con quanto assunto, il nodo *U102* è isolato in uno dei due livelli, o addirittura in entrambi.

Allo stesso modo è possibile utilizzare altre funzioni per estrarre altri oggetti dalla rete *multiplex*; mostro nel seguito l'output della funzione `nodes.multiplex` (A.4) che restituisce l'etichetta di 10 nodi scelti a caso:

```
> nodes.multiplex(M, label = T, index = sample(1:61, 10))

[1] "U26"  "U124" "U134" "U37"  "U107" "U73"  "U142" "U48"  "U53" ...
```

Uno strumento particolarmente utile per queste funzioni è la possibilità di indicizzare l'output solo per determinati livelli dell'oggetto `multiplex`, semplicemente specificandone il nome (anzichè l'identificativo); ad esempio, volendo selezionare il tipo di relazione che interviene tra i livelli *coauthor* e *work*, è possibile farlo utilizzando la funzione `type.multiplex` (A.7) con il comando:

```
> type.multiplex(M, index = c("coauthor","work"))

coauthor      work
"undirected"  "undirected"
```

5.3 Estensione delle principali misure per reti monoplex

Una prima analisi esplorativa può essere condotta *parallelamente* livello per livello, senza tenere conto delle relazioni inter-livello, applicando i principali descrittori e le principali misure di *clustering* diffuse per le reti *monoplex*.

Una funzione utilizzabile è la `degree.multiplex` (A.12) che, col seguente comando, permette di ricavare i *degree* di 10 nodi scelti a caso, all'interno di tutti e 5 i livelli della rete *multiplex*:

```
> degree.multiplex(M, indexNode = sample(1:61, 10))

$lunch
U91  U41  U73 U118  U18  U26 U141  U67  U22  U33
7    5    6    6    8    6    4   12    6    6

$facebook
U91  U41  U73 U118  U18  U26 U141  U67  U22  U33
14    0    0    0    8    0    0   13    0    0

$coauthor
U91  U41  U73 U118  U18  U26 U141  U67  U22  U33
3    1    0    2    2    1    0    0    1    0

$leisure
U91  U41  U73 U118  U18  U26 U141  U67  U22  U33
14    2    7    2    4    1    1    2    1    0

$work
U91  U41  U73 U118  U18  U26 U141  U67  U22  U33
8    3    7    9    3   16    2   20    5    9
```

E' anche possibile vedere in che modo le frequenze dei *degree* dei nodi sono distribuite su ciascun livello, utilizzando la funzione (A.13):

```
> degree.distribution.multiplex(M)

$lunch
deg = 0    deg = 1    deg = 2    deg = 3    deg = 4    deg = 5
0.01639344 0.03278689 0.03278689 0.08196721 0.03278689 0.18032787
deg = 6    deg = 7    deg = 8    deg = 9    deg = 10   deg = 11
0.24590164 0.14754098 0.06557377 0.03278689 0.03278689 0.00000000
deg = 12   deg = 13   deg = 14   deg = 15
0.04918033 0.01639344 0.00000000 0.03278689

$facebook
deg = 0    deg = 1    deg = 2    deg = 3    deg = 4    deg = 5
0.47540984 0.00000000 0.04918033 0.00000000 0.06557377 0.06557377
deg = 6    deg = 7    deg = 8    deg = 9    deg = 10   deg = 11
0.06557377 0.03278689 0.03278689 0.01639344 0.03278689 0.06557377
deg = 12   deg = 13   deg = 14   deg = 15
0.03278689 0.03278689 0.01639344 0.01639344

$coauthor
deg = 0    deg = 1    deg = 2    deg = 3    deg = 4    deg = 5
0.59016393 0.24590164 0.09836066 0.03278689 0.01639344 0.01639344

$leisure
deg = 0    deg = 1    deg = 2    deg = 3    deg = 4    deg = 5
0.22950820 0.16393443 0.18032787 0.04918033 0.14754098 0.06557377
deg = 6    deg = 7    deg = 8    deg = 9    deg = 10   deg = 11
0.03278689 0.06557377 0.03278689 0.01639344 0.00000000 0.00000000
deg = 12   deg = 13   deg = 14
0.00000000 0.00000000 0.01639344

$work
deg = 0    deg = 1    deg = 2    deg = 3    deg = 4    deg = 5
0.01639344 0.01639344 0.14754098 0.18032787 0.18032787 0.04918033
deg = 6    deg = 7    deg = 8    deg = 9    deg = 10   deg = 11
0.06557377 0.04918033 0.06557377 0.06557377 0.00000000 0.03278689
deg = 12   deg = 13   deg = 14   deg = 15   deg = 16   deg = 17
0.01639344 0.00000000 0.01639344 0.00000000 0.04918033 0.00000000
deg = 18   deg = 19   deg = 20   deg = 21   deg = 22   deg = 23
0.00000000 0.00000000 0.01639344 0.01639344 0.00000000 0.00000000
deg = 24   deg = 25   deg = 26   deg = 27
0.00000000 0.00000000 0.00000000 0.01639344
```

L'output permette di vedere che, ad esempio, nel livello *lunch* circa 1 nodo su 4 ha 6 nodi nel suo vicinato, mentre nel livello *work* circa il 40% dei nodi hanno tra i 3 ed i 4 nodi nel loro vicinato e un nodo ha un *degree* pari a 27, ad indicare che collabora con molti colleghi. Si può congetturare che l'attore di riferimento abbia un ruolo speciale nella rete, quale, ad esempio, quello di Direttore del Dipartimento. Il livello *coauthor*, invece, è quello in cui i nodi sono più distanti ed isolati tra di loro: al massimo, un nodo è infatti collegato ad altri 5 nodi.

Un'altra funzione utile è `total.degree.multiplex` (A.14) che permette di sommare i *degree* dei nodi su alcuni livelli selezionati; ad esempio col comando

```
> total.degree.multiplex(M, indexNode = 1:10, indexLayer = c("lunch", "facebook"),
                         verbose = T)

Ottenuto con i livelli lunch facebook .
U102 U139 U33 U106  U107 U118 U123 U1   U21   U22
2    1    6   14    8    6    17   13   10    6
```

si ottiene la somma dei *degree* dei primi 10 nodi sui livelli *lunch* e *facebook*. Allo stesso modo, tramite la funzione `mean.degree.multiplex` (A.15) è possibile calcolare la media dei *degree* dei nodi selezionati sui livelli della struttura `multiplex`:

```
> mean.degree.multiplex(M)

$lunch
[1] 6.327869

$facebook
[1] 4.065574

$coauthor
[1] 0.6885246

$leisure
[1] 2.885246

$work
[1] 6.360656
```

Si può notare che nei livelli *lunch* e *work* ciascun nodo è, in media, collegato ad altri 6 nodi; il livello *coauthor* è invece quello in cui la gran parte dei nodi è isolata (in media, ciascun nodo non è collegato neanche ad un solo altro nodo). Questo risultato si interpreta tenendo conto che, mentre la vicinanza fisica quotidiana incentiva le relazioni sociali extralavorative (*lunch*, *leisure*, *facebook*) e una collaborazione professionale (*work*), l'attività di ricerca appare svolgersi prevalentemente con membri esterni al Dipartimento.

Questi dati vanno chiaramente interpretati osservando anche la varianza dei *degree*, tramite la funzione `variance.degree.multiplex` (A.16)

```
> variance.degree.multiplex(M)

$lunch
[1] 9.335125

$facebook
[1] 22.12685

$coauthor
[1] 1.132491

$leisure
[1] 8.036012

$work
[1] 28.03386
```

E' possibile anche osservare la densità dei grafi associati ai livelli tramite la funzione `density.multiplex` (A.17):

```
> density.multiplex(M)

$lunch
[1] 0.1054645

$facebook
[1] 0.06775956

$coauthor
[1] 0.01147541

$leisure
[1] 0.04808743

$work
[1] 0.1060109
```

I risultati confermano quanto già ci si aspettava: *lunch* e *work* sono certamente i due livelli più densi, ma a parità di media dei *degree* (circa 6 per ogni nodo in entrambi i livelli), la varianza del primo è 3 volte inferiore a quella del secondo livello.

La funzione `aggregated.overlapping.multiplex` (A.10) permette di ottenere la matrice di adiacenza associata all'*overlay monoplex network* ottenuto sommando le matrici di adiacenza dei livelli della struttura `multiplex`. Il seguente codice permette, ad esempio, di selezionare le righe e colonne di 10 nodi scelti a caso all'interno di tale matrice, calcolata aggregando tutti e 5 i livelli:

```
> aggregated.overlapping.multiplex(M, indexNode = sample(1:61, 10), verbose = T)

Matrice ottenuta con i livelli lunch facebook coauthor leisure work .
   U97  U134 U72  U142 U138 U14  U107 U17 U49 U92 ...
18   0    0    0    0    0    0    0    0   0   0
27   0    0    0    0    0    0    0    0   0   1
52   0    0    0    0    2    0    0    0   0   0
39   0    0    0    0    0    0    0    0   0   0
48   0    0    2    0    0    0    0    0   0   0
40   0    0    0    0    0    0    0    3   0   0
5    0    0    0    0    0    0    0    3   0   0
20   0    0    0    0    0    3    3    0   0   0
16   0    0    0    0    0    0    0    0   0   0
59   0    1    0    0    0    0    0    0   0   0
...
```

Si può notare che, ad esempio, 3 livelli su 5 esprimono un arco che coinvolge i nodi *U17* (nodo 20) e *U14* (nodo 40); considerazioni analoghe possono essere svolte utilizzando la `aggregated.topological.multiplex` (A.9).

5.4 Misure di eterogeneità

In una rete *multiplex*, risulta particolarmente importante capire se la distribuzione dei *degree* dei nodi è uniforme in tutti i livelli, o se invece ci sono attori che hanno *degree* alti in particolari livelli, e bassi in altri.

Una prima funzione utile in tal senso è la funzione `entropy.degree.multiplex` (A.18). Questa, tramite l'opzione `display = TRUE`, produce anche un grafico che permette una rapida visualizzazione dei risultati. Ad esempio, il comando:

```
> entropy.degree.multiplex(M, display = T)

U102          U139          U33          U106         U107
-0.000499875  0.286669362   0.672878343  1.416940872  1.339482780
U118          U123          U1           ...
1.191693395   0.917871451   1.426148610  ...
```

riporta il vettore dei risultati ed il grafico in Figura 5.2; eventuali segni negativi fanno comunque riferimento a valori di $H(v_i)$ praticamente nulli. In particolare, selezionando i nodi che hanno una misura $H(v_i) < 0.85$, otteniamo la Figura 5.3 (ora, i bastoncini verranno posizionati nelle stesse posizioni degli identificativi dei nodi i cui indici di entropia soddisfano la disuguaglianza, ma in maniera decrescente; anche la scala delle ordinate risulta cambiata).

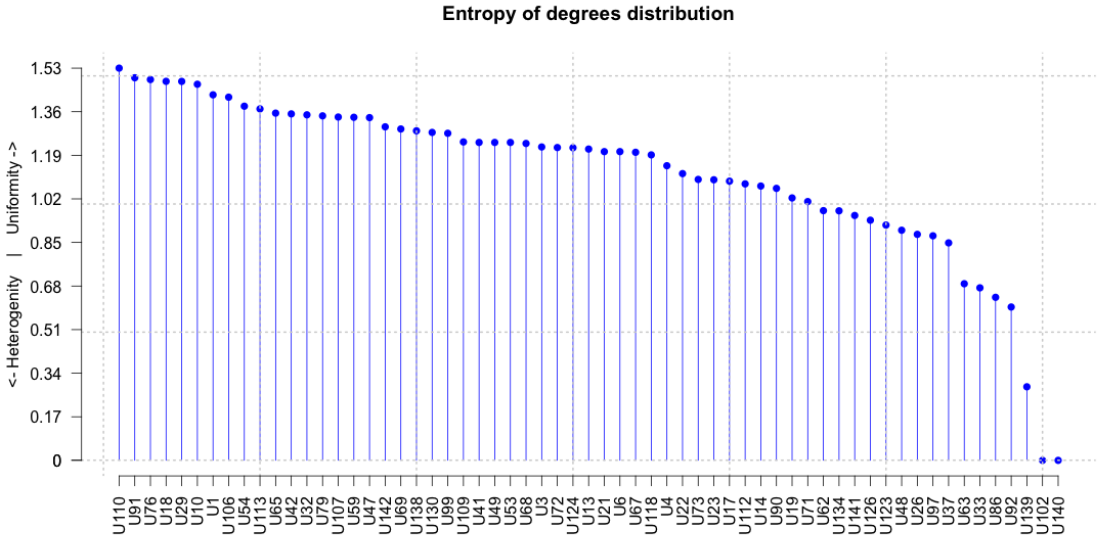

Figura 5.2: Indici di entropia $H(v_i)$

Notiamo, ad esempio, che i nodi *U102* e *U140* risultano avere la distribuzione dei *degree* tra i livelli più eterogenea di tutti gli altri. Nello specifico, i valori dei *degree* sono:

```
> degree.multiplex(M, indexNode = nodes.multiplex(M, index = nodes.multiplex(M,
                    index = (entropy.degree.multiplex(M) < 0.85))))

$lunch
U102 U139  U33  U86  U37  U63  U92 U140
2    1     6    1    6    6    5   0

$facebook
U102 U139  U33  U86  U37  U63  U92 U140
0    0     0    0    0    0    0   0

$coauthor
U102 U139  U33  U86  U37  U63  U92 U140
0    0     0    0    0    0    0   0

$leisure
U102 U139  U33  U86  U37  U63  U92 U140
0    0     0    0    2    0    0   0

$work
U102 U139  U33  U86  U37  U63  U92 U140
0    11    9    2    1    5    2   2
```

e si vede come i nodi *U102* e *U140* siano in effetti connessi in uno solo dei 5 livelli, e isolati nei restanti quattro; inoltre, nel livelli in cui non sono isolati, questi sono in relazione con soli altri due nodi.

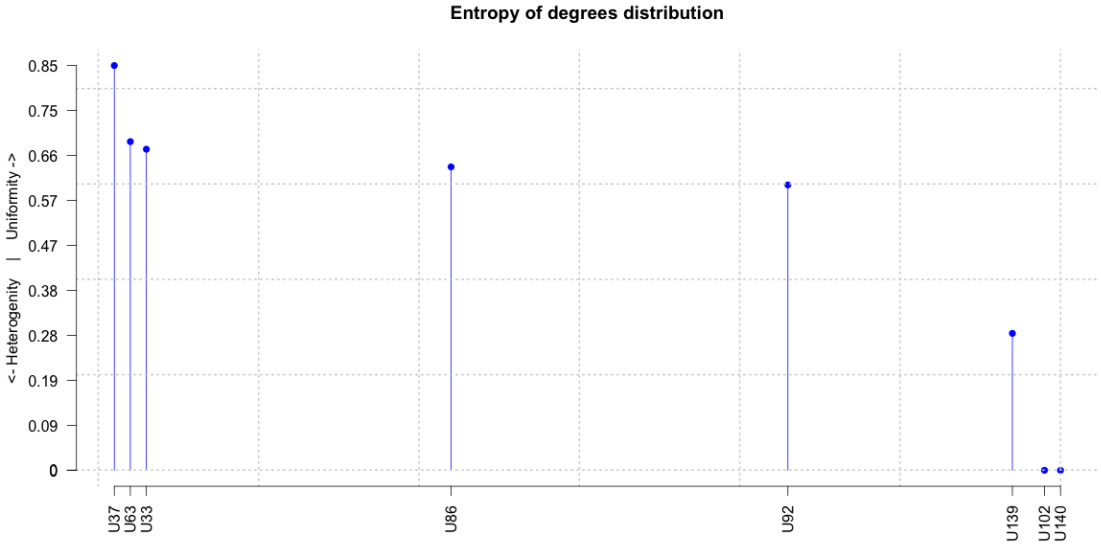

Figura 5.3: Indici di entropia $H(v_i) < 0.85$

I nodi con un valore di $H(v_i)$ più elevato, invece, hanno la seguente distribuzione tra i livelli:

```
> degree.multiplex(M, indexNode = nodes.multiplex(M, index = nodes.multiplex(M,
                   index = (entropy.degree.multiplex(M) > 1.45))))

$lunch
U29  U91  U18  U76  U10 U110
5    7    8    9    3    7

$facebook
U29  U91  U18  U76  U10 U110
5   14    8   10    5    9

$coauthor
U29  U91  U18  U76  U10 U110
1    3    2    2    1    4

$leisure
U29  U91  U18  U76  U10 U110
2   14    4    7    6    7

$work
U29  U91  U18  U76  U10 U110
4    8    3    4    7   14
```

e in particolare il nodo *U110* (che ha l'indice di entropia più alto, pari a 1.53) è connesso in tutti e 5 livelli, con un *degree* pari a 14 nel livello *work*.

Una seconda funzione che permette di ottenere risultati simili è la `participation.degree.multiplex` (A.19):

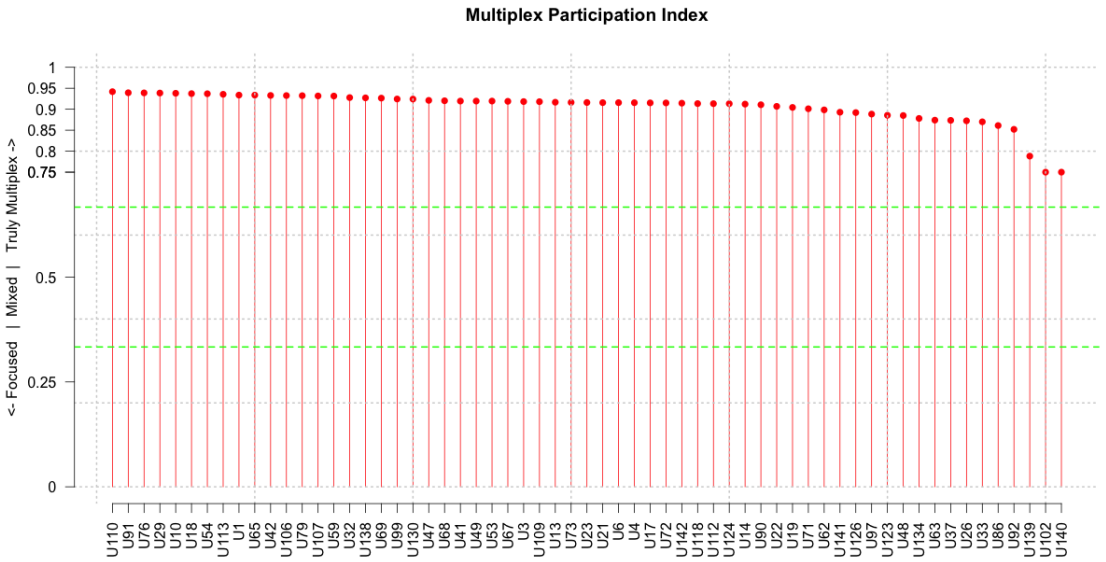

Figura 5.4: Indici di partecipazione $P(v_i)$

```
> participation.degree.multiplex(M, display = T)

U102      U139       U33      U106      U107      U118      U123       U1
0.7500000 0.7881944 0.8700000 0.9325397 0.9316609 0.9134349 0.8855888 0.9336629
U21       U22       U26       U29       U32       U41       U42       U49
0.9156805 0.9068047 0.8723958 0.9385813 0.9277344 0.9194215 0.9328255 0.9194215
...
```

che tramite l'opzione `display = TRUE` permette di ricavare il grafico in Figura 5.4. Notiamo innanzitutto che, bene o male, i nodi con un valore basso di $H(v_i)$ tendono ad avere un valore basso di $P(v_i)$: nonostante ciò, però, tutti e 61 i nodi risultano essere *truly multiplex* perchè stanno sopra alla linea tratteggiata rossa, ovvero hanno un valore di $P(v_i) > 2/3$. L'interpretazione che è possibile dare è che quasi tutti gli attori tendono a *partecipare* in maniera uguale (ovvero, con un numero simile di archi) in ciascun livello della rete *multiplex*.

Un grafico utile da realizzare (codice in Appendice, B.1) è la *heatmap* che confronta le *correlazioni di Kendall* tra i 5 vettori dei *degree* dei livelli della rete *multiplex* con il vettore dei *degree* dell'*overlay monoplex network* (Figura 5.5). Si utilizza la correlazione di Kendall anzichè quella più classica di Pearson, in quanto misura non-parametrica di dipendenza tra due vettori, quando si intende eseguire dei confronti a coppie tra vettori ordinati di variabili quantitative in cui conta la concordanza nell'ordine della singola coppia scelta piuttosto che il valore numerico.

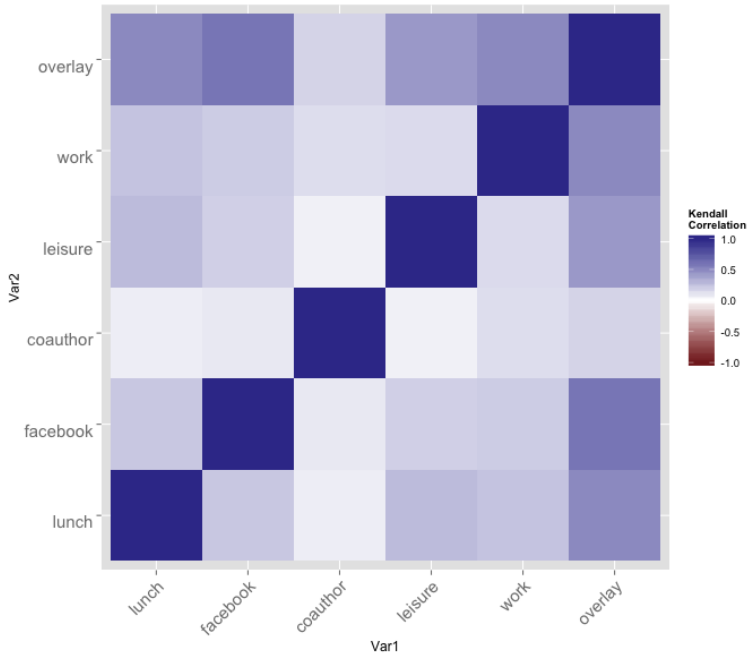

Figura 5.5: Heatmap dei vettori dei *degree* nei singoli livelli e nell'*overlay monoplex network*

Dalla Figura 5.5 si nota immediatamente che le correlazioni più alte si registrano tra i valori dei *degree* dell'*overlay monoplex network* (prima riga/ultima colonna) e quelli dei *degree* dei singoli livelli. Le correlazioni tra coppie di livelli, invece, sono molto basse: in particolare, il livello *coauthor* sembra essere quello meno correlato con tutti gli altri. Questo significa che non necessariamente due colleghi che pubblicano assieme (ovvero, che sono collegati nel livello *coauthor*) hanno altre relazioni di amicizia extra-lavorative, ma probabilmente sono altre le dinamiche che muovono le collaborazioni scientifiche all'interno del Dipartimento.

Un secondo grafico utile per fare confronti tra i valori dei *degree* dei livelli con quelli del *projected monoplex networks* è quello in Figura 5.6, prodotto utilizzando l'interfaccia `muxViz` e gli strumenti grafici da essa forniti. Questo grafico associa a ciascun *settore* degli anelli un nodo (il cui numero identificativo viene riportato a margine dell'anello più esterno) e, per ciascun anello, un colore (secondo la scala riportata in basso) ad ogni settore. I colori più freddi corrispondono a nodi con *degree* alti, mentre quelli caldi corrispondono a nodi con *degree* bassi.

La figura permette di vedere come, in effetti, esista una forte eterogeneità nella distribuzione dei *degree* di ciascun nodo sui vari livelli. Ad esempio, i nodi *U102* e *U140* (rispettivamente associati agli identificativi 1 e 60) che erano risultati avere un valore basso negli indici $H(v_i)$ e

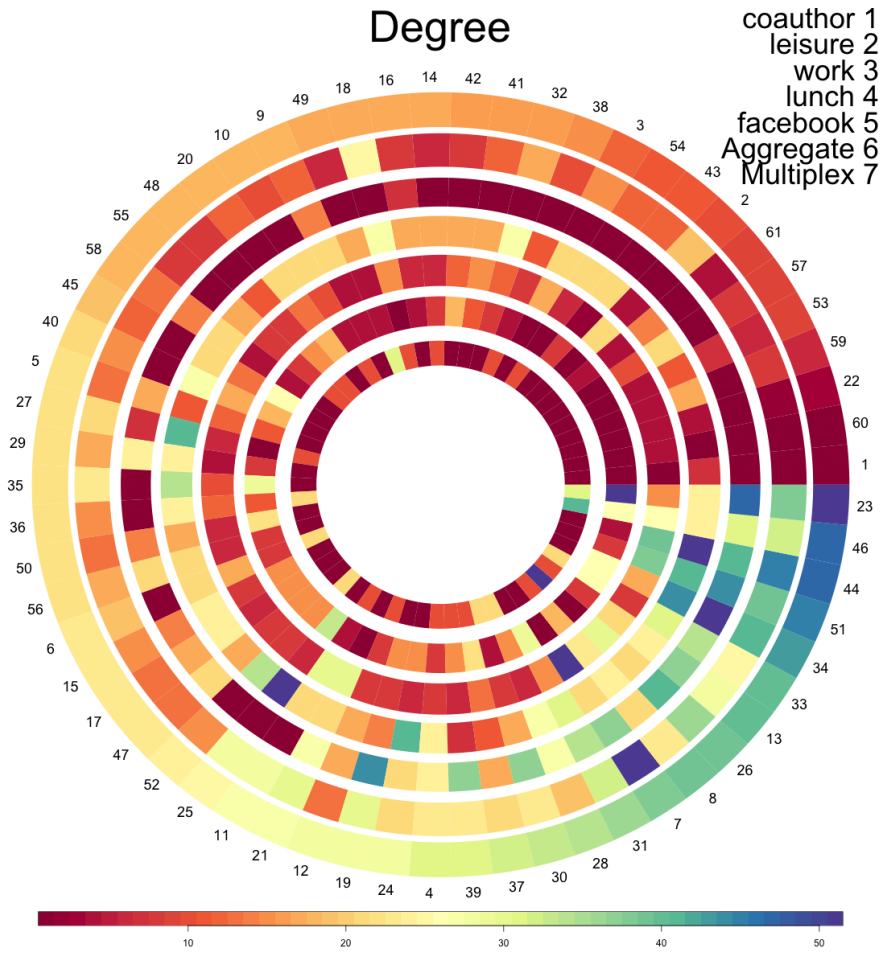

Figura 5.6: Grafico ad anello con i *degree* dei livelli. L'anello esterno rappresenta i *degree* del *projected multiplex network* nel suo complesso (la misura di *multiplex degree centrality*), l'anello più interno fa riferimento al livello *coauthor*

$P(v_i)$ hanno dei settori color magenta in ciascun livello (quindi, *degree* pari a 0) e un unico settore color rosso in un solo livello (in ogni caso, un *degree* molto basso in quest'ultimo, anche se nullo).

Il grafico permette anche di confermare i risultati osservati utilizzando la funzione `mean.degree.multiplex` (A.15): i livelli *coauthor* e *leisure* (corrispondenti ai due anelli più centrali) sono quelli che, in media, hanno dei *degree* molto bassi; questo è confermato dal fatto che quasi tutti i loro settori sono colorati con tinte rosso/magenta. Considerazioni analoghe possono anche essere fatte sui singoli nodi: ad esempio, il nodo con identificativo 25 (in basso a sinistra) risulta avere il *degree* più elevato di tutto il livello *lunch* (settore blu/viola), e il *degree* più basso di tutto il livello *facebook* e di tutto il livello *coauthor* (settore rosso/magenta); ciò significa che, sebbene il nodo 25 (*U109*) sia quello

che più di tutti mangia insieme ad altri colleghi, questo non ha alcun collega tra le proprie amicizie di Facebook e tra le collaborazioni scientifiche. Si potrebbe pensare che, ad esempio, questo attore appartenga al Dipartimento ma faccia parte di un ramo più *amministrativo* e che quindi si trovi con i colleghi solo per pranzare: ulteriori ricerche potrebbero anche mostrare alti indici di centralità locale sul livello *lunch* relativi a quel nodo.

In ogni caso, ricordando che il *degree* è un indice di centralità a tutti gli effetti, possiamo ritenere i nodi sulla parte superiore del grafico come quelli generalmente meno centrali (salvo eccezioni sui singoli livelli), mentre quelli nella parte inferiore come quelli generalmente più centrali (ancora, salvo eccezioni su singoli livelli).

5.5 Misure di transitività per reti multiplex

Un'altra misura particolarmente utile in fase esplorativa è il coefficiente di *clustering*: questo può essere calcolato su ciascun livello e ciascun nodo tramite la funzione `local.clustering.multiplex` (A.20). E' possibile ottenere tali indici associati a 10 nodi scelti a caso, utilizzando il seguente comando:

```
> local.clustering.multiplex(M, indexNode = sample(1:61, 10))

$lunch
U97        U1        U79       U10       U67       U49       U33
0.3928571  0.5714286 0.4358974 1.0000000 0.2727273 1.0000000 0.6666667
U22        U141      U140
0.9333333  1.0000000 NaN

$facebook
U97        U1       U79       U10       U67       U49    U33
NaN 0.3333333 0.3428571 0.5000000 0.3974359 1.0000000    NaN
U22  U141  U140
NaN  NaN   NaN

$coauthor
U97   U1  U79  U10  U67  U49  U33  U22 U141 U140
NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN

$leisure
U97        U1       U79       U10       U67       U49       U33
NaN 0.4000000 0.1428571 0.1333333 0.0000000       NaN       NaN
U22  U141  U140
NaN  NaN   NaN

$work
U97        U1        U79       U10       U67       U49       U33
```

```
0.6071429  0.4090909 0.4722222 0.6666667 0.2052632 1.0000000 0.5833333
U22        U141      U140
0.5000000  1.0000000 1.0000000
```

Notiamo innanzitutto che molti degli indici di *clustering* locale restituiscono un valore `NaN`: questo è legittimo, considerando che alcuni nodi hanno un *degree* nullo su alcuni livelli e che quest'ultimo va a comporre il denominatore di tale indice. In particolare, balzano all'occhio i risultati per il livello *coauthor*: avevamo infatti visto che questo, oltre ad essere il livello meno denso, è anche quello in cui la gran parte dei nodi è isolata. In altre parole, il vettore dei *degree* per questo livello è composto per la maggiore da 0, e pertanto i risultati ottenuti sono plausibili. D'altro canto, il livello *work* (che avevamo visto essere quello più denso) permette di ricavare gli indici di *clustering* locale per i 10 nodi casualmente selezionati; alcuni di questi nodi (*U49*, *U141* e *U140*) hanno addirittura un indice pari ad 1, cioè formano assieme al loro vicinato N_i una *clique*: in altre parole, tutti i nodi appartenenti al vicinato di uno qualunque di questi nodi, sono collegati tra di loro.

A titolo d'esempio, consideriamo il nodo *U141* (identificativo 61). In Figura 5.7 viene utilizzata la funzione `igraph.plot` per plottare il grafo associato al livello *work* e viene evidenziato il vicinato del nodo *U141*, composto dai nodi *U68* e *U4*. E' possibile vedere che il nodo *U141* ha un vicinato di cardinalità 2 (cioè, sul livello *work* ha *degree* pari a 2) e che i due nodi appartenenti a tale insieme sono in relazione tra di loro. In altre parole, questi 3 nodi tendono a formare una *clique* di numerosità 3, ovvero a raggrupparsi tra di loro all'interno del livello *work*; va da sè che nodi con *degree* alti tenderanno ad avere indici di *clustering* locale mediamente bassi, perchè è poco verosimile immaginare che tutti i nodi appartenenti ai loro vicinati siano collegati tra di loro.

Un primo indice di *clustering* che consente di tener conto della complessità informativa gestita dalla rete *multiplex* in esame può essere calcolato dalla funzione `c1.local.multiplex` (A.21). E' possibile ricavare i valori di tale indice su 10 diversi nodi scelti a caso, tramite il comando:

```
> c1.local.multiplex(M, indexNode = sample(1:61, 10))

U23       U42       U91       U32       U68       U106      U118
0.3785714 0.1602564 0.1655983 0.1458333 0.1176471 0.1836735 0.1603774
U67       U37       U53
0.1238806 0.2343750 0.6250000
```

In questo caso, si nota che il nodo *U53* assume un valore piuttosto alto: ciò significa che questo ha una discreta rilevanza nel riuscire a

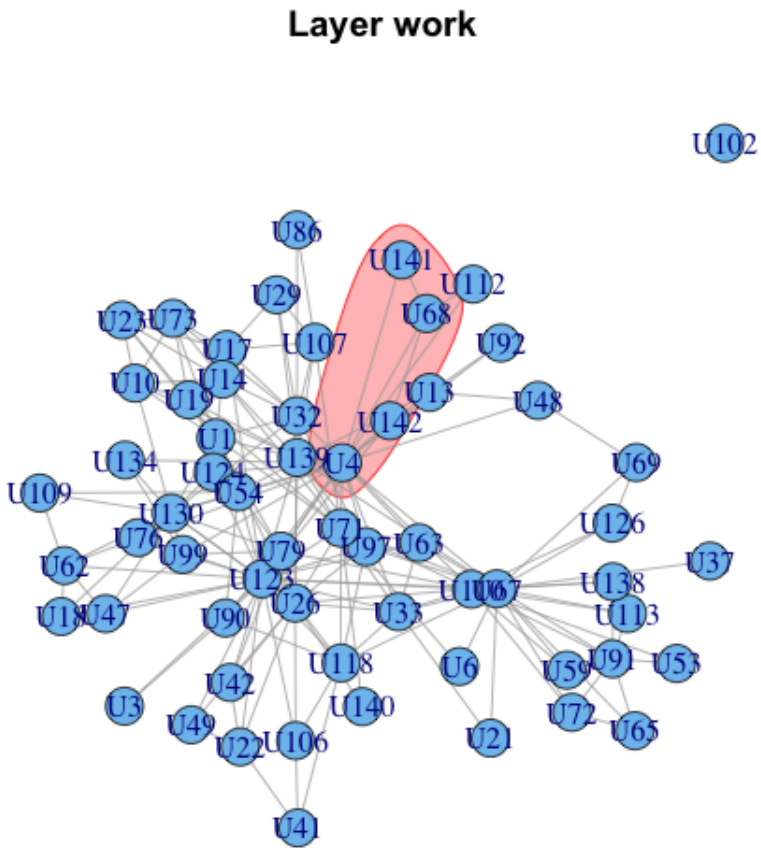

Figura 5.7: Plot del grafo associato al livello *work*, in cui viene evidenziato il vicinato del nodo *U141*

definire delle *semi-cliques* tra i nodi del suo vicinato, tenendo conto delle relazioni inter-livello e delle diverse strutture dei vicinati su ciascun livello della rete *multiplex*.

Allo stesso modo, la funzione `c2.local.multiplex` (A.22) restituisce, con il seguente comando, i risultati su 10 diversi nodi scelti a caso:

```
> c2.local.multiplex(M, indexNode = c(sample(1:61, 10)))

U138         U3       U102        U53        U65        U86       U141
0.15762274 0.07222222 NaN        0.23655914 0.10482180 0.05555556 0.02222222
U76        U130       U139
0.06967465 0.05058538 0.00000000
```

Il calcolo di questo secondo indice evidenzia, ancora una volta, la peculiarità del nodo *U102*, che è uno dei 2 nodi che risultavano avere un valore dell'indice di entropia $H(v_i)$ pari a 0. Richiamando la formula (3.14) che definisce l'indice notiamo che, siccome ciascuno di questi due nodi ha un *degree* nullo in tutti i livelli tranne uno, qualunque prodotto

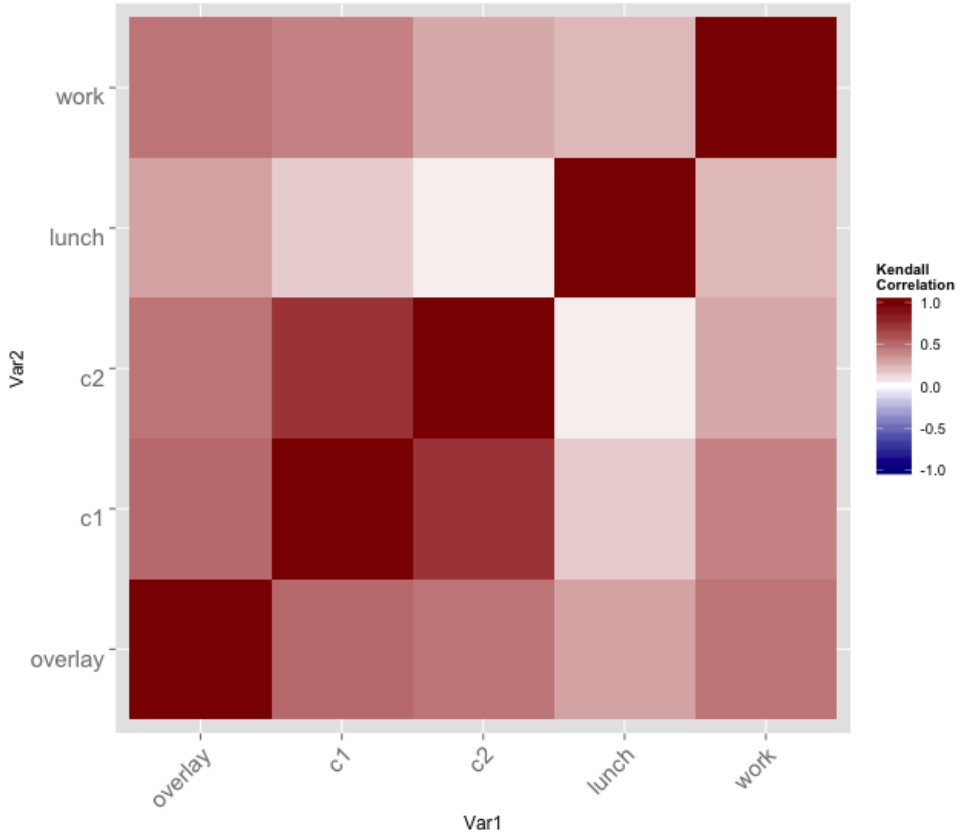

Figura 5.8: Heatmap dei vettori dei coefficienti di *clustering* locale nei livelli *work* e *lunch*, nell'*overlay monoplex network* e nel *multiplex network* tramite gli indici $C_1(n_i)$ e $C_2(n_i)$. Il grafico non tiene conto dei nodi che producono un valore `Nan`.

al denominatore, per qualunque coppia di livelli, sarà certamente nullo e pertanto l'intera somma sarà nulla. Inoltre, il nodo *U53* (che appare anche in questa estrazione degli identificativi degli indici) mantiene ancora una certa rilevanza rispetto agli altri, e tale ne acquistano anche altri nodi (ad esempio, *U138*).

Anche in questo caso, è possibile produrre la *heatmap* (Figura 5.8, codice in Appendice B.2) che mostra le correlazioni di Kendall tra i due indici appena calcolati, gli indici di *clustering* locale dei livelli *work* e *lunch* (quelli più densi), e gli indici di *clustering* locale dell'*overlay monoplex network*. Si nota innanzitutto una marcata correlazione tra i due indici di centralità locale proposti nelle (3.13) e (3.14); ciascuno di questi indici, però, è poco correlato con i due livelli proposti (*lunch* e *work*). In particolare, il livello *lunch* ha una correlazione praticamente nulla con i due indici di *clustering* locale proposti: questo evidenzia, ancora una volta, come spesso lo studio *marginale* dei singoli livelli che compongono

la rete *multiplex* possa portare a risultati differenti (e spesso fuorvianti) rispetto alla realtà. In questo particolare caso, è evidente come i coefficienti di *clustering* calcolati sul solo livello *lunch*, senza tenere conto dei vicinati dei nodi su altri livelli, siano fortemente incorrelati con quelli calcolati tenendone conto. In altre parole è possibile immaginare che i nodi con un elevato indice di *clustering* locale sul livello *lunch* (cioè, che più facilmente di altri riescono a chiudere dei triangoli considerando le diadi presenti nei rispettivi vicinati) non siano necessariamente gli stessi nodi che riescono a chiudere più facilmente di altri dei 2-triangoli o dei 3-triangoli considerando, rispettivamente le 1-triadi o le 2-triadi presenti nei rispettivi vicinati.

Misure di *clustering* globali proposte dalla (3.15) e (3.16) possono essere calcolate utilizzando le funzioni `C1.global.multiplex` (A.23) e `C2.global.multiplex` (A.24), che restituiscono i seguenti valori:

```
> c(C1.global.multiplex(M), C2.global.multiplex(M))

[1] 0.6926840 0.1530809
```

Tali indici sono poco informativi se si considera la sola rete *multiplex*, ma risultano particolarmente utili se invece si desidera fare confronti tra diverse reti. Inoltre, è possibile utilizzare l'argomento `indexLayer` per restringere il calcolo di questi indici includendo solo particolari livelli, e vedere se l'esclusione di alcuni tra questi porta ad un miglioramento o ad un peggioramento dell'indice stesso.

Una seconda misura di *clustering* globale è quella definita dalla (2.36) e implementata dalla funzione `global.overlay.clustering.multiplex` (A.25). In questo caso, il valore restituito è:

```
> global.overlay.clustering.multiplex(M, verbose = TRUE)

Coefficiente di standardizzazione: 1 .
[1] 0.220178
```

ed equivale all'indice di transitività dell'*overlay monoplex network*. Come si vede, l'opzione `verbose = TRUE` permette di restituire anche l'output del coefficiente di standardizzazione; in questo caso, quest'ultimo è pari ad 1 siccome le relazioni sono tutte binarie.

5.6 Indici di multiplex centrality

Altrettanto importanti nello studio di reti *multiplex* sono gli indici di centralità locali, che permettono di dire quanto un particolare nodo è rilevante all'interno del livello nel mettere in relazione nodi non direttamente connessi. Chiaramente, è possibile studiare ognuna di queste misure su ciascun livello separatamente ma, ancora una volta, procedendo in questo modo si tralascerebbe l'informazione risultante dalle relazioni inter-livello e questo potrebbe portare a risultati ben distanti dalla realtà (sopratutto quando queste sono pesate, e hanno pesi diversi a seconda del nodo e della coppia di livelli presi in considerazione).

Una prima misura di centralità, già incontrata in precedenza, riguarda il *degree* di un nodo. Per tenere conto dell'intera rete *multiplex*, ed in particolar modo delle relazioni inter-livello, è possibile ottenere la misura di *multiplex degree centrality* definita nella (3.19) utilizzando la funzione `degree.centrality.multiplex` (A.26) tramite il seguente comando:

```
> degree.centrality.multiplex(M)

U102 U139  U33 U106 U107 U118 U123   U1  U21  U22  U26  U29  U32  U41
   2   14   17   41   29   31   50   51   25   25   36   37   52   23
 U42  U49  U59  U97 U124  U17  U71  U86  U91 U109 U126 U130 U134  U18
  31   23   31   23   37   24   36    5   66   37   33   51   30   45
  U3  U47  U54  U62  U76  U79  U90  U99  U10  U13 U142  U14  U19  U23
  30   43   47   22   52   56   29   30   42   21   41   28   22   22
 U37   U4  U73 U110 U113 U138  U53  U65  U67  U72 U112  U48  U68  U69
  15   61   26   61   31   24   23   30   59   32   13   16   24   29
 U63   U6  U92 U140 U141
  13   25    9    2   13
```

I risultati equivalgono ai *degree* dei nodi calcolati sul *projected monoplex network* definito nella (3.8), e corrispondono agli stessi *degree* calcolati per l'anello più esterno della Figura 5.6.

In generale, i valori restituiti dalla funzione `total.degree.multiplex` (A.14) differiscono da quelli restituiti dalla `degree.centrality.multiplex` (A.26), a maggior ragione quando le relazioni inter-livello hanno pesi diversi. Nel nostro caso, avendo assunto che tutte le relazioni inter-livello sono binarie e che queste esistono ogniqualvolta un nodo non sia isolato in ciascuno dei due livelli legati, i due vettori restituiti sono pressochè identici e altamente correlati.

Una seconda misura di centralità che ha trovato numerose estensioni al caso *multiplex* è la *eigenvector centrality*.

Il primo indice proposto risulta dalla soluzione della (3.25) e può essere calcolato utilizzando la `supra.eigenvector.centrality.multiplex` (A.27): nel seguito riporto il comando per ottenere tale indice su 10 nodi estratti a caso, su tutti e 5 i nodi.

```
> round(supra.eigenvector.centrality.multiplex(M, indexNode = sample(1:61, 10),
        rowStand = T), 5)

             U65    U142    U112     U91     U13     U68     U71
lunch    0.01283 0.01238 0.00595 0.02393 0.00863 0.00722 0.02279
facebook 0.01475 0.04672 0.00621 0.05027 0.00000 0.00000 0.03593
coauthor 0.00000 0.04467 0.00000 0.07122 0.01164 0.01319 0.00000
leisure  0.02612 0.02239 0.00000 0.05344 0.00640 0.00602 0.00000
work     0.00794 0.01362 0.00502 0.01723 0.00759 0.00648 0.03988

             U67     U17     U10
         0.03291 0.00748 0.00900
         0.04861 0.00000 0.02325
         0.00000 0.00000 0.03610
         0.03541 0.01512 0.02357
         0.04009 0.01280 0.01531
```

L'opzione `rowStand = TRUE` garantisce che la somma di ciascuna riga (ovvero, il vettore completo degli indici di *multiplex eigenvector centrality* per ciascun livello) sia pari ad 1.

Il secondo indice proposto fa invece riferimento all'autovettore di Perron calcolato sulla (3.26) e viene implementato dalla funzione chiamata `heter.eigenvector.centrality.multiplex` (A.28). Definiamo innanzitutto la matrice di influenza `W` con diagonale nulla; a questo punto è possibile ricavare gli indici di *multiplex heterogeneous eigenvector centrality* di 10 nodi scelti a caso, tramite il seguente comando:

```
> W <- matrix(1, 5, 5)
> diag(W) <- 0
> round(heter.eigenvector.centrality.multiplex(M, indexNode = sample(1:61, 10),
                                                W = W, rowStand = T), 5)

            U112     U72     U86    U109     U63     U62    U107
lunch    0.00496 0.01703 0.00185 0.01757 0.00632 0.01252 0.01172
facebook 0.00445 0.02936 0.00389 0.02243 0.01561 0.02106 0.00990
coauthor 0.00576 0.02083 0.00313 0.02409 0.01257 0.01696 0.01070
leisure  0.00651 0.02210 0.00354 0.02262 0.01421 0.01643 0.00832
work     0.00564 0.02369 0.00234 0.02806 0.01045 0.01331 0.01064
            U126     U79      U3
         0.01687 0.03773 0.01491
         0.03159 0.03214 0.01338
         0.02544 0.04032 0.01827
         0.01842 0.03914 0.01846
         0.02928 0.04289 0.02207
```

In questo caso, gli indici trovati sono piuttosto *simili* tra i vari livelli, per ciascun nodo: in genere, ha più senso utilizzare questo indice quando la rete *multiplex* ha effettivamente dei pesi sulle relazioni inter-livello che dipendono dalla coppia di livelli considerata nella relazione e, ammettendo che $\widetilde{\mathbf{W}}_{\widetilde{\delta}}^{\widetilde{\gamma}}$ possa non essere necessariamente simmetrica, quando queste sono dirette (tali situazioni fanno comunque riferimento a casi particolari, ad esempio quando si tengono in considerazione *networks* temporali). Ad esempio, cambiando la composizione della matrice di influenza, otteniamo:

```
> W <- matrix(rbinom(25, 5, .5), 5, 5)
> diag(W) <- 0
> round(heter.eigenvector.centrality.multiplex(M, indexNode = sample(1:61, 10),
                                                W = W, rowStand = T), 5)

            U118     U69     U13     U47     U21    U106      U6
lunch    0.00157 0.01862 0.00108 0.04154 0.00830 0.01606 0.00960
facebook 0.00982 0.00881 0.00946 0.01535 0.00776 0.00400 0.00683
coauthor 0.00767 0.01170 0.00751 0.02196 0.00839 0.00760 0.00810
leisure  0.00858 0.01046 0.00812 0.02622 0.00839 0.00834 0.00820
work     0.00327 0.01440 0.00432 0.03008 0.00833 0.01031 0.00795

             U41     U17     U54
         0.00173 0.00434 0.03914
         0.00393 0.01219 0.03243
         0.00299 0.00982 0.03409
         0.00294 0.00796 0.02870
         0.00335 0.00844 0.03920
```

Infine, la *heatmap* in Figura 5.9 (codice in Appendice, B.3) permette alcuni confronti tra le misure di centralità fin qua proposte. La *heatmap* mostra innanzitutto una chiara relazione tra gli indici di *multiplex heterogeneous eigenvector* (avendo assunto che la matrice di incidenza è composta da tutti 1 eccetto la diagonale nulla, ha senso aspettarselo per quanto detto prima), mentre gli indici di *multiplex eigenvector centrality* risultano essere meno correlati tra di loro. Risulta inoltre piuttosto visibile una correlazione relativamente elevata tra questi due indici, ad eccezione di quelli calcolati sul livello *coauthor* che, ancora una volta, appare essere quello più incorrelato con gli altri; va notato anche che i livelli *facebook* e *work* hanno entrambi un indice di *multiplex eigenvector centrality* fortemente correlato con l'indice di *eigenvector centrality* calcolato *marginalmente* sui due livelli: ancora una volta, questo potrebbe dipendere dal fatto che abbiamo assunto le relazioni inter-livello equipesate, e che tale scelta si ripercuote necessariamente sulla *supra-adjacency matrix* con la quale viene calcolato l'indice. Infine, va notato

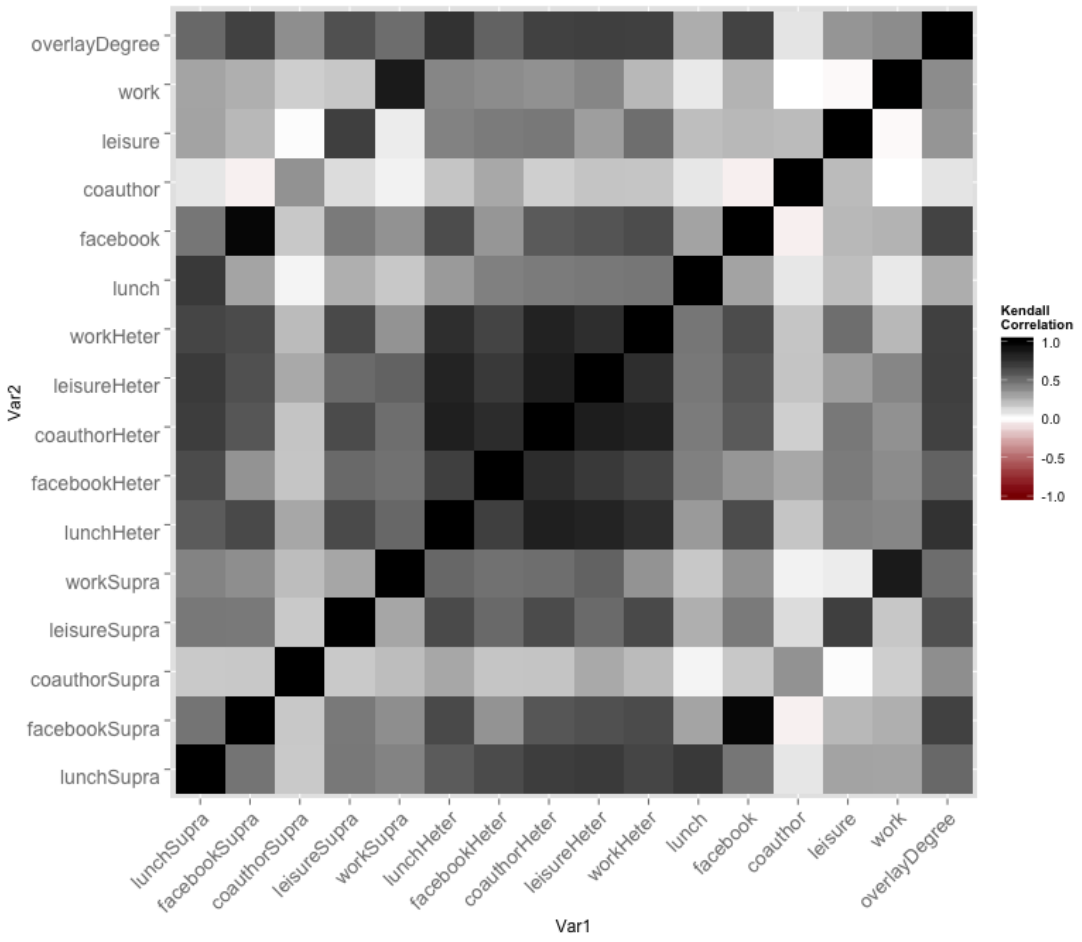

Figura 5.9: Heatmap dei vettori degli indici di *multiplex degree centrality* (A.26), di *multiplex eigenvector centrality* (A.27), di *heterogeneous eigenvector centrality* (A.28) e di *degree centrality* sui singoli livelli

che tutti gli indici (ancora una volta, fatta eccezione del livello *coauthor*) sembrano essere piuttosto correlati con l'indice di *multiplex degree centrality* calcolato dalla (A.26).

Ovviamente, analisi approfondite potrebbero essere svolte tenendo conto di diversi tipi di relazioni inter-livello, escludendo alcuni di questi o particolari nodi, o considerando aggregazioni tramite combinazioni lineari degli indici di *multiplex eigenvector centrality* appena mostrati.

Nel suo complesso, l'analisi esplorativa condotta sulla rete *multiplex* in oggetto mostra l'esistenza di una marcata eterogeneità nelle caratteristiche dei livelli considerati; questa eterogeneità è certamente dovuta, in parte, anche alla natura della relazione considerata in ciascun livello. Ad esempio, i livelli *lunch*, *facebook* e *work* sono quelli in cui, in media, ciascun attore stabilisce più relazioni con altri attori: questo ha

senso se si considera che in genere si tende ad aver stretto l'amicizia su *Facebook* con tutti i conoscenti/colleghi di lavoro, oppure che in un qualunque ambiente di lavoro si tende ad avere interazioni (necessariamente non lavorative) con i propri colleghi; allo stesso tempo, invece, i livelli *leisure* e *coauthor* sono quelli in cui, in media, ciascun attore è collegato con pochi altri colleghi: questa situazione è coerente con l'intuizione che il tempo libero non venga necessariamente trascorso con colleghi di lavoro e che la collaborazione scientifica nasca non solo per effetto della prossimità fisica dovuta al fatto di lavorare entro lo stesso dipartimento ma, più in generale dalla comunanza di interessi di ricerca. Per questa ragione, è più raro instaurare collaborazioni scientifiche con più di uno/due colleghi appartenenti allo stesso dipartimento. La natura di queste relazioni penalizza dunque i livelli meno densi nel calcolo dei principali descrittori, perchè gran parte degli indici (ad esempio, quelli di *clustering*) risultano addirittura difficilmente calcolabili sui singoli livelli.

Una seconda caratteristica emersa è che il tenere conto delle relazioni inter-livello fornisce, in taluni casi, informazioni diverse (o poco correlate) con quelle ricavate senza tenerne conto, o aggregando tutti i livelli in uno unico: ciò suggerisce che in questa rete ha effettivamente importanza il fatto che un collega sia in relazione ad altri colleghi su più livelli, e ciò contribuisce in maniera rilevante allo studio delle misure sviluppate per i contesti *multiplex*. Inoltre, non tutti gli indici sull'eterogeneità della distribuzione dei *degree* nei livelli risultano alti, ad indicare che alcuni di questi sono in relazione con molti altri colleghi in alcuni livelli, e con pochi in altri. Un approfondimento in tal senso potrebbe essere quelli di confrontare le misure di eterogeneità sulla distribuzione dei *degree* con indici di eterogeneità calcolate su altri indici di centralità e capire se a nodi uguali corrispondono uguali valori degli indici.

Un'ultima considerazione da fare è sul livello *coauthor*: come già detto più volte, questo livello è certamente il più problematico in quanto meno denso di tutti gli altri. Potrebbe essere coerente con quanto detto poco fa, svolgere una nuova analisi escludendo questo livello dalla rete *multiplex* e poi confrontare gli indici e le misure calcolate nei due casi separati.

Capitolo 6

Conclusioni e possibili estensioni

Il lavoro che ho svolto all'interno di questa tesi è stato quello di illustrare una precisa formalizzazione matematica per gestire la complessità delle reti *multiplex*: questa formalizzazione risulta da una diretta estensione della notazione sociometrica inizialmente introdotta per le reti *monoplex*, che soffre di alcuni limiti nel prestarsi ad eventuali generalizzazioni. A partire da questa formalizzazione, ho poi cercato di uniformare sotto ad un unico schema notazionale le principali misure descrittive per reti *multiplex*, spesso diffuse in letteratura con nomi e significati diversi a seconda del contesto applicativo entro le quali vengono descritte.

Inoltre, ho sviluppato un pacchetto di funzioni per implementare nell'ambiente R una struttura il più possibile coerente con quanto definito a livello teorico, in grado di poter essere il più possibile generale ed adattabile a qualunque contesto.

Sebbene il pacchetto di funzioni presenti certamente alcuni limiti legati alle mie conoscenze, questo può essere utilizzato per eseguire una prima analisi esplorativa su qualunque rete *multiplex*; ovviamente questa prima fase non esaurisce l'analisi complessiva di una rete *multiplex* ma, anzi, ne costituisce un approccio iniziale puramente descrittivo. L'analisi si completa poi con una fase di modellistica della rete *multiplex* che permette di fare inferenza sulle relazioni esistenti tra i nodi (intra-livello ed inter-livello), non trattata in questa tesi.

Il pacchetto sviluppato può essere certamente esteso in varie direzioni: una prima possibilità è quella di permettere che le relazioni inter-livello possano essere pesate, e che il peso della relazione possa variare non solo da livello a livello, ma anche da nodo a nodo. La funzione (A.2) assume infatti che tutte le relazioni inter-livello siano binarie, e

che queste esistano ogniqualvolta il nodo a cui fanno riferimento non sia isolato in nessuno dei due livelli presi in considerazione: questa assunzione semplifica notevolmente la fase di *input* degli argomenti all'interno della (A.1) perchè consente all'utente di *delegare* alla funzione (A.2) stessa la creazione e la gestione di tali relazioni, ma in alcuni casi questa semplificazione potrebbe non corrispondere alla realtà osservata.

Una seconda estensione, necessaria per completare l'implementazione degli indici di *multiplex centrality* descritti, è quella di sviluppare delle funzioni in grado di calcolare gli indici di *multiplex betweenness centrality* e di *multiplex closeness centrality*: un primo approccio in tal senso è stato proposto da Solé-Ribalta *et al.* (2014), ma utilizzando strutture non direttamente gestibili in ambiente *R*.

Infine, una terza opportunità potrebbe essere quella di permettere di aggiungere nodi, livelli od archi alla rete *multiplex* implementata con la (A.1): questo è possibile solo se si presta particolare attenzione alla natura dei nuovi oggetti che si intendono aggiungere alla struttura `multiplex`, che devono essere coerenti nella loro definizione con gli argomenti della (A.1).

Indice delle funzioni in Appendice

A.1 create.multiplex 93
A.2 add.interlayer.multiplex 95
A.3 interlayer.multiplex 95
A.4 nodes.multiplex 96
A.5 layers.multiplex 96
A.6 adjacency.multiplex 96
A.7 type.multiplex 96
A.8 graph.multiplex 97
A.9 aggregated.topological.multiplex 97
A.10 aggregated.overlapping.multiplex 98
A.11 supra.adjacency.multiplex 98
A.12 degree.multiplex 98
A.13 degree.distribution.multiplex 99
A.14 total.degree.multiplex 99
A.15 mean.degree.multiplex 100
A.16 variance.degree.multiplex 100
A.17 density.multiplex 101
A.18 entropy.degree.multiplex 101
A.19 participation.degree.multiplex 102
A.20 local.clustering.multiplex 102
A.21 c1.local.multiplex 103
A.22 c2.local.multiplex 103
A.23 C1.global.multiplex 104
A.24 C2.global.multiplex 104
A.25 global.overlay.clustering.multiplex 105
A.26 degree.centrality.multiplex 105
A.27 supra.eigenvector.centrality.multiplex 106
A.28 heter.eigenvector.centrality.multiplex 107

Appendice A

Codice delle funzioni sviluppate

Funzione A.1: `create.multiplex`

```
create.multiplex <- function(nodes, layersNames = FALSE, layer1,
    type1, ...){
  N <- nrow(nodes)
  layer1 <- as.matrix(layer1)
  colnames(layer1) <- c("ID1","ID2","Weight")
  layersList <- list(layer1)
  L <- 1

  if(pmatch(type1, c("directed","Directed"), nomatch = 0) > 0){
    type <- "directed"
  }
  else{
  type <- "undirected"
  }

                  ##############################
                  ### Gestione argomento ... ###
                  ##############################

  dots <- list(...)

  for(i in 1:length(dots)){
    if(is.data.frame(dots[[i]]) == TRUE | is.matrix(dots[[i]])){
      if(dim(dots[[i]])[2] == 3 & dim(dots[[i]])[1] <= N*(N-1)){
        if((dots[[i]][,1] %in% nodes[,1]) && (dots[[i]][,2] %in%
            nodes[,1])){
          if(is.numeric(dots[[i]][,3])){
            L <- L + 1
            layersList[[L]] <- as.matrix(dots[[i]])
            colnames(layersList[[L]]) <- c("ID1","ID2","Weight")

            if(i == length(dots)){
              type <- c(type, "undirected")
              break
```

```
33              }
34
35              if(is.character(dots[[i+1]])){
36                if(pmatch(dots[[i+1]], c("directed","Directed"),
                     nomatch = 0) > 0){
37                  type <- c(type, "directed")
38                }
39                else{
40                  type <- c(type, "undirected")
41                }
42              }
43              else{
44                type <- c(type, "undirected")
45              }
46            }
47          }
48        }
49      }
50    }
51
52                    ##############################
53
54    if(is.character(layersNames)){
55      names(layersList) <- layersNames
56    }
57    else{
58      names(layersList) <- sprintf("Layer%i", 1:length(layersList))
59    }
60
61    adjList <- list(NULL)
62    for(j in 1:L){
63      adjList[[j]] <- matrix(0, N, N)
64      for(i in 1:nrow(layersList[[j]])){
65        adjList[[j]][layersList[[j]][,1][i], layersList[[j]][,2][i]]
             <- layersList[[j]][,3][i]
66      }
67      if(type[j] == "undirected"){
68        adjList[[j]] <- adjList[[j]] + t(adjList[[j]])
69      }
70
71      if(sum((diag(adjList[[j]]) != 0) * 1) > 0){
72        diag(adjList[[j]]) <- 0
73        cat("Ho tolto self loops nel livello", j ,".\n")
74      }
75
76    colnames(adjList[[j]]) <- as.character(nodes[, 2])
77    rownames(adjList[[j]]) <- as.character(nodes[, 1])
78    }
79
80    if(is.character(layersNames)){
81      lNames <- layersNames
82      names(adjList) <- lNames
83    }
84    else{
85      lNames <- sprintf("Layer%i", 1:length(layersList))
86      names(adjList) <- lNames
```

```
  }

  multiplex <- list(nodes = nodes, layers = layersList, adjacency =
      adjList, type = type, layersNames = lNames)

  return(multiplex)
}
```

Funzione A.2: `add.interlayer.multiplex`

```
add.interlayer.multiplex <- function(multiplex){
  L <- length(layers.multiplex(multiplex))
  N <- length(nodes.multiplex(multiplex))

  layersList <- layers.multiplex(multiplex)
  combMatrix <- combn(1:L, m = 2)
  interlayerSupraMatrix <- matrix(0, N * L, N * L)

  for(i in 1:ncol(combMatrix)){
    L1 <- combMatrix[1, i]
    L2 <- combMatrix[2, i]

    nodes1 <- as.vector(layersList[[L1]][, -3])
    nodes2 <- as.vector(layersList[[L2]][, -3])
    commonsNodes <- intersect(nodes1, nodes2)

    diagTempMatrix <- matrix(0, N, N)
    for(j in 1:N){
      if(j %in% commonsNodes){
        diagTempMatrix[j, j] <- 1
      }
    }

    interlayerSupraMatrix[(1 + (L1 - 1) * N):(N + (L1 - 1) * N), (1
        + (L2 - 1) * N):(N + (L2 - 1) * N)] <- diagTempMatrix
    interlayerSupraMatrix[(1 + (L2 - 1) * N):(N + (L2 - 1) * N), (1
        + (L1 - 1) * N):(N + (L1 - 1) * N)] <- diagTempMatrix
  }

  multiplex$interlayersMatrix <- interlayerSupraMatrix

  return(multiplex)
}
```

Funzione A.3: `interlayer.multiplex`

```
interlayer.multiplex <- function(multiplex, level1, level2){
  if(level1 == level2){
    stop("La funzione restituisce solo le matrici di adiacenza per
        le interlayer relationships, quindi gli argomenti level1 e
        level2 devono essere diversi tra di loro.\n")
  }

  N <- length(nodes.multiplex(multiplex))

```

```
  interlayerMatrix <- multiplex$interlayersMatrix[(1 + (level1 - 1)
      * N):(N + (level1 - 1) * N), (1 + (level2 - 1) * N):(N + (
     level2 - 1) * N)]
  rownames(interlayerMatrix) <- nodes.multiplex(multiplex, label =
     TRUE)
  colnames(interlayerMatrix) <- nodes.multiplex(multiplex, label =
     TRUE)

  return(interlayerMatrix)
}
```

Funzione A.4: `nodes.multiplex`

```
nodes.multiplex <- function(multiplex, index = 1:nrow(multiplex$
    nodes), label = FALSE){
  if(label == TRUE){
    return(as.character(multiplex$nodes[index, 2]))
  }
  else{
    out <- multiplex$nodes[, 1]
    names(out) <- as.character(multiplex$nodes[, 2])
    return(out[index])
  }
}
```

Funzione A.5: `layers.multiplex`

```
layers.multiplex <- function(multiplex, index = 1:length(multiplex$
    layers), label = FALSE){
  if(label == TRUE){
    return(multiplex$layersNames[index])
  }
  else{
    return(multiplex$layers[index])
  }
}
```

Funzione A.6: `adjacency.multiplex`

```
adjacency.multiplex <- function(multiplex, index = 1:length(
    multiplex$adjacency)){
  return(multiplex$adjacency[index])
}
```

Funzione A.7: `type.multiplex`

```
type.multiplex <- function(multiplex, index = 1:length(multiplex$
    type)){
  out <- multiplex$type
  names(out) <- layers.multiplex(multiplex, label = TRUE)

  return(out[index])
}
```

Funzione A.8: `graph.multiplex`

```
graph.multiplex <- function(multiplex){
  require(igraph)
  G <- list()
  for(i in 1:length(layers.multiplex(multiplex))){
    G[[i]] <- graph.adjacency(adjmatrix = adjacency.multiplex(
        multiplex)[[i]], mode = type.multiplex(multiplex, index = i)
        )
  }
  names(G) <- layers.multiplex(multiplex, label = T)

  return(G)
}
```

Funzione A.9: `aggregated.topological.multiplex`

```
  aggregated.topological.multiplex <- function(multiplex, indexNode
      = 1:length(nodes.multiplex(multiplex)), indexLayer = 1:length
     (layers.multiplex(multiplex)), verbose = FALSE){
  N <- length(indexNode)
  A <- matrix(0, N, N)
  colnames(A) <- nodes.multiplex(multiplex, label = T)[indexNode]
  rownames(A) <- as.character(nodes.multiplex(multiplex))[indexNode
     ]

  for(i in 1:N){
    for(j in 1:N){
      if (i == j) next
      positionX <- indexNode[i]
      positionY <- indexNode[j]
      for(z in indexLayer){
        if(adjacency.multiplex(multiplex)[[z]][positionX, positionY
            ] > 0){
          A[i, j] <- 1
          break
        }
      }
    }
  }

  if(verbose){
    if(length(indexLayer) == 1){
      cat("Matrice ottenuta con il solo livello", layers.multiplex(
          multiplex, label = T)[indexLayer],".\n")
    }
    else{
      cat("Matrice ottenuta con i livelli", layers.multiplex(
          multiplex, label = T)[indexLayer],".\n")
    }
  }

  return(A)
}
```

Funzione A.10: aggregated.overlapping.multiplex

```
aggregated.overlapping.multiplex <- function(multiplex, indexNode
    = 1:length(nodes.multiplex(multiplex)), indexLayer = 1:length
   (layers.multiplex(multiplex)), verbose = FALSE){
  N <- length(indexNode)
  A <- matrix(0, N, N)
  colnames(A) <- nodes.multiplex(multiplex, label = T)[indexNode]
  rownames(A) <- as.character(nodes.multiplex(multiplex))[indexNode
     ]

  for(z in indexLayer){
    A <- A + adjacency.multiplex(multiplex)[[z]][indexNode,
       indexNode]
  }

  if(verbose){
    if(length(indexLayer) == 1){
      cat("Matrice ottenuta con il solo livello", layers.multiplex(
         multiplex, label = T)[indexLayer],".\n")
    }
    else{
      cat("Matrice ottenuta con i livelli", layers.multiplex(
         multiplex, label = T)[indexLayer],".\n")
    }
  }

  return(A)
}
```

Funzione A.11: supra.adjacency.multiplex

```
supra.adjacency.multiplex <- function(multiplex){
  adjList <- adjacency.multiplex(multiplex)
  N <- length(nodes.multiplex(multiplex))
  L <- length(adjList)

  supraMatrix <- multiplex$interlayersMatrix

  for(i in 1:L){
    supraMatrix[(1 + (i - 1) * N):(N + (i - 1) * N), (1 + (i - 1) *
        N):(N + (i - 1) * N)] <- as.matrix(adjList[[i]])
  }

  return(supraMatrix)
}
```

Funzione A.12: degree.multiplex

```
degree.multiplex <- function(multiplex, indexNode = 1:length(
   nodes.multiplex(multiplex)), modeDirected = FALSE){
  require(igraph)
  degreeList <- list()
  for(i in 1:length(layers.multiplex(multiplex))){
    degreeList[[i]] <- degree(graph.multiplex(multiplex)[[i]], v =
       indexNode, mode = "total", loops = FALSE)
    if(type.multiplex(multiplex)[i] == "directed" & modeDirected){
```

```
      degreeList[[i]] <- list(
        Total = degree(graph.multiplex(multiplex)[[i]], v =
            indexNode, mode = "total", loops = FALSE),
        In = degree(graph.multiplex(multiplex)[[i]], v = indexNode,
            mode = "in", loops = FALSE),
        Out = degree(graph.multiplex(multiplex)[[i]], v = indexNode
            , mode = "out", loops = FALSE))
    }
  }
  names(degreeList) <- layers.multiplex(multiplex, label = T)

  return(degreeList)
}
```

Funzione A.13: `degree.distribution.multiplex`

```
degree.distribution.multiplex <- function(multiplex){
  distributionList <- list()
  require(igraph)
  graphList <- graph.multiplex(multiplex)
  for(i in 1:length(layers.multiplex(multiplex))){
    distributionList[[i]] <- degree.distribution(graphList[[i]],
        mode = "total", loops = FALSE)
    names(distributionList[[i]]) <- sprintf("deg = %i", 0:(length(
        distributionList[[i]]) - 1))
  }
  names(distributionList) <- layers.multiplex(multiplex, label = T)

  return(distributionList)
}
```

Funzione A.14: `total.degree.multiplex`

```
total.degree.multiplex <- function(multiplex, indexNode = 1:length(
nodes.multiplex(multiplex)), indexLayer = 1:length(layers.multiplex
(multiplex)), verbose = FALSE){
  require(igraph)
  degreeList <- degree.multiplex(multiplex)

  totalDegree <- rep(0, length(nodes.multiplex(multiplex)))
  names(totalDegree) <- nodes.multiplex(multiplex, label = T)

  for(i in indexLayer){
    totalDegree <- totalDegree + as.vector(degreeList[[i]])
  }

  if(verbose){
    if(length(indexLayer) == 1){
      cat("Ottenuto con il solo livello", layers.multiplex(
          multiplex, label = T)[indexLayer],".\n")
    }
    else{
      cat("Ottenuto con i livelli", layers.multiplex(multiplex,
          label = T)[indexLayer],".\n")
    }
  }
```

```
20
21   return(totalDegree[indexNode])
22 }
```

Funzione A.15: `mean.degree.multiplex`

```
1  mean.degree.multiplex <- function(multiplex, indexNodeMean = 1:
   length(nodes.multiplex(multiplex)), verbose = FALSE){
2    meanList <- list()
3    require(igraph)
4    degreeList <- degree.multiplex(multiplex, indexNodeMean)
5
6    for(i in 1:length(layers.multiplex(multiplex))){
7      meanList[[i]] <- mean(degreeList[[i]])
8    }
9    names(meanList) <- layers.multiplex(multiplex, label = T)
10
11   if(verbose){
12     if(length(indexNodeMean) == 1){
13       cat("Media ottenuta con il solo nodo", nodes.multiplex(
           multiplex, label = T)[indexNodeMean],".\n")
14     }
15     else{
16       cat("Media ottenuta con i nodi", nodes.multiplex(multiplex,
           label = T)[indexNodeMean],".\n")
17     }
18   }
19
20   return(meanList)
21 }
```

Funzione A.16: `variance.degree.multiplex`

```
1  variance.degree.multiplex <- function(multiplex, indexNodeVar = 1:
     length(nodes.multiplex(multiplex)), verbose = FALSE){
2    varList <- list()
3    require(igraph)
4    degreeList <- degree.multiplex(multiplex, indexNodeVar)
5
6    N <- length(indexNodeVar)
7
8    for(i in 1:length(layers.multiplex(multiplex))){
9      varList[[i]] <- (N - 1)/N * var(degreeList[[i]])
10   }
11   names(varList) <- layers.multiplex(multiplex, label = T)
12
13   if(verbose){
14     if(length(indexNodeVar) == 1){
15       cat("Media ottenuta con il solo nodo", nodes.multiplex(
           multiplex, label = T)[indexNodeVar],".\n")
16     }
17     else{
18       cat("Media ottenuta con i nodi", nodes.multiplex(multiplex,
           label = T)[indexNodeVar],".\n")
19     }
20   }
```

```
21
22   return(varList)
23 }
```

Funzione A.17: `density.multiplex`

```
1  density.multiplex <- function(multiplex){
2    densityList <- list()
3    require(igraph)
4    graphList <- graph.multiplex(multiplex)
5
6    for(i in 1:length(layers.multiplex(multiplex))){
7      densityList[[i]] <- graph.density(graphList[[i]], loops = FALSE
         )
8    }
9    names(densityList) <- layers.multiplex(multiplex, label = T)
10
11   return(densityList)
12 }
```

Funzione A.18: `entropy.degree.multiplex`

```
1  entropy.degree.multiplex <- function(multiplex, indexNode = 1:
   length(nodes.multiplex(multiplex)), indexOverlappingLayer = 1:
   length(layers.multiplex(multiplex)), display = FALSE){
2    degreesOverlapped <- degree(graph.adjacency(
       aggregated.overlapping.multiplex(multiplex, indexLayer =
       indexOverlappingLayer), mode = "undirected"), loops = FALSE) #
        Vettore dei degree sull'overlapped
3
4    H <- function(row){
5      last <- length(row)
6      return(-sum((row[-last]/row[last]) * log((row[-last] + 0.001)/
         row[last])))
7    }
8
9    tab <- cbind(simplify2array(degree.multiplex(multiplex))[,
       indexOverlappingLayer], degreesOverlapped)
10   out <- apply(tab, 1, H)[indexNode]
11
12   if(display){
13     plot(indexNode, sort(out, decreasing = T), xlab = "", ylab = "
         <- Heterogenity     |     Uniformity ->", ylim = c(0, max(out)
         ), main = "Entropy of degrees distribution", col = "blue",
         axes = FALSE, type = "h")
14     points(indexNode, sort(out, decreasing = T), pch = 16, col = "
         blue")
15     box(); grid(lwd = 2)
16     axis(1, at = indexNode, labels = names(sort(out, decreasing = T
         )), las = 2)
17     axis(2, at = c(0, seq(min(out), max(out), length.out = 10)),
         labels = as.character(c(0, round(seq(min(out), max(out),
         length.out = 10), 2))), las = 1)
18   }
19   return(out)
20 }
```

Funzione A.19: `participation.degree.multiplex`

```
participation.degree.multiplex <- function(multiplex, indexNode = 1:length(nodes.multiplex(multiplex)), indexOverlappingLayer = 1:length(layers.multiplex(multiplex)), display = FALSE){
  degreesOverlapped <- degree(graph.adjacency(aggregated.overlapping.multiplex(multiplex, indexLayer = indexOverlappingLayer)), loops = FALSE) # Vettore dei degree sull'overlapped

  P <- function(row){
  last <- length(row)
  return(1 - sum((row[-last]/row[last])^2))
  }

  tab <- cbind(simplify2array(degree.multiplex(multiplex))[, indexOverlappingLayer], degreesOverlapped)
  out <- apply(tab, 1, P)[indexNode]

  if(display){
    plot(indexNode, sort(out, decreasing = T), xlab = "", ylab = "<- Focused | Mixed | Truly Multiplex ->", ylim = c(0, 1), main = "Multiplex Participation Index", col = "red", axes = FALSE, type = "h")
    points(indexNode, sort(out, decreasing = T), pch = 16, col = "red")
    box(); grid(lwd = 2)
    axis(1, at = indexNode, labels = nodes.multiplex(multiplex, index = indexNode, label = T), las = 2)
    axis(2, at = c(seq(0, 0.75, by = 0.25), seq(0.75, 1, by = 0.05)), labels = as.character(c(seq(0, 0.75, by = 0.25), seq(0.75, 1, by = 0.05))), las = 1)
    abline(h=c(1/3, 2/3), lwd = 2, col = "green", lty = 2)
  }

  return(out)
}
```

Funzione A.20: `local.clustering.multiplex`

```
local.clustering.multiplex <- function(multiplex, indexNode = 1:length(nodes.multiplex(multiplex))){
  graphList <- graph.multiplex(multiplex)
  require(igraph)
  clusteringList <- list()

  for(i in 1:length(layers.multiplex(multiplex))){
    clusteringList[[i]] <- transitivity(graphList[[i]], type = "local", vids = indexNode)
    names(clusteringList[[i]]) <- nodes.multiplex(multiplex, label = T)[indexNode]
  }

  names(clusteringList) <- layers.multiplex(multiplex, label = T)

  return(clusteringList)
}
```

Funzione A.21: `c1.local.multiplex`

```
c1.local.multiplex <- function(multiplex, indexNode = 1:length(
    nodes.multiplex(multiplex)), indexLayer = 1:length(
    layers.multiplex(multiplex))){
  N <- length(nodes.multiplex(multiplex))
  L <- length(indexLayer)
  if(L < 2) stop("c1 can only be defined for multiplex composed of
      at least 2 layers.")

  adjList <- adjacency.multiplex(multiplex)

  require(gtools)
  couples <- t(permutations(n = L, r = 2, indexLayer))

  numMatrix <- matrix(0, N, N)
  for(i in 1:ncol(couples)){
    numMatrix <- numMatrix + as.matrix(adjList[[couples[1, i]]] %*%
        adjList[[couples[2, i]]] %*% adjList[[couples[1, i]]])
  }

  sumDegree <- rep(0, N)
  for(i in indexLayer){
    sumDegree <- sumDegree + (degree.multiplex(multiplex)[[i]] * (
       degree.multiplex(multiplex)[[i]] - 1)) # Vettore (denom)
  }

  out <- (diag(numMatrix)/((L - 1) * sumDegree))
  names(out) <- nodes.multiplex(multiplex, label = T)

  return(out[indexNode])
}
```

Funzione A.22: `c2.local.multiplex`

```
c2.local.multiplex <- function(multiplex, indexNode = 1:length(
    nodes.multiplex(multiplex)), indexLayer = 1:length(
    layers.multiplex(multiplex))){
  N <- length(nodes.multiplex(multiplex))
  L <- length(indexLayer)
  if(L < 3) stop("c2 can only be defined for multiplex composed of
      at least 3 layers.")
  adjList <- adjacency.multiplex(multiplex)

  require(gtools)
  triples <- t(permutations(n = L, r = 3, indexLayer))
  couples <- t(permutations(n = L, r = 3, indexLayer))

  numMatrix <- matrix(0, N, N)
  for(i in 1:ncol(triples)){
    numMatrix <- numMatrix + as.matrix(adjList[[triples[1, i]]] %*%
        adjList[[triples[2, i]]] %*% adjList[[triples[3, i]]])
  }

  denMatrix <- matrix(0, N, N)
  for(i in 1:ncol(couples)){
```

```
18      denMatrix <- denMatrix + as.matrix(adjList[[couples[1, i]]] %*%
            (matrix(1, N, N) - diag(1, N)) %*% adjList[[couples[2, i
          ]]])
19    }
20
21    out <- diag(numMatrix)/((L - 2) * diag(denMatrix))
22    names(out) <- nodes.multiplex(multiplex, label = T)
23
24    return(out[indexNode])
25  }
```

Funzione A.23: `C1.global.multiplex`

```
1   C1.global.multiplex <- function(multiplex, indexLayer = 1:length(
        layers.multiplex(multiplex))){
2     N <- length(nodes.multiplex(multiplex))
3     L <- length(indexLayer)
4     if(L < 2) stop("C1 can only be defined for multiplex composed of
          at least 2 layers.")
5     adjList <- adjacency.multiplex(multiplex)
6
7     require(gtools)
8     couples <- t(permutations(n = L, r = 2, indexLayer))
9
10    numMatrix <- matrix(0, N, N)
11
12    for(i in 1:ncol(couples)){
13      numMatrix <- numMatrix + as.matrix(adjList[[couples[1,i]]] %*%
            adjList[[couples[2, i]]] %*% adjList[[couples[1, i]]])
14    }
15
16    denMatrix <- matrix(0, N, N)
17    for(i in indexLayer){
18      denMatrix <- denMatrix + as.matrix(adjList[[i]] %*% (matrix(1,
            N, N) - diag(1, N)) %*% adjList[[i]])
19    }
20
21    return(sum(diag(numMatrix))/sum(diag(denMatrix)))
22  }
```

Funzione A.24: `C2.global.multiplex`

```
1   C2.global.multiplex <- function(multiplex, indexLayer = 1:length(
        layers.multiplex(multiplex))){
2     N <- length(nodes.multiplex(multiplex))
3     L <- length(indexLayer)
4     if(L < 3) stop("C2 can only be defined for sysyems composed of at
          least 3 layers.")
5     adjList <- adjacency.multiplex(multiplex)
6
7     require(gtools)
8     triples <- t(permutations(n = L, r = 3, indexLayer))
9     couples <- t(permutations(n = L, r = 3, indexLayer))
10
11    numMatrix <- matrix(0, N, N)
12    for(i in 1:ncol(triples)){
```

```
    numMatrix <- numMatrix + as.matrix(adjList[[triples[1, i]]] %*%
        adjList[[triples[2, i]]] %*% adjList[[triples[3, i]]])
  }

  denMatrix <- matrix(0, N, N)
  for(i in 1:ncol(couples)){
    denMatrix <- denMatrix + as.matrix(adjList[[couples[1, i]]] %*%
        (matrix(1, N, N) - diag(1, N)) %*% adjList[[couples[2, i
       ]]])
  }

  return(sum(diag(numMatrix))/sum(diag(denMatrix)))
}
```

Funzione A.25: `global.overlay.clustering.multiplex`

```
global.overlay.clustering.multiplex <- function(multiplex,
    indexLayer = 1:length(layers.multiplex(multiplex)), verbose =
    FALSE){
  overlayMatrix <- aggregated.overlapping.multiplex(multiplex,
      indexLayer = indexLayer)
  L <- length(indexLayer)
  N <- length(nodes.multiplex(multiplex))
  adjList <- adjacency.multiplex(multiplex)

  max <- 0
  for(i in 1:N){
    for(j in 1:N){
      if(overlayMatrix[i, j] > max){
        max <- overlayMatrix[i, j]
      }
    }
  }

  if(verbose) cat("Coefficiente di standardizzazione:", max/L, ".\n
      ")

  numMatrix <- overlayMatrix %*% overlayMatrix %*% overlayMatrix
  denMatrix <- overlayMatrix %*% (matrix(L, N, N) - diag(L, N)) %*%
      overlayMatrix

  return(sum(diag(numMatrix))/((max/L) * sum(diag(denMatrix))))
}
```

Funzione A.26: `degree.centrality.multiplex`

```
degree.centrality.multiplex <- function(multiplex, indexNode = 1:
    length(nodes.multiplex(multiplex))){

  overlayMatrix <- aggregated.overlapping.multiplex(multiplex)
  N <- length(nodes.multiplex(multiplex))

  interlayersMatrix <- matrix(0, N, N)
  for(i in 1:length(layers.multiplex(multiplex))){
    for(j in 1:length(layers.multiplex(multiplex))){
      if(i == j){
```

```
        next
      }
      else{
        interlayersMatrix <- interlayersMatrix +
           interlayer.multiplex(multiplex, level1 = i, level2 = j)
      }
    }
  }

  projMatrix <- overlayMatrix + interlayersMatrix

  out <- as.vector(projMatrix %*% rep(1, N))
  names(out) <- nodes.multiplex(multiplex, label = T)

  return(out[indexNode])
}
```

Funzione A.27: supra.eigenvector.centrality.multiplex

```
supra.eigenvector.centrality.multiplex <- function(multiplex,
   indexNode = 1:length(nodes.multiplex(multiplex)), rowStand =
   TRUE, testIrreducibility = FALSE, maxPower = 100){
  supraMatrix <- supra.adjacency.multiplex(multiplex)
  N <- length(nodes.multiplex(multiplex))
  L <- length(layers.multiplex(multiplex))

  if(testIrreducibility){
    irreducible <- function(matrix, maxP){
      Mstart <- matrix
      m <- 1
      M <- Mstart
      repeat{
        M <- M %*% Mstart
        m <- m + 1
        if(sum((M == matrix(0, nrow(Mstart), ncol(Mstart)))) == 0){
          return(TRUE)
        }

        if(m == maxP){
          return(list(FALSE, m))
        }
      }
    }

    if(!(irreducible(supraMatrix, maxPower)[[1]])){
      cat("ATTENZIONE: ho fatto", irreducible(supraMatrix, maxPower
         )[[2]], "iterazioni e non e' garantita l'ipotesi di
         irriducibilita' sul Teorema di Perron-Frobenius.\n")
    }
  }

  supraEigenvector <- eigen(supraMatrix)$vectors[,1]
  outMatrix <- matrix(supraEigenvector, nrow = L, ncol = N, byrow =
      T)
  rownames(outMatrix) <- layers.multiplex(multiplex, label = T)
  colnames(outMatrix) <- nodes.multiplex(multiplex, label = T)
```

```
33
34    if(rowStand){
35      outMatrix <- t(apply(outMatrix, 1, function(row) row/sum(row)))
36    }
37
38    if(sign(Re(outMatrix[1,1])) == -1){
39      return((-1) * Re(outMatrix[, indexNode]))
40    }
41    else{
42      return(Re(outMatrix[, indexNode]))
43    }
44  }
```

Funzione A.28: heter.eigenvector.centrality.multiplex

```
1  heter.eigenvector.centrality.multiplex <- function(multiplex,
       indexNode = 1:length(nodes.multiplex(multiplex)), W = matrix(1,
       length(layers.multiplex(multiplex)), length(layers.multiplex(
       multiplex))), rowStand = TRUE, testIrreducibility = FALSE,
       maxPower = 100){
2     N <- length(nodes.multiplex(multiplex))
3     L <- length(layers.multiplex(multiplex))
4     adjList <- adjacency.multiplex(multiplex)
5
6     adjTransposeBlockVector <- do.call(cbind, lapply(adjList, t))
7     adjTransposeBlockVectorMatrix <- do.call(rbind, replicate(L,
          adjTransposeBlockVector, simplify = FALSE))
8     perronMatrix <- kronecker(W, matrix(1, N, N)) *
          adjTransposeBlockVectorMatrix
9
10    if(testIrreducibility){
11      irreducible <- function(matrix, maxP){
12        Mstart <- matrix
13        m <- 1
14        M <- Mstart
15        repeat{
16          M <- M %*% Mstart
17          m <- m + 1
18          if(sum((M == matrix(0, nrow(Mstart), ncol(Mstart)))) == 0){
19            return(TRUE)
20          }
21          if(m == maxP){
22            return(list(FALSE, m))
23          }
24        }
25      }
26
27
28      if(!(irreducible(perronMatrix, maxPower)[[1]])){
29        cat("ATTENZIONE: ho fatto", irreducible(perronMatrix,
              maxPower)[[2]], "iterazioni e non e' garantita l'ipotesi
              di irriducibilita' sul Teorema di Perron-Frobenius.\n")
30      }
31    }
32
33    perronEigenvector <- eigen(perronMatrix)$vectors[,1]
```

```
34
35    outMatrix <- matrix(perronEigenvector, nrow = L, ncol = N, byrow
          = T)
36    rownames(outMatrix) <- layers.multiplex(multiplex, label = T)
37    colnames(outMatrix) <- nodes.multiplex(multiplex, label = T)
38
39    if(rowStand){
40      outMatrix <- t(apply(outMatrix, 1, function(row) row/sum(row)))
41    }
42
43    if(sign(Re(outMatrix[1,1])) == -1){
44      return((-1) * Re(outMatrix))
45    }else{
46      return(Re(outMatrix))
47    }
48
49  }
```

Appendice B

Codice per le heatmap

Codice B.1: Figura 5.5

```
degree <- degree.multiplex(M)
overlayMatrix <- aggregated.overlapping.multiplex(M)
dVectors <- data.frame(lunch=degree$lunch,
                       facebook=degree$facebook,
                       coauthor=degree$coauthor,
                       leisure=degree$leisure,
                       work=degree$work,
                       overlay=degree(graph.adjacency(overlayMatrix,
                           mode="undirected"),v=nodes.multiplex(M)))
library(ggplot2)
library(reshape2)
qplot(x=Var1,y=Var2,data=melt(cor(dVectors, method = "kendall")),
   fill=value, geom="tile") +
  scale_fill_gradient2(limits=c(-1, 1), name = "Kendall\
     nCorrelation") +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, size =
     15, hjust = 1), axis.text.y = element_text(vjust = 1, size =
     15, hjust = 1))
```

Codice B.2: Figura 5.8

```
T <- transitivity(graph.adjacency(aggregated.overlapping.multiplex(
   M), mode = "undirected"), type = "local", vids = 1:61)
c1 <- c1.local.multiplex(M)
c2 <- c2.local.multiplex(M)
lunch <- local.clustering.multiplex(M)[[1]]
work <- local.clustering.multiplex(M)[[5]]

# Tolgo i nodi che restituiscono valori Nan:

clustering <- data.frame(overlay=T[-c(1, 2, 22, 38, 43, 60)],
                         c1=c1[-c(1, 2, 22, 38, 43, 60)],
                         c2=c2[-c(1, 2, 22, 38, 43, 60)],
                         lunch=lunch[-c(1, 2, 22, 38, 43, 60)],
                         work=work[-c(1, 2, 22, 38, 43, 60)])
library(ggplot2)
library(reshape2)
```

```
qplot(x=Var1,y=Var2,data=melt(cor(clustering, method = "kendall")),
    fill=value, geom="tile") +
  scale_fill_gradient2(limits=c(-1, 1), name = "Kendall\
     nCorrelation", low="darkblue", high="darkred", guide="colorbar
     ") +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, size =
     15, hjust = 1), axis.text.y = element_text(vjust = 1, size =
     15, hjust = 1))
```

Codice B.3: Figura 5.9

```
eigenVectors <- round(supra.eigenvector.centrality.multiplex(M,
   rowStand = TRUE), 4)
eigenLunch <- eigenVectors[1,]
eigenFacebook <- eigenVectors[2,]
eigenCoauthor <- eigenVectors[3,]
eigenLeisure <- eigenVectors[4,]
eigenWork <- eigenVectors[5,]

# Indici locali di eigenvector centrality calcolati livello per
   livello

eLunch <- round(evcent(graph.multiplex(M)[[1]])[[1]], 4)
eFacebook <- round(evcent(graph.multiplex(M)[[2]])[[1]], 4)
eCoauthor <- round(evcent(graph.multiplex(M)[[3]])[[1]], 4)
eLeisure <- round(evcent(graph.multiplex(M)[[4]])[[1]], 4)
eWork <- round(evcent(graph.multiplex(M)[[5]])[[1]], 4)

W <- matrix(1, 5, 5)
diag(W) <- 0
heterEigen <- heter.eigenvector.centrality.multiplex(M, W = W)

eigenvec <- data.frame(lunchSupra=eigenLunch,
                       facebookSupra=eigenFacebook,
                       coauthorSupra=eigenCoauthor,
                       leisureSupra=eigenLeisure,
                       workSupra=eigenWork,
                       lunchHeter=heterEigen[1,],
                       facebookHeter=heterEigen[2,],
                       coauthorHeter=heterEigen[3,],
                       leisureHeter=heterEigen[4,],
                       workHeter=heterEigen[5,],
                       lunch=eLunch,
                       facebook=eFacebook,
                       coauthor=eCoauthor,
                       leisure=eLeisure,
                       work=eWork,
                       overlayDegree=degree.centrality.multiplex(M))
qplot(x=Var1,y=Var2,data=melt(cor(eigenvec, method = "kendall")),
   fill=value, geom="tile") +
  scale_fill_gradient2(limits=c(-1, 1), name = "Kendall\
     nCorrelation", low="darkred", high="black", guide="colorbar")
     +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, size =
     14, hjust = 1), axis.text.y = element_text(vjust = 1, size =
     14, hjust = 1))
```

Bibliografia

Barnes J. A. (1954). Class and committees in a norwegian island parish. *Human Relations*, **7**, 39–58.

Barrat A.; Barthélemy M.; Pastor-Satorras R.; Vespignani A. (2004). The architecture of complex weighted networks. *Proceedings of the National Academy of Science*, **101**, 3747–3752.

Battiston F.; Nicosia V.; Latora V. (2014). Structural measures for multiplex networks. **89**(3), 032804.

Beauchamp M. A. (1965). An improved index of centrality. *Behavioral Science*, **10**(2), 161–163.

Bonacich P. (1972). Factoring and weighting approaches to status scores and clique identification. *Journal of Mathematical Sociology*, **2**(1), 113–120.

Brin S.; Page L. (1998). The anatomy of a large-scale hypertextual web search engine. *Computer Networks and {ISDN} Systems*, **30**(1–7), 107 – 117. Proceedings of the Seventh International World Wide Web Conference.

Butts C. T. (2014). *sna: Tools for Social Network Analysis*. R package version 2.3-2.

Csardi G.; Nepusz T. (2006). The igraph software package for complex network research. *InterJournal*, **Complex Systems**, 1695.

De Domenico M.; Solé-Ribalta A.; Omodei E.; Gómez S.; Arenas A. (2013). Centrality in Interconnected Multilayer Networks. *ArXiv e-prints*.

De Domenico M.; Solé-Ribalta A.; Cozzo E.; Kivelä M.; Moreno Y.; Porter M. A.; Gómez S.; Arenas A. (2013). Mathematical formulation of multilayer networks. *Phys. Rev. X*, **3**, 041022.

De Domenico M.; Porter M. A.; Arenas A. (2014). MuxViz: A Tool for Multilayer Analysis and Visualization of Networks. *ArXiv e-prints*.

Einstein A. (1916). Die grundlage der allgemeinen relativitätstheorie. *Annalen der Physik*, **354**(7), 769–822.

Freeman L. C. (1977). A set of measures of centrality based on betweenness. *Sociometry*, **40**(1), 35–41.

Freeman L. C. (1978). Centrality in social networks conceptual clarification. *Social Networks*, **1**(3), 215 – 239.

Holme P.; Saramäki J. (2012). Temporal networks. *Physics Reports*, **519**(3), 97 – 125. Temporal Networks.

Junker B. H.; Schreiber F. (2008). *Analysis of biological networks*. John Wiley & Sons.

Katz L. (1953). A new status index derived from sociometric analysis. *Psychometrika*, **18**(1), 39–43.

Kemper A. (2009). *Valuation of Network Effects in Software Markets: A Complex Networks Approach*. Contributions to Management Science. Physica-Verlag HD.

Kivelä M.; Arenas A.; Barthelemy M.; Gleeson J. P.; Moreno Y.; Porter M. A. (2014). Multilayer networks. *Journal of Complex Networks*, **2**(3), 203–271.

Kolda T. G.; Bader B. W. (2009). Tensor decompositions and applications. *SIAM Review*, **51**(3), 455–500.

Luce R. D.; Perry A. D. (1949). A method of matrix analysis of group structure. *Psychometrika*, **14**.

Magnani M.; Micenkova B.; Rossi L. (2013). Combinatorial Analysis of Multiple Networks. *ArXiv e-prints*.

Moreno J. L. (1946). Sociogram and sociomatrix: A note to the paper by forsyth and katz. *Sociometry*, **1**, 342 – 374.

Opsahl T.; Panzarasa P. (2009). Clustering in weighted networks. *Social Networks*, **31**(2), 155 – 163.

R Development Core Team (2011). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.

Ricci M. M. G.; Levi-Civita T. (1900). Méthodes de calcul différentiel absolu et leurs applications. *Mathematische Annalen*.

Romance M.; Solá L.; Flores J.; García E.; García del Amo A.; Criado R. (2015). A Perron-Frobenius theory for block matrices associated to a multiplex network. *Chaos Solitons and Fractals*, **72**, 77–89.

Sabidussi G. (1966). The centrality index of a graph. *Psychometrika*, **31**(4), 581–603.

Solá L.; Romance M.; Criado R.; Flores J.; García del Amo A.; Boccaletti S. (2013). Eigenvector centrality of nodes in multiplex networks. *Chaos*, **23**(3), 033131.

Solé-Ribalta A.; De Domenico M.; Gómez S.; Arenas A. (2014). Centrality rankings in multiplex networks. In *Proceedings of the 2014 ACM Conference on Web Science*, WebSci '14, pp. 149–155, New York, NY, USA. ACM.

Warnes G. R.; Bolker B.; Lumley T. (2015). *gtools: Various R Programming Tools*. R package version 3.4.2.

Wasserman S.; Faust K. (1994). *Social Network Analysis: Methods and Applications*. Cambridge University Press, New York.

Watts D. J.; Strogatz S. H. (1998). Collective dynamics of 'small-world' networks. *Nature*, **393**, 440–442.