Corso di Laurea Magistrale in Ingegneria Informatica

Studio dell'estrazione di sottografi mutati nei tumori tramite l'analisi di network motifs

Relatore:

Prof. **Fabio Vandin**

Studente:

Luca Venir

Anno accademico 2018-2019

A lei.

Indice

Capitolo 1

Introduzione

In questa sezione viene introdotto brevemente il lavoro svolto nella tesi. Nella Sezione 1.1 viene fornita un'introduzione alla tesi, specificando la disciplina nella quale essa si colloca; viceversa, nella Sezione 1.2, viene fornita una panoramica sul contesto nella quale la tesi si inserisce, specificando anche tutte le nozioni teoriche necessarie per comprendere le finalità del progetto. Nella Sezione 1.3 introdotto il problema sul quale s'accentrano gli sforzi della comunità scientifica nel contesto. Infine, nella Sezione 1.4, viene riportata la sinossi del resto della tesi, al fine di orientarne la lettura.

1.1 Premesse

La tesi si inserisce in una sottoarea del *Machine Learning* (o "Apprendimento Automatico"), chiamata *Pattern Recognition* (o "Riconoscimento di Pattern"). Il *Machine Learning* è, oggi, una delle aree dell'informatica più esplorate e con applicazioni più disparate [1]: queste due discipline hanno subito uno sviluppo sostanziale negli ultimi dieci anni [2].
Apprendimento automatico e riconoscimento automatico di pattern sono ter-

mini strettamente legati: Shalev-Shwartz e Ben-David si riferiscono in generale a "*Machine Learning*" quando si esegue un generico processo di rilevamento automatico di *pattern* all'interno di dati [1]. Distinguere nettamente i due campi non è facile, come evidenziato da Bishop: mentre il *Pattern Recognition* trova origini in tutti i settori dell'Ingegneria, il *Machine Learning* ha origini nella *Computer Science*: tuttavia, queste attività possono essere viste come due facce della stessa medaglia [2].

In generale è lecito affermare che *Machine Learning* è un termine che viene usato quando ci si riferisce agli algoritmi e ai modelli che permettono ai calcolatori di *apprendere* da un insieme di dati di partenza (in un processo detto "fase di *training*"), dove l'apprendimento si concretizza nella conversione di tali dati in conoscenza (o sapienza) a proposito di quei dati [1]. In particolare, nel *Pattern Recognition*, l'*input* sono quindi dei dati (e, sperabilmente, una qualche forma di conoscenza *a priori* a proposito di essi), mentre l'*output* di tali tecniche è una qualche forma di *expertise* sulle informazioni di partenza, che si concretizza con la scoperta automatica di regolarità negli *input* (appunto, *automatic pattern recognition*) [2]: con tali regolarità è possibile effettuare una classificazione di dati in diverse categorie [2], le quali dipendono dal contesto.

Esistono diverse classificazioni del *Machine Learning*. La principale distinzione avviene tra due macro-categorie di *learning*: *supervised* e *unsupervised* learning [1]. Nel primo caso (i.e. nell'apprendimento supervisionato), nella fase di training, l'algoritmo ha a disposizione un feedback immediato sulla qualità del suo *learning*, ovvero un'etichetta sulla corretta natura del dato; l'obiettivo è quello di predire una o più feature sugli input futuri (e.g. classificazione). Nel secondo caso (i.e. nell'apprendimento non supervisionato), l'algoritmo non ha informazioni *a priori* sui dati, ed in effetti l'obiettivo è

piuttosto quello di ricavare nuove informazioni dai dati presi in input (e.g. *clustering*). Esiste inoltre un apprendimento che si pone a metà tra supervisionato e non supervisionato, detto *reinforcement learning* [1]: in questo caso l'obiettivo dell'algoritmo è quello di *rinforzare* (o aggiungere) delle informazioni che, pur essendo già a nostra disposizione, da sole non sono sufficienti nel contesto di applicazione.

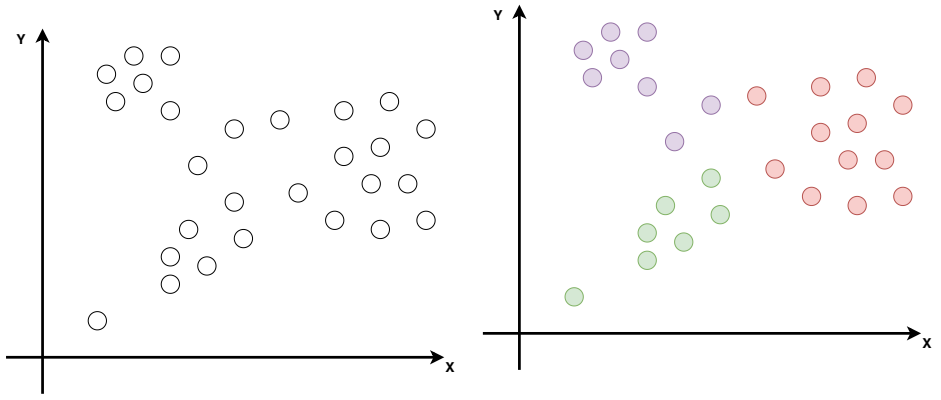
Come esempio di algoritmo di apprendimento automatico supervisionato, prendiamo quello usato quotidianamente per distinguere tra e-mail *classificabili* con due etichette: *spam* e *non spam*. Qui, l'insieme dei dati di partenza è un gruppo di mail già ricevute precedentemente: la loro etichetta è già nota (da qui, apprendimento supervisionato), e nello specifico distinguiamo tra "spam" e "non spam". L'algoritmo quindi *impara* dal gruppo di dati in input al fine di poter fornire, in futuro, una sua classificazione su nuove e-mail.

Come esempio di apprendimento non supervisionato, si prenda come riferimento Figura 1.1a: l'algoritmo non ha a disposizione etichette che gli permettono di *imparare* come distinguere alcuni elementi dagli altri; verosimilmente, anche un essere umano potrebbe eseguire il *clustering* indicato in Figura 1.1b. Per questo motivo l'apprendimento non supervisionato è sicuramente un task più complicato.

La tesi si concentra sullo sviluppo di un sistema di *unsupervised learning*, come si evidenzia in Sezione 1.2.

1.2 Contesto

Il contesto applicativo della tesi è la Bioinformatica, e più nello specifico la "*Cancer Genomics*" (in Italiano, Oncogenomica), la quale è una sottoarea della genomica che analizza i geni associati al cancro; si concentra in par-



a Un possibile *input set*.

b Il clustering ottenuto da un possibile algoritmo.

Figura 1.1: Apprendimento non supervisionato

ticolar modo su alterazioni genomiche nel cancro. In particolare, la *Cancer Genomics* studia le sequenze del DNA e le differenze di espressione genica tra cellule tumorali e cellule ospiti normali (ovvero non mutate) e ha lo scopo di comprendere le basi genetiche della proliferazione delle cellule tumorali e l'evoluzione del genoma del cancro (che avviene per mutazione delle sequenze stesse).

Tecnicamente, "tumore" e "cancro" non sono termini interscambiabili, poiché essi non sono sinonimi: il termine *tumore* si riferisce ad una massa di cellule che ha subito una qualche mutazione nella sequenza del DNA; il termine *cancro*, invece, si riferisce ad un pericoloso e specifico tipo di tumore, spesso identificato con il termine *tumore maligno*, che è resistente ai trattamenti e si diffonde rapidamente nel resto del corpo. In generale, i tumori sono proliferazioni di cellule mutate; sono quindi causati da cambiamenti nella sequenza del DNA delle stesse.

Generalmente, è possibile, naturalmente (nel corso della vita di un individuo), avere delle mutazioni all'interno della sequenza genetica di alcune

cellule, senza terribili conseguenze: tali mutazioni sono dovute a causa di "*errori*" durante la moltiplicazione delle cellule; per porne rimedio, il corpo umano ha in realtà a disposizione dei geni (i "DNA repair genes") che cercano tali errori e li correggono.

Uno dei possibili *iter* che permettono ad una cellula di diventare cancerosa è quando la mutazione avviene proprio nei geni riparatori stessi: in questo modo essi non riescono più a correggere o rilevare mutazioni all'interno della cellula. Esistono, naturalmente, diversi altri modi di sviluppare il cancro, e ciò avviene sempre attraverso la mutazione di specifici gruppi di geni.

In poche parole, esistono delle mutazioni *driver* nella sequenza genetica, poiché *fondamentali* nella creazione e nello sviluppo della massa tumorale; d'altra parte, esistono delle mutazioni non *driver*, ovvero non interessanti, perché di fatto non partecipano attivamente a tale processo.

I tumori sono noti per le alterazioni delle *funzioni* di alcune parti della cellula ospite; ciò che rende difficile l'individuazione delle mutazioni *driver* non è tanto l'individuazione delle singole mutazioni, quanto piuttosto l'individuazione del *gruppo* di mutazioni che alterano una specifica funzione della cellula; tra gli esempi di funzioni delle cellule che più notoriamente mutano in un tumore, vi sono un'insolita accelerazione della divisione cellulare (i.e. riproduzione), la perdita di controllo sulla crescita della cellula o, come detto precedentemente, l'incapacità di rilevamento di tali mutazioni e la loro correzione.

In questo contesto è quindi centrale l'individuazione della funzione cellulare che viene inficiata da una o più mutazioni genetiche; è possibile legare tra loro geni afferenti alla stessa funzione cellulare, andando a creare un *grafo di interazione proteina-proteina*. Tale *grafo* rappresenta le interazioni note tra geni presenti nel genoma umano, interazioni che vanno aldilà dell'individua-

zione di mutazioni o tumori: i vertici del grafo rappresentano il singolo gene, mentre i lati rappresentano la connessione (o il collegamento) che c'è tra una proteina e l'altra.

Degli obiettivi della *Cancer Genomics*, uno è quello di individuare gruppi

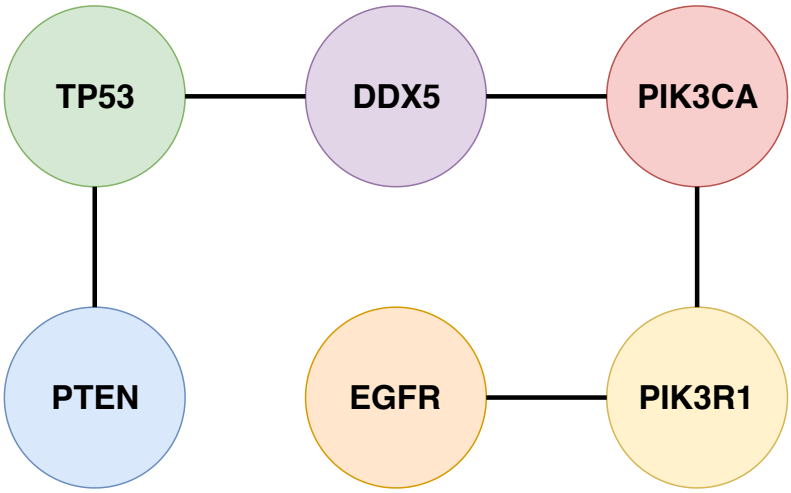


Figura 1.2: Esempio di grafo di interazione proteina-proteina, i cui vertici hanno associato il nome della proteina in esame: se una proteina interagisce con un'altra, v'è un lato che le collega.

(chiamati *pathways*) di geni interconnessi nel grafo di interazione proteina-proteina che, con un certo grado di confidenza, sono *driver* (i.e. determinanti) nello sviluppo tumorale, a partire da campioni biologici; le tecniche che cercano di rispondere a tale obiettivo verranno approfondite in Sezione 2. In breve, nella *computer science*, si vedono i *pathway* come *pattern* da riconoscere automaticamente.

Nelle prossime sezioni, verranno introdotti ulteriori concetti chiave utili per la comprensione dello stato dell'arte del contesto.

1.2.1 Catene di Markov

Le *Catene di Markov* (o "processi Markoviani") sono un modello aleatorio (o un processo aleatorio) che descrivono una ben determinata sequenza di eventi aleatori, in cui la probabilità di passaggio all'evento aleatorio successivo dipende solo dall'ultimo stato raggiunto. Devono il loro nome al matematico Russo Andrey Markov. Di seguito, viene presentato un esempio semplificato di Catena di Markov[1].

L'azienda "OJ" produce succo d'arancia e ha una quota di mercato pari al 20%. Le ambizioni dell'azienda sono quelle di incrementare la propria quota di mercato, e per questo l'amministratore sceglie di implementare una campagna pubblicitaria aggressiva; i consulenti pubblicitari propongono alla società la seguente previsione su base annua: dopo la messa in onda dello spot, i clienti che già bevono il succo "OJ" continueranno a consumare il prodotto, ovvero senza cambiare *brand*, con una probabilità pari a 0.9 (di conseguenza, tali clienti cambieranno idea con probabilità 0.1). Viceversa, i pubblicitari prevedono che chi non consuma "OJ", sarà spinto a spostarsi verso il *brand* aziendale con probabilità 0.7 (viceversa, tali clienti rimarranno fedeli agli altri succhi d'arancia con probabilità 0.3). La situazione nell'esempio viene riassunta dal diagramma degli stati di Figura 1.3 e, più formalmente, attraverso la *Matrice di Transizione*, completa di *Stato Iniziale*.

$$\mathbf{P} = \begin{bmatrix} .9 & .1 \\ .7 & .3 \end{bmatrix} \tag{1.1}$$

$$S_0 = \begin{bmatrix} .2 & .8 \end{bmatrix} \tag{1.2}$$

Si noti come lo stato iniziale indichi sia la quota di mercato dell'azienda OJ, sia il resto del mercato. Al fine di prevedere il cambiamento annuale

[1] L'esempio è tratto da un video didattico su YouTube, il cui ID è: uvYTGEZQTEs

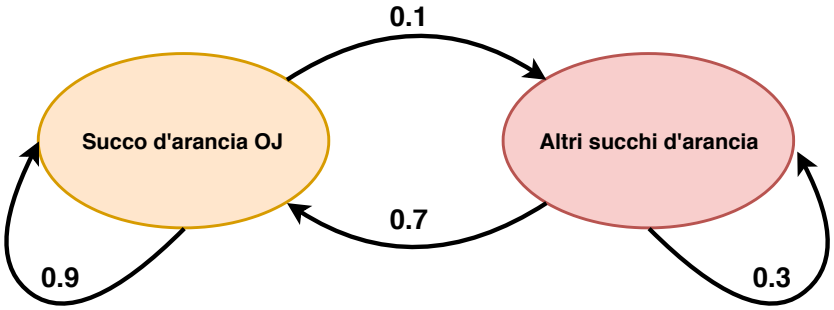


Figura 1.3: La campagna di marketing istituita dai succhi di frutta *OJ*.

delle quote di mercato (i.e. lo stato S_1) è sufficiente applicare la seguente trasformazione lineare:

$$S_1 = S_0 \cdot \mathbf{P} = \begin{bmatrix} .2 & .8 \end{bmatrix} \cdot \begin{bmatrix} .9 & .1 \\ .7 & .3 \end{bmatrix} = \begin{bmatrix} .74 & .26 \end{bmatrix} \tag{1.3}$$

L'equazione 1.3 rappresenta il risultato dell'operazione di marketing effettuata dall'azienda "OJ"; si è passati da uno stato iniziale (equazione 1.2), ad uno stato successivo che a tutti gli effetti dipende dal precedente. A questo punto, l'imprenditore può decidere di ripetere identicamente l'operazione di marketing: la matrice di transizione è la stessa, e al secondo anno risulta:

$$S_2 = S_1 \cdot \mathbf{P} = \begin{bmatrix} .74 & .26 \end{bmatrix} \cdot \begin{bmatrix} .9 & .1 \\ .7 & .3 \end{bmatrix} = \begin{bmatrix} .848 & .152 \end{bmatrix} \tag{1.4}$$

L'equazione 1.4 mostra un'ulteriore miglioramento rispetto all'anno fiscale passato, ma inizia già ad evidenziarsi una perdita d'efficacia della campagna di marketing. Ciò si rinforza ulteriormente se l'imprenditore si ostina a ripetere l'identica operazione anche all'anno successivo:

$$S_3 = S_2 \cdot \mathbf{P} = \begin{bmatrix} .848 & .152 \end{bmatrix} \cdot \begin{bmatrix} .9 & .1 \\ .7 & .3 \end{bmatrix} = \begin{bmatrix} .8696 & .1304 \end{bmatrix} \tag{1.5}$$

L'effetto sempre più contenuto della trasformazione lineare che si osserva nell'equazione 1.5 è dovuto alla *stabilità* del processo stocastico di Markov in

esame; da qui è possibile individuare una *distribuzione stazionaria* di una Catena di Markov, ovvero il vettore S_∞ che si ottiene ripetendo infinite volte l'operazione stocastica. Più semplicemente, è possibile ottenere la distribuzione stazionaria cercando quel vettore che se trasformato attraverso $\mathbf{P}$ rimane se stesso; più brevemente, il vettore S soddisfa:

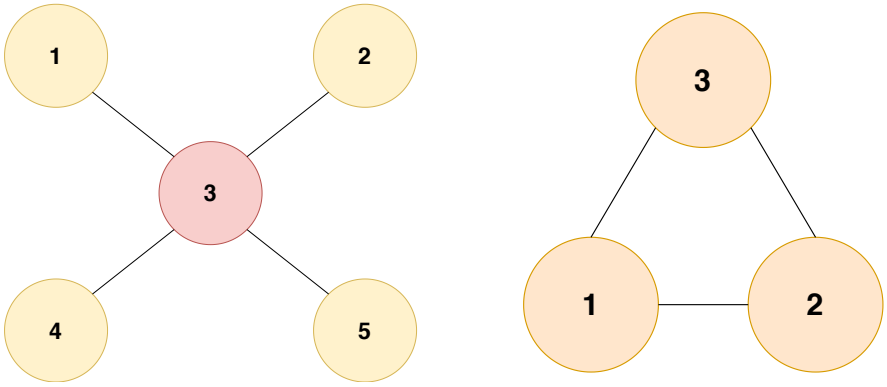
$$S \cdot \mathbf{P} = S$$

1.2.2 Random Walks

Le *random walks* sono particolari Catene di Marvok che descrivono una successione di *random steps* in un certo spazio matematico discreto. Nel contesto della tesi, lo spazio matematico discreto considerato è un grafo non diretto $G = (V, E)$, mentre uno *step* è un possibile spostamento da un nodo ad un suo vicino nel grafo; la random walk è, quindi, una successione di *step casuali* su vertici del grafo. Più in particolare, nel contesto della tesi vengono considerate delle random walks su grafi non diretti *con restart*. Esse sono quelle random walk che, partendo da un certo vertice s detto sorgente, eseguono, con probabilità α, uno step verso un vicino del nodo corrente scelto casualmente in maniera uniforme; di conseguenza, la probabilità di passare da un nodo i ad un nodo j dipende dal grado di i, che influenza la scelta casuale uniforme da effettuare. Viceversa, con probabilità $1 - \alpha$, la walk ricomincia dal nodo sorgente s (da qui, *restart*). Per descrivere il processo senza il restart è sufficiente la matrice di transizione $\mathbf{W}$, i cui elementi sono:

$$w_{ij} = \begin{cases} \frac{1}{deg(i)}, \text{ se il nodo i è adiacente a j} \\ 0 \text{ altrimenti} \end{cases} \tag{1.6}$$

Il parametro α, invece, controlla la "località" della walk: più è basso, più è probabile un *restart*. A scopo di esempio, si usi come riferimento il grafo



a Struttura *a stella* b Struttura lineare

Figura 1.4: Grafi non diretti

nella Figura 1.4. Supponendo di voler eseguire una *random walk con restart* sul grafo a sinistra (Figura 1.4a), se si scegliesse come *sorgente* s uno tra i nodi 1, 2, 4 e 5, le possibilità "di movimento" sono limitate; il grafo è a stella: è possibile muoversi solamente verso il nodo 3 con probabilità 1. Viceversa, dal nodo 3 è possibile raggiungere tutti gli altri con probabilità 0.25. Di conseguenza, la matrice di transizione $\mathbf{W}$ associata al processo descritto è:

$$\mathbf{W} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0.25 & 0.25 & 0 & 0.25 & 0.25 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}. \tag{1.7}$$

Osservando la matrice, non sorprende trovare nulle tutte le entrate (i, j) della matrice $\mathbf{W}$ che corrispondono a lati i, j assenti nel grafo (inclusi i cappi, che non possono essere presenti in un grafo non diretto). Un'altra importante considerazione va fatta sulla non-simmetria della matrice $\mathbf{W}$, a differenza della matrice di adiacenza di un grafo non diretto: tale caratteristica è causata dai diversi gradi dei vertici del grafo.

Si consideri ora il grafo a destra in Figura 1.4b; in questo caso i nodi sono tutti adiacenti tra loro, facendo risultare equiprobabile lo spostamento da un vertice ai suoi vicini: la matrice di transizione $\mathbf{W}$ diventa:

$$\mathbf{W} = \begin{bmatrix} 0 & 0.5 & 0.5 \\ 0.5 & 0 & 0.5 \\ 0.5 & 0.5 & 0 \end{bmatrix}. \tag{1.8}$$

Si noti come la matrice in equazione 1.8 è simmetrica: questa pura casualità è data dalla particolare struttura del grafo preso in esame.

Anche le random walks con restart su grafi non diretti ammettono stazionarietà: partendo da una sorgente g ed effettuando infiniti *step aleatori uniformi* (con probabilità di restart $1-\alpha$), si ottiene il vettore distribuzione stazionaria s_g, le cui i-esime componenti rappresentano la probabilità di finire sul nodo i-esimo. L'equazione che descrive la distribuzione stazionaria del processo è la seguente:

$$\alpha \mathbf{W} s_g + (1-\alpha) \cdot e_g) = s_g \tag{1.9}$$

dove e_g è il vettore con 1 nella componente in posizione g, e 0 nelle altre. L'equazione 1.9, risolta per s_g, diventa:

$$s_g = (1-\alpha)(\mathbf{I} - \alpha \mathbf{W})^{-1} \cdot e_g \tag{1.10}$$

L'equazione 1.10 evidenzia la possibilità di ottenere la distribuzione stazionaria s_g a partire da $\mathbf{W}$ e α; s_g è infatti quel vettore che ha per i-esima componente la probabilità di finire, dopo infiniti step, sul vertice i del grafo attraverso una random walk con restart (avendo come sorgente, il vertice g). Nel contesto della tesi, si è interessati a ripetere l'analisi considerando tutti i vertici del grafo in esame come possibili sorgenti. In tal caso si ottiene, con analoghe considerazioni, quella che viene chiamata "matrice di diffusione" $\mathbf{F}$, nella quale la (i, j)-esima entrata di tale matrice rappresenta la probabilità di

passare al nodo j-esimo a partire dal nodo sorgente i. La matrice è ricavabile dall'equazione 1.11.

$$\mathbf{F} = (1 - \alpha)(\mathbf{I} - \alpha\mathbf{W})^{-1}. \tag{1.11}$$

Si noti, infine, come la g-esima colonna di $\mathbf{F}$ è esattamente s_g.

1.2.3 Graph Clustering

Con *clustering* ci si riferisce ad una delle più note tecniche di Unsupervised Machine Learning: in generale, un algoritmo che esegue un clustering su un certo *dataset* ha il compito di raggruppare un certo insieme di oggetti in modo che oggetti "simili" finiscano negli stessi gruppi e allo stesso modo oggetti "diversi" vengono raggruppati in gruppi diversi. Il clustering è la tecnica di unsupervised learning per eccellenza, perché il suo compito è proprio quello di dare un senso a dei dati senza etichette; oltre alla difficile natura del problema, non v'è un chiaro metodo per valutare e distinguere tra un "buon" clustering e uno "cattivo".

Più rigorosamente, è possibile definire il clustering come il seguente *task*.

Dato un generico insieme di elementi, $\mathcal{X}$, ed una metrica di distanza su di esso, $d\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$ (tale che sia simmetrica e che $d(x, x) = 0$, $\forall x \in \mathcal{X}$), si vuole ottenere una partizione del dominio $\mathcal{X}$ in dei sottoinsiemi $C = (C_1, C_2, ..., C_k)$. La valutazione della *qualità* dei cluster $C_1, ..., C_k$ ottenuti avviene proprio attraverso l'uso della funzione distanza d; a volte, in alternativa, si può definire una funzione di similarità $s\colon \mathcal{X} \times \mathcal{X} \to [0, 1]$. Unico requisito per s è $s(x, x) = 1$, $\forall x \in \mathcal{X}$.

Tra le svariate applicazioni, la tesi prende in analisi il clustering su grafi; a tutti gli effetti, come menzionato in Sezione 1.2, i lati dei grafi presi in esame misurano la *similarità* tra proteine all'interno del genoma umano. Il clustering su grafi, dato un grafo $G = (V, E)$, ha l'obiettivo di trovare una (o

più) partizioni sul grafo in modo che i lati che connettono nodi su partizioni diverse abbiano peso basso, mentre i lati che connettono nodi sulle stesse partizioni abbiano peso alto.

Su un clustering su un grafo, è possibile usare la *conduttanza* come metrica di valutazione della qualità dei cluster, metrica definita su altre due misure, il *taglio* e il *volume* su un insieme di vertici S. Dato un insieme di vertici $S \subseteq V$, il taglio $cut(S)$ contiene il numero di lati che hanno un terminale in S, e l'altro in $\bar{S} = V \setminus S$. Allo stesso tempo, il volume su un insieme di vertici $S \subseteq V$ è il numero di terminali all'interno di S, ovvero $vol(S) = \sum_{u \in S} d_u$, dove d_u è il grado del nodo u.

La conduttanza ϕ su un cluster S di nodi è quindi data da:

$$\phi(S) = \frac{cut(S)}{vol(S)}. \tag{1.12}$$

Si noti come la conduttanza misura il rapporto tra i lati che connettono dentro il cluster con quelli fuori dal cluster, rispetto al numero di terminali totali del cluster; se la conduttanza è bassa, il cluster S è di buona qualità, perché pochi lati connettono S con $\bar{S}$ (in altre parole, il taglio è piccolo), e viceversa ci sono molti terminali all'interno di S (ovvero il cluster è ben connesso). Viceversa, quando la conduttanza è alta, il cluster S non è di buona qualità poiché non è ben isolato da $\bar{S}$ o perché non è molto coeso con i nodi al suo interno. Alternativamente, è possibile interpretare la conduttanza come una probabilità: $\phi(S)$ è quindi la probabilità che, scegliendo casualmente un lato adiacente ad nodo u scelto a caso in S, porti al di fuori di S, cioè in $\bar{S}$ (assumendo che il volume del cluster è almeno la metà del volume totale del grafo).

Nella Figura 1.5(A) v'è un esempio di calcolo della conduttanza di un cluster all'interno di un piccolo grafo.

I grafi hanno per natura una struttura discreta, ed è possibile interpretarla

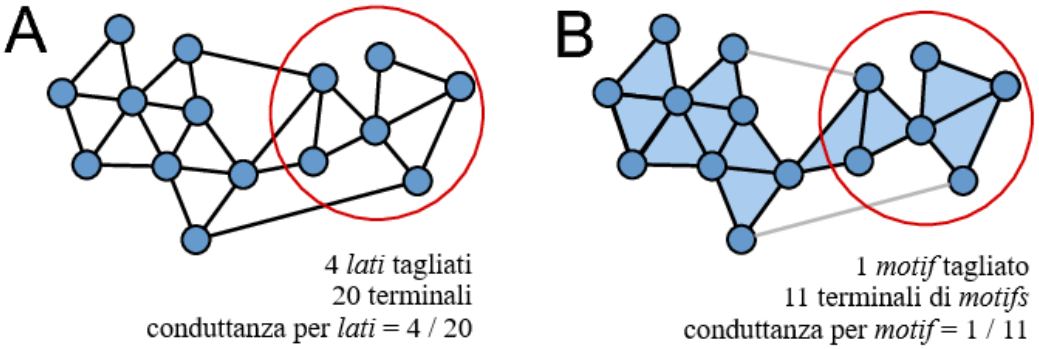


Figura 1.5: Conduttanza misurata su tagli di lati (A) e su tagli di motifs (B) [3]

geometricamente: ad esempio, all'interno di un grafo non diretto, è possibile visualizzare un "triangolo", un "quadrato" o forme più complicate, che altro non sono che sotto-grafi con una struttura ben definita; nell'esempio in esame, i "triangoli" sono quei sotto-grafi composti da tre vertici vicini tra loro (i.e. sotto-grafi completi da tre vertici), mentre i "quadrati" sono quei sotto-grafi composti da quattro vertici con due vicini a testa, tali che l'unico vertice non adiacente ad essi è adiacente ad entrambi i propri vicini. Tali particolari sotto-strutture nei grafi vengono chiamate *graphlets* o *motifs*, rispettando il linguaggio del Machine Learning e del Data Mining: tali *motif* sono proprio delle regolarità nei dati, che spesso, come menzionato precedentemente, si è interessati ad individuare.

Come esempi di possibili *network motifs* all'interno di grafi non diretti, oltre ai già precedentemente menzionati triangoli e quadrati, troviamo anche le cliques di taglia k (i.e. sotto-grafi completi di taglia k). Naturalmente, i possibili motifs sono infiniti; tuttavia ogni motif, indipendentemente dalla sua struttura, può essere esteso tramite una "coda", ovvero un lato e un vertice, adiacenti ad un qualsiasi terminale del motif preso in considerazione.

Il clustering su grafi può essere quindi esteso, in quanto può avere come obiettivo quello di raggruppare vertici tenendo in considerazione le strutture ad

alto livello (come i triangoli) presenti all'interno del grafo in esame.
L'*higher-order graph clustering* è quel particolare tipo di clustering su grafi che cerca una partizione ottimale non più rispetto ai lati tagliati (e terminali degli stessi), ma rispetto ai *motifs* tagliati (e terminali di questi) [3]. Si prenda come esempio il grafo in Figura 1.5(B); all'interno di esso è possibile contare undici triangoli, ed è possibile contare anche altri motifs di differente natura (e.g. quadrati, cliques con coda, etc.): l'*higher-order graph clustering* si occupa di trovare il modo migliore per raggruppare i vertici di quel grafo cercando di tagliare meno *motifs* possibili e raggruppandone il numero più alto possibile.
Anche per questo tipo di clustering su grafi è possibile usare una metrica di valutazione della qualità dei cluster simile a quanto esaminato fin'ora: dato un insieme $S \subseteq V$ e un motif M, la *conduttanza per motifs*, $\phi_M(S)$, prende in considerazione il *taglio per motifs*, $cut_M(S)$, e il *volume per motifs*, $vol_M(S)$. $cut_M(S)$ è il numero di istanze di M che hanno almeno un terminale (i.e. un nodo) in S e almeno uno in $\bar{S}$ (ricordando che $\bar{S} = V \setminus S$). $vol_M(S)$ è il numero di istanze di M i cui terminali sono in S, ovvero il numero di volte che i nodi in S partecipano ad un'istanza di M. Di conseguenza, la conduttanza ϕ_M su un cluster S di nodi è data da:

$$\phi_M(S) = \frac{cut_M(S)}{vol_M(S)}. \tag{1.13}$$

Anche in questo caso la conduttanza espressa nell'equazione 1.13 esprime la coesione (o l'isolamento) del cluster all'interno del grafo, dove tuttavia tale misura è ricavata dai motif coinvolti, ovvero da una struttura di ordine superiore e non solo dai lati del grafo; anche per l'*higher-order graph clustering*, quando $\phi_M(S)$ è bassa, il cluster S è di buona qualità, e viceversa. Con l'interpretazione probabilistica, $\phi_M(S)$ è la probabilità che, scegliendo

casualmente un terminale di un motif M adiacente ad un nodo u scelto a caso in S, si finisca al di fuori di S, ovvero in $\bar{S}$.

1.3 Problema

In Sezione 1.2 si è introdotto il concetto di grafo di interazione proteina-proteina: uno degli obiettivi della *Cancer Genomics* è quello di individuare *pathways driver* per lo sviluppo di masse tumorali; per far ciò è fondamentale un dataset di campioni biologici, che contengono informazioni circa l'avvenuta mutazione o meno di alcune proteine in quel campione.
L'obiettivo che si pone la tesi è quello di utilizzare la tecnica più nota dell'*unsupervised learning* su tale dataset, ovvero il *clustering*, avendo già a disposizione una struttura ben definita (il grafo di interazione proteina-proteina), al fine di ottenere possibili nuovi gruppi di geni fondamentali per il cancro (i.e. *pathways driver* nei tumori). Più brevemente: l'obiettivo della tesi è di estrarre nuovi sotto-grafi mutati a partire da un campione, tramite l'analisi di grafi proteina-proteina e sfruttando le tecniche citate in 1.2.3 (*higher-order graph clustering*).
In prima approssimazione, il campione sopracitato può essere pensato come un elenco di mutazioni avvenute nei singoli pazienti. In realtà, come viene evidenziato in Sezione 2.1, il dataset considerato è un elenco di proteine alle quali viene associato uno *score* che dipende dalle sopracitate mutazioni.
Infine è di fondamentale importanza validare i risultati ottenuti, sia dal punto di vista statistico (i.e. dimostrare che i nuovi sotto-grafi mutati ottenuti non sono frutto di un'estrazione casuale), sia dal punto di vista biologico (i.e. constatare che le proteine presenti nei nuovi sotto-grafi mutati estratti siano almeno in parte note nel contesto medico). Il lavoro di questa tesi si concen-

trerà soprattutto su quest'ultimo caso, validando i dati affiancandoli ad un elenco di proteine di cui è già nota l'importanza per lo sviluppo tumorale. Riassumendo, l'obiettivo della tesi è di estrarre nuovi gruppi di mutazioni genetiche, a partire da un dataset statistico e sfruttando l'*higher-order graph clustering* per individuare nuovi *subnetworks* mutati nei tumori e validando i risultati tramite confronto con un elenco di proteine mutate nei tumori già note nell'ambiente accademico.

1.4 Sinossi

In questa sezione viene descritta la struttura del resto della tesi.
Nella Sezione 2 viene effettuata una carrellata delle tecniche stato dell'arte nel contesto della *cancer genomics* e che costituiscono il punto di partenza della tesi.
Nella Sezione 3 viene analizzato il lavoro svolto durante l'elaborazione della tesi, osservando gli algoritmi implementati e specificando meglio gli obiettivi della tesi.
La Sezione 4 è la sezione che descrive i dataset presi in input, i parametri scelti per l'esecuzione degli algoritmi e i risultati ottenuti dalle sperimentazioni.
La Sezione 5 racchiude le considerazioni finali sul progetto, commentando i risultati ottenuti, a confronto tra loro; viene anche fornita una nota sui possibili sviluppi futuri del progetto.

Capitolo 2

Stato dell'Arte

In questa sezione, viene introdotto lo stato dell'arte delle tecniche che rispondono al problema presentato in sezione 1.3.
Nella Sezione 2.1 viene mostrato *HotNet 2*, un algoritmo che risponde direttamente al problema, pubblicato su *Nature* nel 2015 [4].
Nella Sezione 2.2, vengono mostrate tecniche note per il *pattern recognition* di motifs all'interno di grafi, per diversi motif considerati nel progetto.
Nella Sezione 2.3 viene mostrato l'algoritmo di Page Rank (general purpose) [3] che ha ispirato le tecniche presentate al capitolo 3.
Infine, nella Sezione 2.4, vengono mostrate le tecniche per valutare i risultati prodotti dagli algoritmi considerati.

2.1 HotNet2

HotNet2 [4] è l'algoritmo che risponde direttamente al problema presentato in Sezione 1.3; in particolare, si vuole identificare quei sotto-grafi di interazione proteina-proteina che sono mutati più di quanto ci si può aspettare per caso (i.e. i sotto-grafi identificati sono un risultato statisticamente significa-

tivo).

Questo task risulta particolarmente complicato in quanto in tali reti vi sono un numero di possibili sotto-grafi di taglia significativa (e.g. ≥ 8 geni) molto grande (e.g. $> 10^{14}$). Inoltre, questi sono particolarmente "densi", poiché molte geni (e specialmente quelle legate al cancro) hanno un un numero di vicini molto alto. Infine, come anticipato nella Sezione 1.3, il compito più complicato è legare lo *score* ottenuto da campioni biologici alle informazioni ottenibili dalle *protein-protein interaction networks*. *HotNet2* cerca quindi di effettuare un clustering che includa le informazioni presenti negli score dei singoli geni, oltre che alle informazioni topologiche di un grafo di interazione proteina-proteina. In altre parole, *HotNet2* prende in input un vettore h, contenente lo *score* per ogni proteina, ottenuto a partire dai campioni biologici, e il grafo di interazione; il suo output è una lista di gruppi di proteine, legate tra loro nel grafo di interazione proteina-proteina.

A tal scopo *HotNet2* effettua un processo di "diffusione del calore" all'interno del *network*; il calore non è altro che il vettore h introdotto sopra: è lo *score* associato alle singole proteine, chiamato anche *heat* dei geni. Il processo di diffusione avviene attraverso degli *step*: in questi, ogni proteina trattiene un po' del suo *heat* e ne cede la rimanente parte ai suoi vicini. La quantità di calore ceduta è regolata dal parametro α. Gli *step* vengono ripetuti fino a quando non viene raggiunta una situazione di equilibrio, nella quale il calore non si diffonde più nel grafo. Chiaramente, la situazione finale dipende sia dalla topologia, che dal calore iniziale associato alle proteine. Il processo appena presentato non è altro che una *random walk con restart*, le cui peculiarità sono state descritte in Sezione 1.2.2.

In tale *random walk* la sorgente non è unica: tutti i vettori con *heat* non nullo partecipano alla *walk*; in questo senso, la matrice di diffusione in equa-

zione 1.11 è adatta a rappresentare il metodo. Tuttavia, la matrice $\mathbf{F}$, pur descrivendo il processo Markoviano correttamente, considerando i vari vertici del grafo come possibili sorgenti, non associa ad essi lo score dato in input (vettore h). Gli autori di *HotNet2* mostrano come questo può essere facilmente preso in considerazione definendo la matrice $\mathbf{D}_h$ come quella matrice diagonale che ha entrate pari alle componenti del vettore h e usando la matrice:

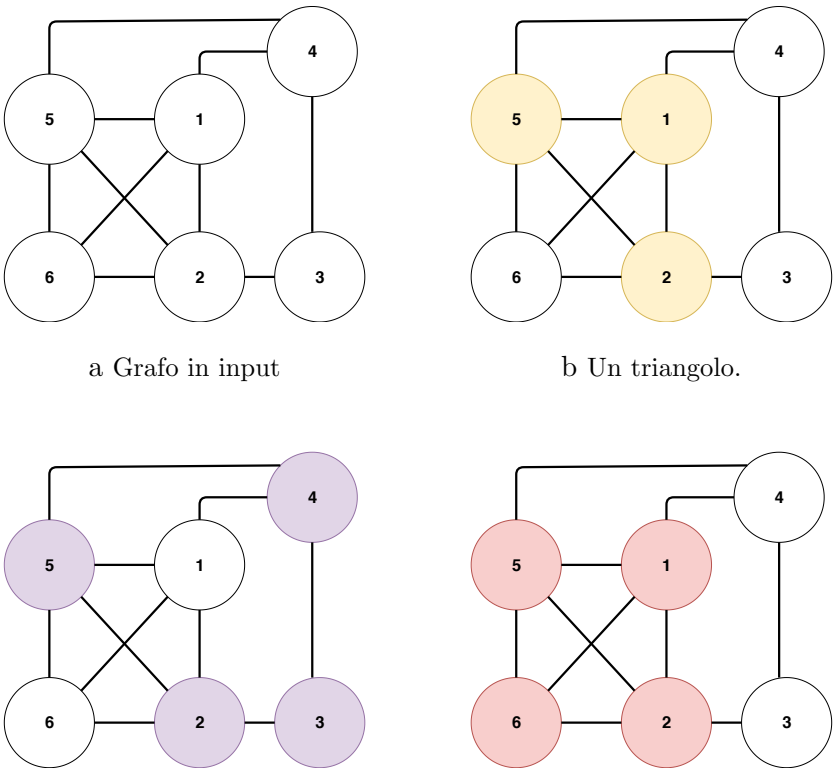
$$\mathbf{E} = \mathbf{F} \cdot \mathbf{D}_h \tag{2.1}$$

La matrice $\mathbf{E}$ rappresenta quindi la conclusione del processo di diffusione del calore; l'entrata (i, j)-esima della matrice rappresenta la quantità di calore diffuso dal nodo g_i al nodo g_j, se il calore $h(i)$ che viene posto inizialmente sul nodo g_i viene diffuso con una *random walk con restart*. Tale risultato può essere interpretato come una misura di similarità tra i nodi g_i e g_j [4].
Il processo di diffusione, da solo, non è sufficiente per effettuare un clustering sul grafo; la matrice $\mathbf{E}$ ottenuta è lo strumento con il quale il clustering viene effettuato, poiché per costruzione tiene conto dello score iniziale sui geni. A partire da essa, dato un parametro δ, viene definito un grafo *diretto* $H = (V, E_E)$, dove $E_E = \{(i, j) : e_{i,j} > \delta\}$. Il parametro δ rappresenta la soglia presso la quale una similarità tra i geni ottenuta viene considerata significativa. Il grafo H incarna tale restrizione: esso condivide i vertici del grafo di interazione proteina-proteina, $G = (V, E)$, e su essi v'è un *arco* se e solo se lo score di interazione ottenuto nella corrispondente entrata della matrice $\mathbf{E}$ (i.e. $e_{i,j}$) è più grande del parametro δ.
Per ottenere i *cluster*, *HotNet2* utilizza il grafo diretto H, individuandone le componenti *fortemente connesse*. Si ricordi che una componente fortemente connessa di un grafo diretto H è un sotto-grafo massimale di H in cui, data una qualsiasi coppia di nodi, deve esistere un cammino orientato tra essi. Le

componenti fortemente connesse ottenute sono quindi i cluster cercati.
Infine, *HotNet2* effettua un test statistico per determinare la significatività statistica dei risultati ottenuti; tali test non sono tuttavia oggetto della tesi, e per questo motivo non verranno approfonditi in questo contesto.

2.2 *Graph Motif Recognition*

Come anticipato nella Sezione 1.2.3, esistono diversi tipi di *graphlets* nei grafi; questa sezione vuole mostrare quali tecniche ci sono a disposizione per contare motifs all'interno di un grafo.



a Grafo in input

b Un triangolo.

c Un quadrilatero.

d Una clique.

Figura 2.1: Individuazione di motifs all'interno di un grafo

2.2.1 Triangoli

Dato un grafo $G = (V, E)$, un triangolo è un sotto-grafo completo (o clique) di taglia tre; in altre parole, è un gruppo di tre vertici appartenenti a G, tutti adiacenti tra loro. In Figura 2.1b, ad esempio, è possibile individuare un'istanza del motif triangolo nel sotto-grafo composto dai vertici $\{1, 2, 5\}$. Per individuare tutti i triangoli presenti in un grafo, viene di seguito introdotto l'algoritmo proposto da Chiba & Nishizeki nel 1985 [5]. L'algoritmo è basato sulla seguente intuizione: un triangolo su un grafo non è altro che un vicino in comune a due nodi. La strategia presentata dagli autori è cauta nell'enumerazione completa dei *motifs* triangolari: per individuare i vicini in comune, viene proposta una colorazione dei nodi adiacenti all'i-esimo esaminato, e allo stesso tempo, per evitare duplicati, viene eseguito un ordinamento per l'esplorazione dei nodi in base al loro grado nel grafo, con eliminazione del vettore v_i da G, una volta esaminato. Il vantaggio di questa scelta è di natura analitica, per limitare il tempo di esecuzione: eliminando nodi vertici di grado più alto, il grafo tende a diventare sempre più sparso ad ogni interazione; ciò previene anche di esaminare alcuni lati troppe volte. Come dimostrato dagli autori, l'Algoritmo 2.2.1 ha complessità lineare nel numero dei lati del grafo; l'algoritmo elenca tutti i triangoli presenti nel grafo, senza ripetizioni in $O(a(G)m)$ operazioni, dove $a(G)$ è detta *arboricità* del grafo G, mentre $m = |E|$. Per inciso, l'arboricità di un grafo non diretto è il minimo numero di *spanning trees* necessari per coprire tutti i lati del grafo. L'arboricità è, infatti, una misura della *densità* del grafo; presi due grafi non diretti, a parità di vertici, il grafo con più lati avrà un'arboricità più alta. Viceversa, presi due grafi non diretti, il grafo con un'arboricità più alta ha sicuramente un sotto-grafo "denso".

Algorithm 2.2.1 find_triangles

Require: $G = (V, E)$

Ensure: Stampa tutte le istanze triangolo in G

```
1: Ordina i vertici del grafo per il loro grado in modo non crescente
2: L'indice i rappresenta tale ordinamento
3: for i = 1 to n − 2 do
4:     Colora i vertici adiacenti a v_i
5:     for all vertice u colorato do
6:         for all vertice w adiacente a u do
7:             if w è colorato then
8:                 output (v_i,w,u) come triangolo
9:             end if
10:            de-colora il vertice u
11:        end for
12:    end for
13:    elimina v_i da G
14: end for
```

2.2.2 Quadrilateri

Dato un grafo $G = (V, E)$, un quadrato è un sotto-grafo composto da quattro vertici, a due a due adiacenti tra loro; in altre parole, un vertice $u \in G$ partecipa ad un quadrato se il vicino del suo vicino è anche adiacente a u stesso. In Figura 2.1c, ad esempio, è possibile individuare un'istanza del motif quadrato nel sotto-grafo composto dai vertici $\{5, 4, 3, 2\}$. Al fine di individuare tutti i quadrati presenti in un grafo, Chiba & Nishizeki hanno presentato un algoritmo basato su considerazioni simili a quello mostrato in Sezione 2.2.1 [5].

L'intuizione su cui si basa l'Algoritmo 2.2.2 è la seguente. Se, presi due vertici v e w, si definisce l'insieme U che contiene tutti quei vertici $u_1, ..., u_L$ (con $L \geq 2$) adiacenti sia a v che a w, allora questi $L + 2$ vertici inducono un grafo bipartito completo. Di conseguenza, qualsiasi quadrupla $\{v, w, u_i, u_j\}$, con $1 \leq i < j \leq L$, è un quadrato per definizione. L'algoritmo si concentra quindi nell'individuare grafi bipartiti, dove è sufficiente ritornare in output le triple del tipo $\{v, w, U\}$, che rappresentano tutti quei quadrati ottenibili come combinazione di v, w e una qualsiasi coppia $u_i, u_j \in U$. Si noti che in un quadrato v e w sono a distanza due tra di loro. Ad esempio, prendendo come riferimento $v = 2$ e $w = 4$ in Figura 2.1, l'insieme U è costituito dai vertici $\{1, 3\}$: è quindi sufficiente ritornare in output la tripla $\{2, 4, \{1, 3\}\}$ per elencare i quadrati composti dai vertici $\{1, 2, 3, 4)\}$ e $\{1, 2, 4, 5\}$.

L'algoritmo proposto itera su tutti i vertici $v \in V$, cercando tutti i quadrati che contengono v; per ogni vertice w a distanza due da v, l'algoritmo cerca tutti i vertici $u_1, ..., u_L$ adiacenti ad entrambi i nodi, salvandoli nell'insieme U. Se $|U| > 2$, allora è stato trovato almeno un quadrato e lo si ritorna in output; infine, una volta trovati i quadrati che contengono v, l'algoritmo lo elimina dal grafo per evitare duplicati.

Nel loro articolo, Chiba e Nishizeki mostrano che la complessità dell'Algoritmo 2.2.2 è la stessa dell'Algoritmo 2.2.1.

Algorithm 2.2.2 find_quadrangles

Require: $G = (V, E)$

Ensure: Stampa tutte le istanze quadrato in G

1: Ordina i vertici del grafo per il loro grado in modo non crescente
2: L'indice i rappresenta tale ordinamento
3: **for** $i = 1$ to n **do**
4: $U[w] = \emptyset$, $\forall w \in V$
5: **for all** vertice u adiacente a v_i **do**
6: **for all** vertice $w \neq v_i$ adiacente a u **do**
7: $U[w] = U[w] \cup \{u\}$
8: **end for**
9: **end for**
10: **for all** vertice w tale che $|U[w]| \geq 2$ **do**
11: **output** la tripla (v_i,w,$U[w]$)
12: **end for**
13: elimina v_i da G
14: **end for**

2.2.3 *Cliques*

Dato un grafo $G = (V, E)$, una *clique* di taglia k è un sotto-grafo completo di k vertici; le *cliques* sono quindi gruppi di k nodi tutti adiacenti tra loro. In Figura 2.1d si può osservare solo una clique di taglia $k = 4$, ovvero il sotto-grafo composto dai vertici $\{1, 5, 6, 2\}$. Di nuovo, Chiba & Nishizeki presentano un'intuizione che permette di individuare tutte le cliques in un grafo [5].

Per risolvere il problema, gli autori riprendono le considerazioni effettuate per risolvere il problema per triangoli, presentate qui in Sezione 2.2.1; innan-

zitutto, si osserva come una clique di taglia $k = 2$ non è altro che una coppia di nodi adiacenti. Per individuare un triangolo, l'Algoritmo 2.2.1 cerca coppie di nodi già adiacenti tra loro e verifica che entrambi gli elementi della coppia siano adiacenti ad un terzo nodo; a tutti gli effetti, re-interpretando, un triangolo è una clique di taglia $k = 3$: per individuarla, l'algoritmo prima individua una clique di taglia $k = 2$, poi s'assicura che tutti gli elementi presenti nella clique di taglia $k = 2$ fossero adiacenti ad un terzo nodo.
Gli autori propongono quindi un algoritmo ricorsivo, che può individuare una clique di taglia $k = L$ contenente un certo vertice v, individuando una clique di taglia $k = L - 1$ indotta dai vicini di v. Come nei precedenti algoritmi, gli autori propongono un ordinamento sui vertici in base al loro grado prima di iniziare e, di nuovo, viene eliminato il nodo corrente elaborato v_i dal grafo per evitare inutili ripetizioni.

L'Algoritmo 2.2.3 individua correttamente tutte le cliques di taglia L in un grafo G, effettuando la prima chiamata $find\_k - cliques(G_k = G, k = L)$. Come dichiarato dagli stessi autori, l'algoritmo è stato così presentato per far passare l'intuizione della soluzione al problema e per l'analisi di correttezza, ma non per agevolarne una diretta implementazione; l'Algoritmo 2.2.3 presuppone un salvataggio in memoria del grafo corrente G_k, per ogni ricorsione: ciò è chiaramente *unfeasible* per memoria. Chiba & Nishizeki suggeriscono una diretta implementazione che etichetta i vettori appartenenti al grafo G_k con k, in modo tale da distinguerli da altri grafi indotti ottenuti in ricorsioni successive.
Danisch et. al., in un articolo del 2018 [6], propongono un algoritmo che ha intenzione di migliorarne le performance dell'algoritmo proposto da Chiba & Nishizeki; a tal proposito, gli autori effettuano tre osservazioni:

1. L'algoritmo non è parallelizzabile, perché presuppone di eliminare se-

Algorithm 2.2.3 find_k-cliques

Require: $G_k = (V_k, E_k)$, k

Ensure: Stampa tutte le istanze *k-cliques* in G

```
1: C è una stack globale
2: if k = 2 then
3:     for all e = (u, v) ∈ E_k do
4:         output {u, v} ∪ C
5:     end for
6: else
7:     Ordina i vertici del grafo per il loro grado in modo non crescente
8:     L'indice i rappresenta tale ordinamento
9:     for i = 1 to n do
10:        Ottieni G_{k-1} come sotto-grafo indotto dai vicini di v_i in G_k
11:        C.push(v_i)
12:        find_k-cliques(G_{k-1}, k − 1)
13:        C.pop()
14:        V_k = V_k \ v_i
15:    end for
16: end if
```

quenzialmente un nodo dal grafo indotto corrente (linea 14 dell'Algoritmo 2.2.3);

2. L'ordinamento in base al grado dei nodi è un buon modo per limitare il tempo d'esecuzione dell'Algoritmo 2.2.3 in fase di analisi, ma difficilmente è, nella pratica, l'ordinamento migliore;

3. Un'ulteriore intuizione sull'ordine di esplorazione dei lati e dei nodi del grafo permette agli autori di attuare un miglioramento sul tempo d'esecuzione, scegliendo di utilizzare un grafo *diretto*, ricavato da G.

L'Algoritmo 2.2.4 è la proposta degli autori che risponde ai problemi presentati. L'ordinamento η è un qualsiasi ordinamento sui nodi del grafo G: gli autori mostrano che se l'ordinamento viene effettuato per *degenerazione* si ottiene uno speed-up notevole. In breve, la degenerazione (o *core*) di un grafo G è il minimo grado tra tutti i nodi all'interno di un qualsiasi sotto-grafo indotto di G: la degenerazione è una misura diametralmente opposta all'arboricità, in quanto è una misura di quanto è "sparso" il grafo.
Gli autori propongono, per implementare lo speed-up e per semplificarne l'implementazione, due strutture dati.
La prima proposta è un grafo diretto aciclico (anche detto DAG), ottenuto dal grafo non diretto preso in input; tale grafo, $G^d = (V, E^d)$ condivide i vertici con il grafo non diretto G, mentre v'è un arco (u, v) in E^d se e solo se esiste il lato $\{u, v\}$ in G e $\eta(u) > \eta(v)$. In questo modo si ottiene un DAG che rappresenta l'ordinamento (e.g. la degenerazione su G) e che implementa effettivamente uno speed-up sul numero di lati percorsi: su questo grafo si ha la garanzia che un lato viene percorso una volta sola (in quanto diretto e aciclico), anche scegliendo un qualsiasi sotto-grafo indotto da un qualsiasi vertice v incontrato in una qualsiasi ricorsione.

La seconda proposta degli autori è l'implementazione efficiente dei DAG; l'obiettivo è quello di individuare efficientemente il grafo indotto dai vicini di un vertice v. Detti $\delta^+(v)$ l'*out-degree* del vertice v e $\Delta^+(v)$ la lista contenente tutti suoi i vicini (uscenti), $G^d[\Delta_{G^d}(v)]$ è il grafo indotto dai vicini di v. Di conseguenza, il grafo diretto viene rappresentato tramite una lista di adiacenza, la quale per ogni vertice v memorizza il suo *out-degree* $\delta^+(v)$ e tutti i suoi vicini nel vettore $\Delta(v)$.
Per evitare la creazione non controllata di liste di adiacenza (per ogni grafo indotto da un vertice v), ci si avvale di un vettore associativo ausiliario U, le cui componenti vengono tutte inizializzate a k, un etichetta che rappresenta la taglia k della clique da trovare nella ricorsione corrente. Quando, in una generica ricorsione a livello k, viene richiesto di creare $G^d[\Delta_{G^d}(v)]$, invece che memorizzare un'altra lista di adiacenza, nel vettore U le etichette dei vicini $\Delta_{G^d}(v)$ di v vengono portate a $k-1$. In questo modo è facile distinguere tra nodi appartenenti ai sotto-grafi indotti dai vertici $v \in V(G^d)$.
Come anticipato, queste scelte implementative portano ad un'ultima osservazione: i diversi sotto-grafi $G^d[\Delta_{G^d}(v)]$ possono essere elaborati indipendentemente, dato che le cliques trovate in un sotto-grafo non verrebbero ritornate più di una volta, poiché G^d è un grafo diretto aciclico; di conseguenza, anche il vettore U può essere gestito in parallelo.

2.3 *Motif-Based Approximate Personalized Page Rank*

L'algoritmo di *Page Rank* è alla base del motore di ricerca *Google*, che ha lo scopo di valutare il posizionamento delle pagine web in corrispondenza di una qualsiasi *query*; attualmente, l'algoritmo è parte di un sistema in

Algorithm 2.2.4 Danisch_find_k-cliques

Require: $G^d = (V, E^d)$, k, η, C

Ensure: Stampa tutte le istanze *k-cliques* in G

```
1: if k = 2 then
2:     for all e = (u, v) ∈ E^d do
3:         output {u, v} ∪ C
4:     end for
5: else
6:     for all v ∈ V(G^d) do
7:         Danisch_find_k-cliques(G^d[Δ_{G^d}(v)],k − 1, C ∪ {u})
8:     end for
9: end if
```

Google più avanzato. Il contesto di *Page Rank* è lontano da quello della Bioinformatica. Tuttavia, in questa sezione si mostra come una variante di *Page Rank*, *Personalized Page Rank* (PPR), è a tutti gli effetti una *random walk con restart*. Non solo: la tesi si ispira ad risultato ottenuto nel 2017 da Yin et. al. [3], che ha lo scopo di aumentare la portata dei risultati ottenibili con un PPR. Il metodo, come mostrato in questa sezione, è una modificazione del grafo di partenza sul quale effettuare la random walk che tiene conto dei motifs nel grafo.

2.3.1 Page Rank

Page Rank è un algoritmo che assegna uno *score* ad ogni elemento di un certo numero di documenti (solitamente, connessi nel *web*); il significato di tale score è quello di catturare l'*importanza* delle pagine valutate, rispetto al resto dei contenuti presenti negli altri documenti. Nel contesto web, la grande sfida è l'efficienza, data la grande mole di dati.

A tal scopo, il grafo è la rappresentazione più naturale di documenti (o in

generale contenuti) interconnessi tra loro in *Internet*; un algoritmo di *Page Rank* assegna quindi uno *score* ad ogni vertice del grafo ottenuto: tale valore numerico rappresenta l'*importanza* di un nodo, relativamente all'intero grafo (esempio in Figura 2.2). Con importanza di un contenuto s'intende la probabilità che una persona, navigando su internet "a caso", arrivi a tale documento; per l'utente, navigare Internet casualmente significa seguire gli ipertesti presenti sulla pagina (quella visualizzata attualmente), oppure significa saltare direttamente ad un'altra pagina (digitando un indirizzo nell'apposita barra). Per questo motivo, lo score di una misura di *Page Rank* è

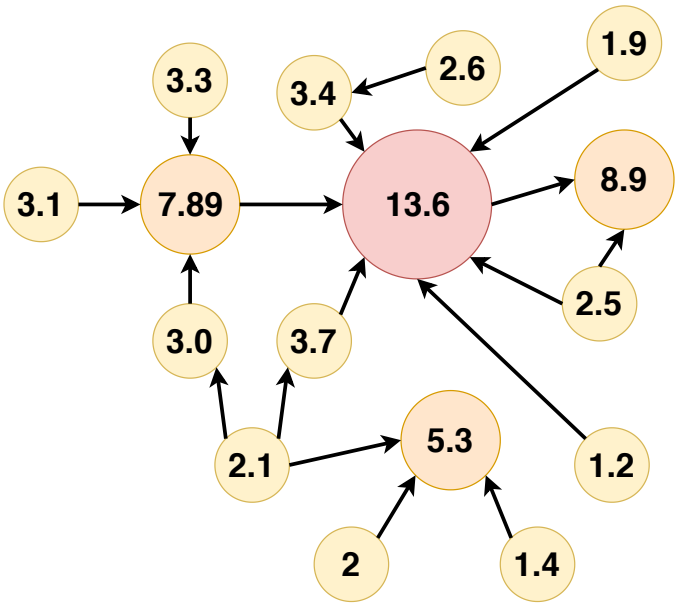


Figura 2.2: Esempio di *score* assegnati ai nodi di un grafo, ad esempio il *Web*.

dato proprio da una misura ottenibile da una *random walk con restart* (leggermente diversa da quanto definito fin'ora): con una certa probabilità α, l'utente segue un *link*, passando quindi da un nodo del grafo ad un altro; viceversa, con probabilità $1 - \alpha$, l'utente salta ad un'altra, scegliendo quindi un qualsiasi altro nodo del grafo, casualmente.

Personalized Page Rank (PPR) è invece una metrica simile al *Page Rank*, dove la misura di centralità viene misurata dalla prospettiva di un insieme di specifici nodi inizialmente definito (da qui, *personalized*), ad esempio chiamato Q; l'importanza dei nodi nel grafo non viene più misurata prendendo *tutti* i nodi come valide alternative al *restart*: piuttosto, esso può avvenire solo tra nodi appartenenti all'insieme Q. Questa peculiare scelta è giustificata dall'utilizzo reale di *Internet*, nel quale difficilmente un utente effettua un salto ad un contenuto *davvero* casuale, quando piuttosto, se esegue dei salti, l'utente sceglie di effettuarli prendendo come riferimento dei nodi "ben noti" (e.g. homepage).

Da un punto di vista più matematico, se l'insieme Q contiene un solo nodo i, allora l'i-esima colonna della matrice in equazione 1.11 è esattamente lo score da assegnare ai nodi del grafo. In questo senso *PPR* e la prima fase di *HotNet2* sono identiche: entrambi gli algoritmi vogliono ottenere una certa la matrice di transizione $\mathbf{F}$; per far ciò, come evidenziato dall'equazione 1.11, è necessario svolgere alcune operazioni matriciali: inversioni e moltiplicazioni righe per colonna. Tali operazioni diventano particolarmente onerose per moli di dati elevate, come avviene ad esempio in Internet; ancor più importante è la dinamicità della rete del *web*, che rende necessario effettuare spesse volte il calcolo degli *score* dei contenuti. Per questa ragione esistono diversi algoritmi di approssimazione; tra i più citati esiste "Approximate Personalized Page Rank" (APPR), proposto da Andersen et. al. [7], il cui tempo di esecuzione dipende fortemente dalla qualità dell'approssimazione desiderata. Questa problematica non è tuttavia presente nel contesto di questa tesi, ovvero nei grafi di interazione proteina-proteina, i quali pur essendo di medie dimensioni, richiedono una singola computazione (a parità di input e parametri).

2.3.2 *Motif-Based Page Rank*

Yin et. al. [3], cercano un metodo per catturare nei vettori di *Page Rank* informazioni topologiche ad "alto livello" nei grafi, anche detti più semplicemente *motifs* o *graphlets*, presentati in Sezione 1.2.3.

L'idea è la seguente: in una prima fase, dato un motif, l'algoritmo ha il compito di contare tutte le sue istanze nel grafo; il conteggio va eseguito contando il numero di volte che i singoli *lati* del grafo partecipano al motif. In altre parole, si vuole ottenere un grafo pesato $G_w = (V, E_w)$, dove e_{ij} è il numero di volte in cui il lato $\{i, j\}$ partecipa al motif dato. In tale grafo è possibile interpretare i pesi tra due vertici i e j come tanti lati paralleli quante sono le volte che il lato $\{i, j\}$ partecipa ad una struttura più ad alto livello (i.e. il motif scelto) nel grafo originale G. Volendo rappresentare i grafi con matrici di adiacenza, in questa prima fase si ottiene quindi la matrice $\mathbf{W}$, che rappresenta i pesi tra i vertici nel grafo G_w; tale matrice risulta simmetrica in quanto si assume di avere grafi non diretti in input. Formalmente, presi due nodi $i, j \in V$, sia S l'insieme che contiene tutti i motif ai quali partecipano sia i che j; allora, l(i, j)-esima entrata della matrice $\mathbf{W}$ è:

$$w_{ij} = \begin{cases} |S|, \text{ se il nodo i è adiacente a j} \\ 0 \text{ altrimenti} \end{cases} \tag{2.2}$$

Al fine di calcolare il PPR, gli autori [3] eseguono la random walk con restart da un unico nodo sorgente u a partire dalla matrice di transizione G_w, che definisce un processo Markoviano; gli autori chiamano l'algoritmo *Motif-based Approximate Personalized Page Rank* (MAPPR).

Riassumendo, MAPPR non è altro che un algoritmo che esegue una random walk con restart a partire da un nodo sorgente u, che approssima il calcolo della distribuzione stazionaria con un algoritmo proposto da Andersen e.t al.

[7], a partire da una matrice di transizione che dipende dal numero di istanze di un certo motif scelto sui lati del grafo preso in input.

2.4 Valutazione

In un qualsiasi esperimento scientifico, è buona norma validare i propri risultati con dei metodi che permettono di effettuarne una corretta valutazione. Anche nel contesto della cancer-genomics, vi sono diversi metodi; ad esempio, come precedentemente menzionato in Sezione 2.1, i test statistici sono all'ordine del giorno.

Questa tesi, invece, prende ispirazione dai metodi utilizzati in [8], che effettuano la valutazione della bontà dei risultati ottenuti tramite confronto. Nel contesto della tesi, i risultati sono i cluster di proteine mutate nei tumori: il confronto avviene prendendo tali cluster e confrontandoli con un elenco di proteine già note per essere *driver* nel cancro.

Più specificatamente, viene definito l'insieme P che contiene tutte le proteine già note alla comunità scientifica per il cancro e, conseguentemente, l'insieme N, complementare, che contiene tutte le altre.

Detto TP l'insieme dei True Positive, ovvero l'insieme che contiene le proteine correttamente identificate come mutate dall'algoritmo, una prima misura di correttezza dell'algoritmo avviene attraverso il True Positive Rate:

$$TPR = \frac{|TP|}{|P|}.$$

TPR misura il rapporto tra il numero di proteine correttamente identificate dall'algoritmo come cancerose (TP) e il numero di tutte le proteine note alla comunità (P). Nel contesto della tesi il TPR è una misura che permette di quantificare la bontà di un algoritmo, o quantomeno di presentare dei risultati in linea con quanto già ottenuto fin'ora dalla comunità scientifica. TPR,

ovviamente, può variare tra 0 (i.e. $TP = \emptyset$) e 1 (i.e. $TP \equiv P$).
Viceversa, si vuole valutare anche con che frequenza l'algoritmo commette errori, che solitamente sono di due tipi; generalmente, si distingue tra errori di tipo uno, ovvero quando si ottengono falsi positivi, e di errori di tipo due, ovvero quando si ottengono falsi negativi. Nel contesto della tesi ci si riferisce solamente ad errori di tipo uno. A tal scopo, si definisce FP come l'insieme dei False Positive, ovvero quell'insieme che contiene le proteine identificate come mutate dall'algoritmo, ma che in realtà non vengono riconosciute come tali dalla comunità scientifica (i.e. tali proteine appartengono in realtà all'insieme N). Di conseguenza, il False Positive Rate

$$FPR = \frac{|FP|}{|N|}$$

è la misura per errori di tipo uno, perché esso non è altro che il rapporto tra il numero di proteine erroneamente identificate dall'algoritmo come cancerose (FP), quando in realtà non lo sarebbero, e il numero di tutte le proteine note alla comunità come non cancerose (N). Nel contesto della tesi il FPR misura non solo lo scostamento con quanto già scoperto dalla comunità scientifica, ma anche della capacità dell'algoritmo di scoprire nuovi *pathways* fin'ora non noti; chiaramente, la validità di tali scoperte va confermata da propri test statistici. Anche FPR può variare tra 0 ($FP = \emptyset$) e 1 ($FP \equiv N$).
Va infine specificato che TPR e FPR sono coefficienti dipendono da un parametro (o più parametri), ad esempio chiamato k. Tale parametro, nel contesto della tesi, regola il numero di cluster da valutare. Ad esempio, se un algoritmo fornisce in output sette cluster, il parametro k rappresenta la minima taglia dei cluster che devono essere valutati: se ad esempio $k = 4$, allora solo i cluster con taglia ≥ 4 vengono inclusi nell'analisi per il calcolo di FPR e TPR. L'obiettivo è quello di osservare le valutazioni sui cluster al variare del parametro k.

Al fine di visualizzare tali variazioni, come viene effettuato anche in [8], si effettua un *plot* di TPR e FPR; tale grafico evidenzia l'andamento dei due coefficienti, al variare del parametro k: le curve ottenute vengono chiamate "Receiver Operating Characteristic" ROC e sono ampiamente utilizzati in letteratura [8] per visualizzare il rapporto che c'è tra *hit rate* e *falsi allarmi* di qualche tipo.

Ogni punto di una curva ROC rappresenta i coefficienti TPR e FPR: fissato un certo k, si otterrà un certo TPR e un certo FPR, che possono essere rappresentati con un punto, la cui ordinata corrispondonde al valore di TPR e ascissa al valore di FPR. È chiaro che, al variare di k, la curva sarà monotonicamente crescente: nel contesto della tesi, più geni si includono nell'analisi, più può aumentare il TPR (o al più rimanere uguale, se si aggiungono solo falsi positivi); viceversa, anche il FPR può solo che aumentare (o al più rimanere uguale, se si aggiungono solo *true positives*).

In una curva ROC, quindi, la posizione dei punti nel grafico rappresenta il rapporto che c'è tra TPR e FPR e, soprattutto qual è la parametrizzazione ideale per il sistema progettato: le curve permettono di effettuare un confronto qualitativo tra i metodi implementati. Quando si effettuano questo tipo di analisi, solitamente, a parità di TPR si preferirà la parametrizzazione che permette di ricavare un FPR minore. Al fine di effettuare conclusioni su quale dei metodi implementati sia il "migliore" (i.e. sbaglia meno spesso e, contemporaneamente, ha un *hit rate* maggiore), si effettua il calcolo della "Area Under the Curve" (AUC). L'area sottesa alle curve ROC rappresenta direttamente la qualità del metodo presentato, poiché se il metodo presentato è di alta qualità, allora avrà pochi falsi positivi e avrà un buon *hit-rate*; di conseguenza, la curva ROC s'innalzerà verso le ordinate più velocemente, e verso le ascisse meno velocemente: per questa ragione la AUC sarà più alta

in corrispondenza di metodi migliori dal punti di vista dell'accuratezza.
La parametrizzazione, nel contesto della tesi, viene effettuata come segue. Siano s_{max} e s_{min} rispettivamente la massima e la minima taglia dei cluster ottenuti: allora, al fine di costruire la curva *ROC*, è possibile far variare il parametro $k$da $s_{max}+1$ a s_{min}. Quando $k = s_{max}+1$, nessun cluster viene preso in considerazione, e quindi (inevitabilmente) $TPR(k) = 0$ e $FPR(k) = 0$, ottenendo il primo punto del *plot*, $(0, 0)$. Al variare di k, quindi, si ottengono diverse valutazioni che corrispondono a determinati punti del grafico: quando $k = s_{min}$, vengono presi in considerazione tutti i cluster ritornati dall'algoritmo e si raggiunge l'ultimo punto del *plot*, presso il quale TPR e FPR sono massimi. In Figura 2.3 è possibile osservare diversi esempi di *ROC curves*.

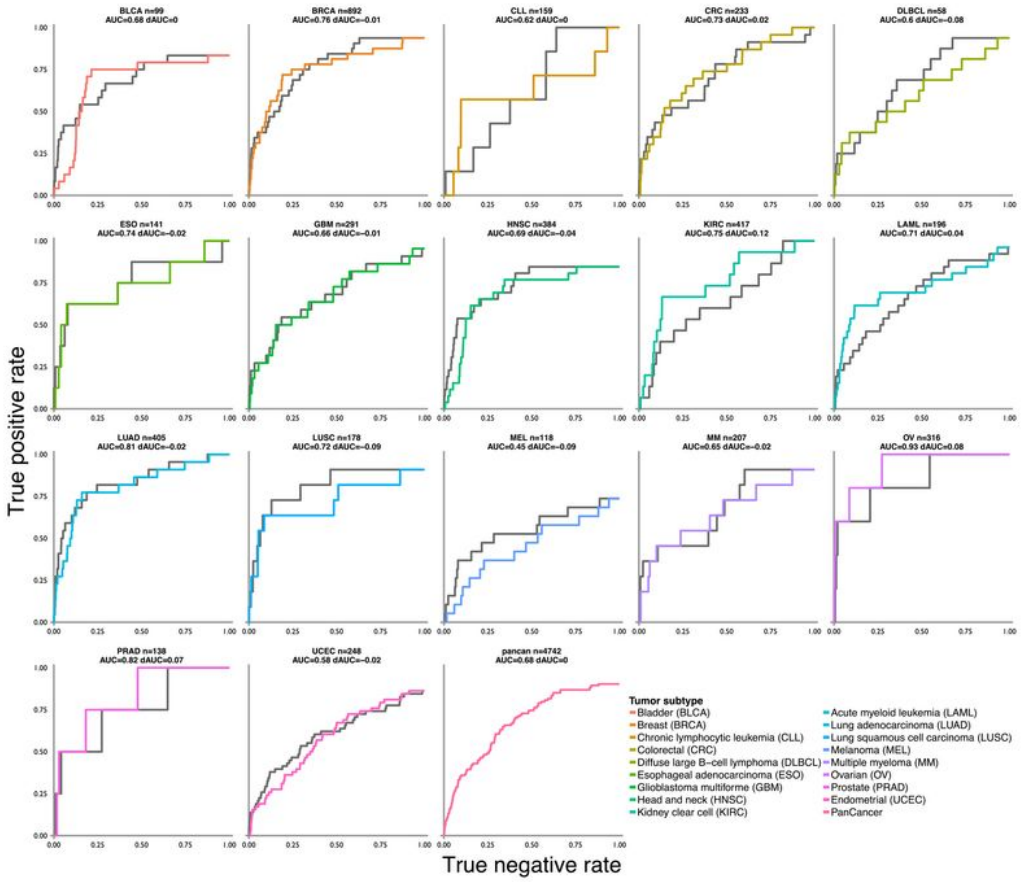


Figura 2.3: In [8], gli autori valutano una tecnica, *NetSig*, per diversi tipi di tumori (e.g. glioblastoma multiforme, adenocarcinoma seno, prostata, etc.)

Capitolo 3

Metodi

Nella sezione corrente, si vuole mostrare il lavoro svolto nel corso della tesi: si parlerà infatti dagli obiettivi che ha l'elaborato (Sezione 3.1) e si descriverà l'algoritmo implementato (Sezione 3.2) per risolvere i problemi presentati.

3.1 Obiettivi

Il lavoro della tesi è incentrata sull'estrazione di sotto-grafi mutati, con tecniche presentate precedentemente, a partire da due elementi chiave: uno *score*, detto *heat*, ricavato da un *dataset* reale e composto da campioni biologici (i.e. mutazioni), e un grafo di interazione proteina-proteina. Quanto descritto fin'ora è esattamente il lavoro mostrato dagli autori di *HotNet2*: in questa tesi si vogliono ricavare i risultati ottenibili con l'algoritmo se le random walk con restart vengono effettuate sul grafo pesato $G_w = (V, E_w)$, ricavato dal conteggio dei motifs sul grafo di interazione proteina-proteina, come presentato in Sezione 2.3.2.
Un obiettivo fondamentale della tesi è stato la corretta implementazione del conteggio dei motifs sui grafi: tutti gli algoritmi presentati in Sezione 2.2 mo-

strano come stampare in output i motif, quando in realtà in [3] viene richiesto di contare il numero di lati del grafo che partecipano ad istanze di tali motif, al fine di ricavare $\mathbf{W}$, come approfondito in Sezione 2.3.2. Nella Sezione 3.2 vengono mostrate le modifiche implementate per risolvere tale problema. Inoltre, nel corso del lavoro alla tesi, si è scelto di implementare una soluzione ad una versione alternativa del problema del conteggio, al quale d'ora in poi ci si riferirà con "*soft-version*"; nel ricavare $\mathbf{W}$, e quindi G_w, piuttosto che disconnettere completamente tutti quei lati che non partecipano ad alcun motif, si sceglie di mantenere i lati nel grafo originale, aggiungendo in G_w un lato parallelo ad ogni lato in G che partecipa al motif scelto. In questo modo le relazioni originali del grafo di interazione proteina-proteina vengono mantenute, e vengono "migliorate", tutte quelle relazioni che partecipano ad una struttura ad alto livello (i.e. ad un motif), senza cancellare i lati che non vi partecipano.

Un altro problema indirizzato dalla tesi è il seguente: $\mathbf{W}$, pur rappresentando il nuovo grafo $G_w = (V, E_w)$, non rappresenta (direttamente) una random walk su di esso. Per questo motivo, l'obiettivo è quello di ricavare correttamente la matrice di transizione $\mathbf{W}^t$ corrispondente.

Interpretando i pesi sui lati del grafo G_w come archi paralleli tra vertici in V, si può affermare che il grado di un nodo v dipende dal numero di volte in cui i suoi lati adiacenti partecipano ad un dato motif; per questa ragione, in una random walk con restart su tale grafo, se si prende come sorgente v, la probabilità di spostarsi lungo un arco adiacente a v che partecipa ad un elevato numero di istanze del motif è più alta. Data la matrice $\mathbf{W}$, rappresentante il conteggio dei motifs per lati, e assumendo che essa rappresenti un grafo connesso, è facile ricavare la matrice di transizione $\mathbf{W}^t$, i cui elementi

sono:

$$w_{ij}^t = \frac{w_{ij}}{\sum_{j \in V} w_{ij}} \tag{3.1}$$

Si noti come, confrontando con la matrice di transizione proposta nella Sezione 1.2.2 nell'equazione 1.6, la matrice $\mathbf{W}^t$ viene costruita in modo analogo: in grafi non pesati il numeratore in equazione 3.1 è sempre pari a 1, mentre il denominatore coincide sempre con il grado del nodo. Anche per quanto riguarda la matrice $\mathbf{W}^t$, per costruzione, si viene a perdere la simmetria che si trovava invece in $\mathbf{W}$, la matrice di adiacenza che rappresenta il grafo pesato G_w.

Una volta implementate queste modifiche, viene eseguito il processo di diffusione del calore e il conseguente clustering sulle componenti fortemente connesse in H, come mostrato in Sezione 2.1.

L'ultimo passo della tesi è la valutazione di tali sotto-grafi mutati tramite i coefficienti TPR e FPR; l'obiettivo finale è quello di confrontare i diversi risultati ottenibili, cambiando il motif da conteggiare (e.g. triangoli, quadrati, cliques, etc.) e il parametro δ (si veda la Sezione 2.1). Si vuole infatti indagare come i motif scelti influenzano i sotto-grafi mutati ottenibili con la metodologia di HotNet2: data una coppia motif-δ, si indaga su quanti di questi questi contengono geni *well-known* per essere *cancer-genes*, e quali invece sono noti alla comunità scientifica come tali.

In poche parole, l'obiettivo della tesi è di osservare il cambiamento dei sotto-grafi mutati ritornati da un algoritmo che tiene in considerazione l'analisi dei motifs nel grafo: al fine di effettuare tali confronti, ci si affida ai coefficienti TPR e FPR, che rappresenteranno, in questo lavoro, la principale metrica di qualità dei cluster.

3.2 Algoritmo

Il breve, l'algoritmo scritto segue i seguenti passi:

1. Fase di pre-processing e *parsing* sui dati in input, al fine di semplificare le restanti parti dell'algoritmo, completo di acquisizione degli input e dei parametri in opportune strutture dati;

2. Fase di conteggio motif e conseguente ricavo della matrice $\mathbf{W}$, che come spiegato in Sezione 2.3.2, rappresenta un grafo pesato (o con archi paralleli tra i vertici) ricavato a partire dal motif M e dal grafo in input G;

3. Fase di ricavo della matrice di transizione $\mathbf{W}^t$, che rappresenta la random walk con restart sul grafo pesato ottenuto;

4. Fase di diffusione del calore sul grafo, che individua la distribuzione stazionaria a partire dal parametro α (dove $1 - \alpha$ è la probabilità di restart) e da $\mathbf{W}^t$;

5. Fase di clustering, nel quale si ottengono i sotto-grafi mutati cercati, i.e. le componenti fortemente connesse dal grafo H, come approfondito nella Sezione 2.1;

6. Fase di valutazione, nella quale si calcolano FPR e TPR per l'esperimento eseguito.

Nelle successive sezioni, tali fasi vengono descritte in modo più dettagliato.

3.2.1 Input e pre-processing

Al fine di standardizzare l'input fornito ad HotNet2, ovvero il calore h e il grafo di interazione proteina-proteina G, nel corso dello svolgimento del lavo-

ro della tesi si è scelto di ri-elaborare il *dataset* fornito; l'obiettivo è quello di rendere indistinguibile all'algoritmo un grafo di interazione proteina-proteina da tutti quelli ricavabili dal conteggio dei motifs.
Innanzitutto, dato il dataset rappresentante il grafo, sono stati etichettati tutti i vertici con il loro nome; già in questa fase comparivano etichette che non avevano una corrispondenza nel grafo, o elementi del grafo isolati; per questo motivi questi elementi sono stati filtrati.
In più, è stato applicato un ulteriore filtro per rimuovere tutti quei vertici la cui etichetta, pur avendo uno score in h non nullo, non compare nei *file* del grafo di interazione proteina-proteina. Nell'applicare queste modifiche al dataset, è stato effettuato un re-mapping dei vertici, in modo da tenere ordinata l'indicizzazione dei vertici e la loro etichetta.
Infine, il dataset così ricavato è stato salvato a parte, senza sovrascrivere i dati originali; d'ora in poi, gli algoritmi implementati lavorano su tali dati processati.
Tutti i parametri necessari per far funzionare il programma sono stati acquisiti via linea di comando, e gestiti da un opportuno parsing; tra questi ci sono: motif scelto (e.g. triangolo, quadrato, clique, etc.), α, δ, k (quando viene scelto clique come motif), il metodo per ricavare la distribuzione stazionaria (inversione di matrice o APPR) e, infine, se si vuole risolvere la versione *soft* del problema presentata nella sezione 3.1 (piuttosto che quella standard).

3.2.2 Motif counting

Nella seconda fase, l'algoritmo ha a disposizione il dataset, pulito, sul quale effettuare il conteggio dei motifs. I motif considerati nel progetto sono tutti sotto-grafi del grafo di interazione proteina-proteina, e per questo sono non

diretti; essi sono:

- Triangoli;
- Quadrati;
- Clique di taglia 4;
- Clique di taglia 5;
- Triangoli "con coda";
- Quadrati "con coda".

Gli ultimi due motif, quelli "con coda", non riportati in Sezione 2.2, sono semplicemente quei motif ai quali viene modificata la struttura originale, aggiungendo un nodo e un lato adiacente ad uno degli altri nodi partecipanti al motif stesso; si noti che, per alcuni motif (come nel caso dei triangoli e nel caso dei quadrati), non è necessario specificare quale nodo del motif originale viene esteso con un altro lato e un altro nodo (per simmetria). In più, l'algoritmo è stato eseguito anche senza contare alcun motif, al fine di eseguire un confronto con l'algoritmo originale di HotNet2.
Con l'obiettivo di contare i motifs, in Sezione 2.2 sono stati citati i metodi implementati nel progetto: tali metodi eseguono una stampa in output dei motifs individuati nel grafo. Al fine di effettuare il conteggio dei motifs per lati, tali algoritmi sono stati modificati come segue: piuttosto che fornire in output i motif trovati, questi sono stati salvati in una lista di motifs. Da un punto di vista implementativo, i motifs sono insiemi di vertici del grafo che vi partecipano. Iterando su tale lista, ovvero per ogni insieme motif M trovato, si individua il lato $\{i, j\}$ presente tra i vertici i e $j \in M$: a questo punto l'(i, j)-esima entrata della matrice $\mathbf{W}$ viene incrementata di uno e, per

simmetria, anche l'entrata (j, i)-esima subisce la stessa sorte. Si prenda come esempio il motif "quadrato". Come spiegato in Sezione 2.2.2, si ricordi che l'algoritmo che conta quadrati stampa in output una tripla che rappresenta il grafo bipartito completo indotto dai primi due vertici appartenenti alla tripla, mentre il terzo elemento della tripla sono tutti quei vertici $u_1, ..., u_L$ che vi appartengono. L'algoritmo 3.2.1, salva tali triple in una lista e, per ognuna di esse, vengono conteggiati i lati che appartengono al grafo bipartito da essa rappresentata. Un'ultima nota va fatta per i *tailed* motifs, per i

Algorithm 3.2.1 Counting edges participating in quadrangles

Require: $G = (V, E)$

Ensure: **W**

```
L = find_quadrangles(G)
for all T ∈ L do
    v = T[1]
    w = T[2]
    U = T[3]
    for all le coppie ordinate u_i, u_j ∈ U do
        w_{v,u_i} = w_{v,u_i} + 1
        w_{v,u_j} = w_{v,u_i} + 1
        w_{u_j,w} = w_{v,u_i} + 1
        w_{u_i,w} = w_{v,u_i} + 1
        // per simmetria:
        w_{u_i,v} = w_{v,u_i} + 1
        w_{u_j,v} = w_{v,u_i} + 1
        w_{w,u_j} = w_{v,u_i} + 1
        w_{w,u_i} = w_{v,u_i} + 1
    end for
end for
```

quali contare le occorrenze di partecipazione dei lati del grafo ad un tailed motif a partire dal motif "senza coda" è semplice: prendendo come esempio il

triangolo, per individuare un'eventuale "coda" è sufficiente iterare sui vertici appartenenti al triangolo e indagare la presenza di un lato che esce dal motif, i.e. che collega il vertice ad un altro non appartenente al motif. L'Algoritmo 3.2.2 riassume quanto detto.

Algorithm 3.2.2 Counting edges participating in tailed triangles

Require: $G = (V, E)$

Ensure: **W**

```
L = find_triangles(G)
for all T ∈ L do
    u = T[1]
    v = T[2]
    w = T[3]
    // check per i vicini di u
    for all j vicino di u do
        if j ≠ v and j ≠ w then
            // trovato un triangolo con coda
            incrementa il conteggio dei lati corrispondenti (i.e. per
            {v, u}, {v, w}, {w, u}, {j, u})
        end if
    end for
    ripeti il check anche per i vicini di v e per i vicini di w
end for
```

3.2.3 Diffusione del calore

A questo punto, avendo a disposizione $\mathbf{W}$, è possibile ricavare $\mathbf{W}^t$; grazie alle considerazioni effettuate nella Sezione 3.1, la terza fase può essere svolta dall'algoritmo in tempo $O(n^2)$, prendendo le singole entrate della matrice $\mathbf{W}$ (che rappresenta il numero di lati paralleli nel grafo G_w) e dividendo per la somma della riga corrispondente (i.e. il grado del vertice considerato).

Una volta estratta la matrice di transizione $\mathbf{W}^t$, essa viene salvata su un file temporaneo: tutti i nodi disconnessi vengono filtrati, mentre quelli rimanenti re-indicizzati con cura, in maniera simile a quanto fatto nel pre-processing. Questo viene effettuato per due motivi:

1. Il file temporaneo salvato è riutilizzabile, è facile da esaminare per un eventuale *DEBUG* e rispetta lo standard adottato nel pre-processing, rendendo trasparente all'algoritmo il grafo in input;

2. Una volta esaminato il grafo di interazione proteina-proteina, non è necessario ripetere la computazione dei conteggi dei motifs, i quali sono piuttosto onerosi, in confronto con le altre fasi dell'algoritmo.

Nel quarto passaggio si esegue il processo di diffusione del calore: in primis, viene individuata la distribuzione stazionaria della random walk con restart con parametro α. Per far ciò, durante lo svolgimento della tesi, s'è scelto di implementare sia la semplice inversione di matrice (equazione 1.11), sia l'algoritmo APPR spiegato in Sezione 2.3.1. Nonostante lo sforzo implementativo, dopo aver notato che l'inversione di matrice non è troppo onerosa (dati gli input), s'è scelto di utilizzare solo l'inversione matriciale. Si è quindi ottenuta la matrice $\mathbf{E}$ (implementando la moltiplicazione righe per colonne in equazione 2.1), e da questa il grafo diretto H (come mostrato in Sezione 2.1).

3.2.4 Clusters ed *evaluations*

Nella quinta fase dell'algoritmo si esegue il clustering; come anticipato in Sezione 2.1, per fare ciò è sufficiente individuare le componenti fortemente del grafo diretto H. Per far ciò, esistono diversi algoritmi a disposizione; come approfondito in Sezione 3.2.5, per questo specifico *task*, si è scelto di affidarsi

ad una funzione di libreria già pronta, dato che questa risponde direttamente alla richiesta. Le componenti fortemente connesse ottenute sono state quindi ordinate per taglia in ordine non crescente e salvate anch'esse su un file di output; per una migliore leggibilità degli output, all'interno dei cluster viene eseguito un ordinamento lessicografico dei geni al loro interno.
Nella sesta e ultima fase del lavoro, l'algoritmo calcola FPR e TPR dei cluster ottenuti e, al variare della taglia degli stessi, effettua un *plot* di questi due coefficienti, ottenendo una curva ROC come indicato in Sezione 2.4. Per far ciò, dato il parametro δ (i.e. peso delle entrate (i, j) della matrice $\mathbf{E}$ al di sopra delle quali il grafo H ammette un lato tra i vertici i e j) e il motif scelto (e.g. triangolo), l'algoritmo calcola TPR e FPR al variare di k e salva i risultati su un apposito file di output.
Al fine di eseguire un confronto tra motif scelti e parametro δ, ogni *plot* viene eseguito fissando un δ, sovrapponendo le curve relative a diversi motif scelti, come viene presentato nella Sezione 4.2.

3.2.5 Note implementative

Durante il lavoro svolto per la tesi, quando possibile, s'è scelto di implementare la maggior parte degli algoritmi descritti senza ricorrere interamente a funzioni già pronte o ri-utilizzando parti del codice di *HotNet2*. Questa scelta è stata effettuata principalmente perché *HotNet2* non accetta in input matrici pesate, e modificarne il codice può risultare più complicato.
Il linguaggio di programmazione scelto per le implementazioni è *Python*, che risulta essere molto popolare in questo momento storico, soprattutto per quanto riguarda l'analisi dei dati e il Machine Learning in generale. L'uso di Python in questa tesi è ben motivato, e non solo per la sua semplicità d'uso: Python, in particolare, agevola incredibilmente l'acquisizione dell'input, sia

da linea di comando, che da file; inoltre, s'è fatto un uso intenso di diverse librerie a disposizione del linguaggio.
Il codice degli esperimenti per la tesi è ricco di funzioni e strutture dati ricavati dalla libreria *NumPy*, ben nota nel mondo della *Data Science*: *NumPy* serve infatti a gestire operazioni vettoriali e matriciali all'interno del proprio codice. Si è invece fatto un uso più modesto della libreria *NetworkX*, dato che la maggior parte degli algoritmi è stato implementato interamente da zero; *NetworkX* serve infatti a gestire operazioni su grafi in maniera estremamente semplice. In particolare, sono state utilizzate solo quelle funzioni che individuano le componenti fortemente connesse in un grafo e quelle che in esso ne elencano tutte le possibili cricche. Inoltre, dato che le esecuzioni dell'algoritmo sono state lanciate su un server esterno (si veda Sezione 4), la libreria *Paramiko* è risultata particolarmente utile per il lancio dei *jobs* e la comunicazione con questo. Infine, per quanto riguarda i grafici, la libreria *Matplotlib* è risultata fondamentale e di facile utilizzo.

Capitolo 4

Esperimenti

In questa sezione vengono riportati i risultati ottenuti dagli esperimenti, ovvero i sotto-grafi mutati riportati dall'algoritmo, completi di curve ROC. Tutti gli esperimenti sono stati eseguiti sul *cluster* di calcolo *BLADE* dell'Università di Padova.
Nella Sezione 4.1 vengono mostrati i *dataset* utilizzati per gli esperimenti, completi di breve descrizione e visualizzazione, a scopo di esempio.
Nella Sezione 4.2, invece, vengono mostrati i risultati ottenuti, come ad esempio i *plot* delle curve ROC.

4.1 *Dataset, Setup, Environment*

Il grafo di interazione proteina-proteina utilizzato per gli esperimenti nella tesi è HINT+HI2012, uno dei tre utilizzati dagli autori di *HotNet2* [4]. Questo dataset è stato creato dagli autori unendo due diversi database di interattomi (i.e. insieme delle interazioni tra geni), HI-2012, più recente, e HINT, definito "di alta qualità" in termini di specificità e sensitività [4]. HINT+HI2012 consiste di 9859 proteine, con 40705 iterazioni fra queste; in

altre parole, il grafo in input ha quasi 10^4 nodi e poco più di $4 \cdot 10^4$ lati. È possibile visualizzare una piccola porzione del grafo in esame in Figura 4.1. Anche il calore da diffondere all'interno di tale grafo è uno degli *score*

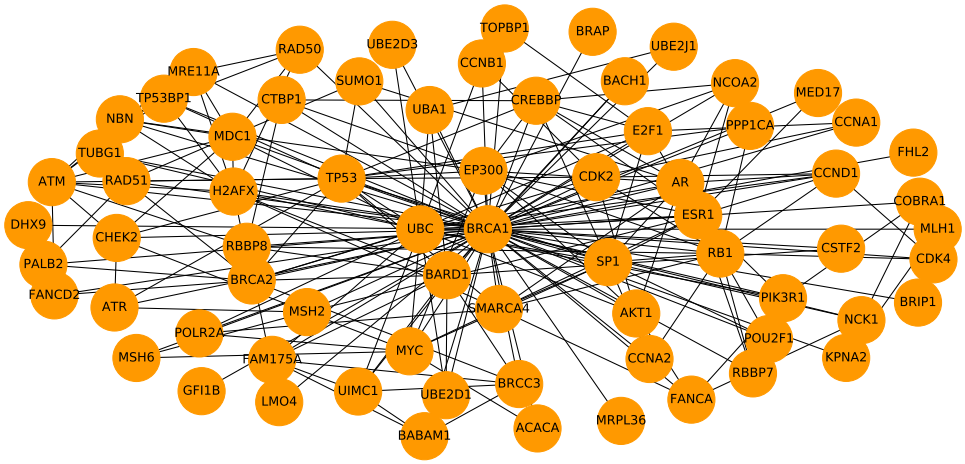


Figura 4.1: Una porzione, i.e. un sotto-grafo, ricavato dal grafo usato per gli esperimenti. Il sotto-grafo in Figura è stato ricavato mettendo al centro il nodo "BRCA1" (Breast Cancer 1, gene oncosopressore nel cancro alla mammella) e prendendo tutti i suoi vicini (e le interazioni tra loro); tra questi, si notino i geni "BRCA2" (che ha una funzione simile a "BRCA1"), "TP53" (il soppressore tumorale più noto) e "PIK3R1" (proteina non legata al tumore al seno direttamente, ma che muta in esso).

utilizzato per le sperimentazioni in [4]: il nome del *dataset* è MutSigCV, ed è stato ricavato attraverso il calcolo della significatività statistica delle mutazioni dei singoli geni all'interno di un certo numero campioni biologici. La procedura per tale calcolo, basata su BMR (Background Mutation Rate), non è oggetto della tesi e non verrà specificata qui. In ogni caso, MultiSigCV presenta un'alta specificità nei suoi dati [4]. In tabella 4.1 è possibile osservare un esempio di score assegnati ad alcuni geni del database. In tabella 4.2, invece, viene mostrata una parte dell'elenco di geni noti come cancerosi alla comunità scientifica, utilizzati dall'algoritmo per il calcolo di TPR e FPR, come specificato in Sezione 2.4. Per quanto riguarda la scelta dei parametri

Gene	Heat
BRCA1	7.68813323706
BRCA2	1.60947342773
TP53	12.0012468235
PIK3R1	12.0012468235
BRIP1	0.0
CHEK2	0.596371154076
ATR	0.0
AKT1	8.6794468314
BACH1	0.0
MSH6	3.43194572633

Tabella 4.1: Alcuni score associati ad alcuni dei geni mostrati in Figura 4.1; si noti come è possibile avere uno score nullo per alcuni geni.

per l'esecuzione, questa è stata fatta senza operazione di tuning. Leiserson et. al., nel parametrizzare *HotNet2*, eseguono un'approfondita analisi teorica e sperimentale per il *tuning* del loro modello [4]. Nel lavoro di questa tesi, invece, il parametro α è stato fissato a $\alpha = 0.6$ per ogni *run*; è chiaro che, più grande è α, maggiore sarà la diffusione del calore e ci si potrà aspettare quindi sotto-reti mutate più grandi (e viceversa): il valore di 0.6 è parso, semplicemente, ragionevole. Allo stesso tempo, in tabella 4.3 è possibile vedere la lista del parametro δ scelto per ogni *run* dell'algoritmo, i cui risultati sono riportati in Sezione 4.2. Il parametro in questione caratterizza i risultati forniti, anche al variare dei motifs da analizzare nella rete (tabella 4.4. Si noti come il motif "Nessuno" indica la scelta di non usare alcun *graphlet-counting* nel grafo, per effettuare un confronto con l'approccio privo di analisi su strutture ad alto livello. Infine, ogni configurazione viene ripetuta nella versione *soft* del problema, al fine di eseguirne un confronto, come indicato in Sezione 3.1.

Cosmic classic	**Other genes**	**Recently emerging genes**
RB1	MLL2	TGFBR2
FGFR3	CDK12	TBL1XR1
KIT	IL7R	ARID5B
BRCA1	GATA2	ERBB4
CDK4	ARID2	ING1
NRAS	BAP1	CDKN1B
CDH1	CD79B	MAP3K1
CDKN2A	MYD88	NFKBIA
VHL	AKT1	MLL
KRAS	PIK3R1	RBM10
CTNNB1	FBXW7	HIST1H3B
BRAF	SMARCA4	SPOP
EGFR	SF3B1	MYCN
NOTCH1	DNMT3A	MTOR
SMARCB1	FAM46C	LCP1
CEBPA	XPO1	DIS3
MET	PRDM1	MAP2K1
ERCC2	BCOR	SMAD2
WT1	PBRM1	NSD1
CREBBP	GATA3	FOXQ1
TSC1	PIK3CA	SOX17
HRAS	PHF6	CD70
PTEN	MLL3	GNA13
TP53	ASXL1	RIT1

Tabella 4.2: Alcuni dei geni noti per presentare mutazioni nel cancro [8]. La prima e la seconda colonna fanno riferimento ad un database noto alla comunità scientifica, la quale riconosce come *cancer genes* tale elenco. La terza colonna include geni riconosciuti come cancerosi più recenti.

Riassumendo, negli esperimenti effettuati si osservano le curve ROC al variare dei motif scelti per l'analisi, fissato un ben determinato valore di δ per

δ
0.0005
0.0055
0.0192
0.0670
0.1005
0.1340
0.1508
0.2261

Tabella 4.3: I valori del parametro δ scelti per l'algoritmo.

Motifs
Triangolo
Quadrato
Clique, $k = 4$
Clique, $k = 5$
Triangolo con coda
Quadrato con coda
Nessuno

Tabella 4.4: I motifs scelti per le run dell'algoritmo.

ogni *run*.

4.2 Risultati

Al fine di osservare la struttura del grafo di interazione proteina-proteina, in tabella 4.5 si possono osservare il numero di occorrenze dei vari motif scelti all'interno di esso. Nella stessa tabella, vengono anche elencate le taglie degli insiemi V ed E_w del grafo pesato non orientato $G_w = (V, E_w)$ ottenuto attra-

verso il *motif counting*, come spiegato in Sezione 3.1; è interessante osservare l'impatto delle occorrenze dei motif sulla taglia del grafo.

In tabella 4.6 viene mostrato un esempio di sotto-grafi mutati ottenuti dal-

Motifs	**Numero di occorrenze**	$\|\mathbf{V}\|$	$\|\mathbf{E_W}\|$
Triangolo	22152	3790	36310
Quadrato	519900	6004	67934
Clique, $k = 4$	14840	1685	16812
Clique, $k = 5$	11909	769	7974
Triangolo con coda	5055389	9027	76406
Quadrato con coda	294466311	9512	80226

Tabella 4.5: Motifs trovati all'interno del grafo di interazione proteina-proteina dato, e, in corrispondenza di questi, la taglia del grafo $G_W = (V, E_W)$ ottenuto. In alcuni casi, la taglia di E_W è più alta della taglia di E nel grafo originale, poiché il grafo G_W è diretto (come spiegato in Sezione 2.1), e il numero di archi può essere al più il doppio dell'originale.

l'algoritmo[1]; in particolare, si possono trovare i sotto-grafi estratti quando l'analisi viene effettuata con il motif "Clique" (con $k = 4$, *soft version*), con $\alpha = 0.2$ e $\delta = 0.0267$. L'esempio mostrato rappresenta bene i risultati ottenuti negli esperimenti di questa tesi: infatti, indipendentemente dalla configurazione dei parametri e dalla scelta dei motifs, l'algoritmo ritorna un cluster di taglia molto grande e pochi altri di taglia molto piccola. Ciò è dovuto soprattutto alla conformazione del grafo di interazione proteina-proteina, con molti nodi che hanno un grado molto elevato e altrettanti con grado piccolo.

In Figura 4.2, si possono osservare i grafici ROC, che misurano TPR e FPR per ogni possibile configurazione. Ogni *plot* corrisponde ad un fissato valore di δ, e le singole curve rappresentano l'andamento degli indici al variare della

[1]È possibile trovarne un'enumerazione completa (i.e. per ogni possibile configurazione) in `https://github.com/lucavenir/hogcpcnn/tree/master/out/HINT%2BHI2012`

#	Taglia	Sotto-grafo mutato estratto
1	212	..., **BRCA1**, BRCA2, .., **PIK3CA**, **PIK3R1**, ..., TNF, **TP53**, ...
2	3	RAD21, SMC3, **STAG2**
3	2	NR4A2, RXRA
4	2	PCBP1, QKI
5	2	GOLGA5, RAB1A
6	2	GOLGB1, SLC2A3

Tabella 4.6: Sotto-grafi mutati trovati all'interno del grafo di interazione proteina-proteina attraverso l'analisi effettuata con il Motif "Clique" ($k = 4$, *soft version*); parametri: $\delta = 0.0267$, $\alpha = 0.2$. Il sotto-grafo più grande, da 212 elementi, è stato elencato solo in parte. Gli elementi in grassetto sono i *cancer genes* già noti, che si possono trovare in tabella 4.2. Si noti come questa particolare configurazione dei parametri non è stata inclusa nelle sperimentazioni, ma è qui solo a scopo di esempio.

taglia dei sotto-grafi estratti. Dai risultati si possono evincere due considerazioni.

In primo luogo, all'aumentare dei geni inclusi per il calcolo di TPR e di FPR (i.e. al diminuire di k), i due coefficienti aumentano; ciò non sorprende, poiché (come spiegato in Sezione 2.4), includendo più proteine ci si può aspettare al più un aumento dei *true positives* o dei falsi positivi, ma non una diminuzione di questi.

In secondo luogo, se si prendessero le curve come unica misura della qualità dei sotto-grafi mutati ritornati, si potrebbe concludere che, all'aumentare di δ, la bontà dei cluster diminuisce; in realtà, la misura che danno TPR e FPR è indicatrice di quanto l'algoritmo ritorni geni mutati già noti alla comunità scientifica. In altre parole, se lo *score* dell'algoritmo fosse "ottimo" (i.e. $TPR \approx 1$, $FPR \approx 0$), probabilmente sarebbe meno incline a ritornare sotto-grafi mutati *mai visti prima*; l'obiettivo della *cancer genomics* è, infatti, come già specificato nell'introduzione (Sezione 1), quello di indivi-

duare nuove mutazioni da studiare. In ogni caso, l'analisi effettuata grazie alle curve ROC permette di stabilire la "bontà" dei risultati ottenuti, poiché un TPR molto basso (e un FPR alto) non indicherebbe *necessariamente* un risultato rilevante; un buon score serve, dopotutto, a certificare in qualche modo l'attendibilità dei risultati, poiché ci si aspetta che un buon algoritmo ritorni *in ogni caso* proteine mutate già note, se funziona correttamente.
Le curve ritornate dall'algoritmo indicano quindi un buon score; in ogni grafico esiste sempre almeno una tecnica (i.e. motif analizzato) che ritorna un TPR di almeno 0.7 (includendo tutti i geni nei sotto-grafi mutati). In più, si noti come, indipendentemente dal motif analizzato, gli score sono migliori per il motif "triangolo" e per il motif "triangolo con coda". Un'ulteriore osservazione fa fatta per il motif "5clique" (clique di taglia 5): pur avendo sempre score più bassi rispetto alle altre tecniche, è la tecnica che produce un FPR più piccolo e che è più sensibile al cambiamento del parametro k. Abbassandolo, infatti, vengono inclusi dei sotto-grafi mutati che evidentemente contengono un gran numero di geni *true positives*. Si può quindi dedurre che la capacità di fornire sotto-grafi mutati non noti alla comunità è molto ridotta in questo motif.
Al fine di effettuare un confronto con *HotNet2*, è utile la curva etichettata come "no" nei grafici di Figura 4.2. Tale curva indica il clustering effettuato *senza* l'analisi di *network motifs*. In altre parole, è il risultato che fornisce direttamente *HotNet2*. Si può notare come tale tecnica non primeggia mai nel grafico: l'analisi effettuata con triangoli e clique di taglia 4 hanno sempre score migliori, mentre altre, come i quadrati con coda, hanno score minori. A tal proposito, ulteriori considerazioni vengono effettuate in Sezione 5.1. Infine, si può osservare che, effettuando l'analisi di motif "quadrato con coda", i risultati (i.e. (TPR, FPR) per tale tecnica) cambiano sensibilmente all'au-

mentare di δ. Infatti, per valori di δ piccoli, il cluster ritornato dall'algoritmo è paragonabile a quello ritornato con altre tecniche; quando δ aumenta, sia i valori di TPR che quelli di FPR diminuiscono sensibilmente, fino ad arrivare ad un minimo di $FPR \approx 0$ e $TPR \approx 0.35$. Ciò implica due conclusioni: in primis, un FPR così basso indica che l'analisi effettuata attraverso questo motif non aiuta nella ricerca di nuovi sotto-grafi; in secondo luogo, inoltre, un TPR basso implica che la tecnica non è ottimale nemmeno per rilevare geni già noti. In realtà, considerazioni simili possono essere svolte tutti gli altri motif, tranne per i "triangoli con coda" e le clique di taglia 4; quest'ultimi, infatti, sembrano essere non particolarmente sensibili all'aumento del parametro δ.

In Figura 4.3, invece, si possono osservare i risultati per quanto riguarda la *soft version* del problema formulato in Sezione 3. Mantenere gli archi che non partecipano a nessuna struttura ad alto livello impatta fortemente sui risultati: ad esempio, per quanto riguarda $\delta = 0.000496$, in tutte le tecniche usate si ottiene un solo cluster (molto grande), con coppia (TPR, FPR) praticamente identica in tutti i motif. Emerge, nuovamente, il motif "triangolo con coda", ma nella versione *soft*, la performance sembra peggiorare notevolmente all'aumentare di δ: un comportamento molto diverso rispetto ai grafici di Figura 4.2.

Spicca nettamente, invece, il motif "clique" ($k = 5$); in questo caso, a scapito di un leggero aumento del coefficiente FPR rispetto agli altri, con questa tecnica si ottiene sempre un $FPR \approx 0.9$. Questo è sicuramente dovuto alla natura del grafo, con gradi dei nodi elevati: una clique cattura al meglio tale struttura, e grazie alla versione *soft* il calore ha la possibilità di diffondersi anche attraverso altri lati che non partecipano direttamente ad un sotto-grafo completo (di taglia 5).

Come ultima nota, è possibile notare un calo sensibile delle performance di tutte le tecniche implementate, quando δ aumenta, esattamente come accadeva in Figura 4.2: solo le analisi effettuate tramite il motif "quadrato" e "clique" ($k = 5$) sembrano essere meno sensibili al parametro.

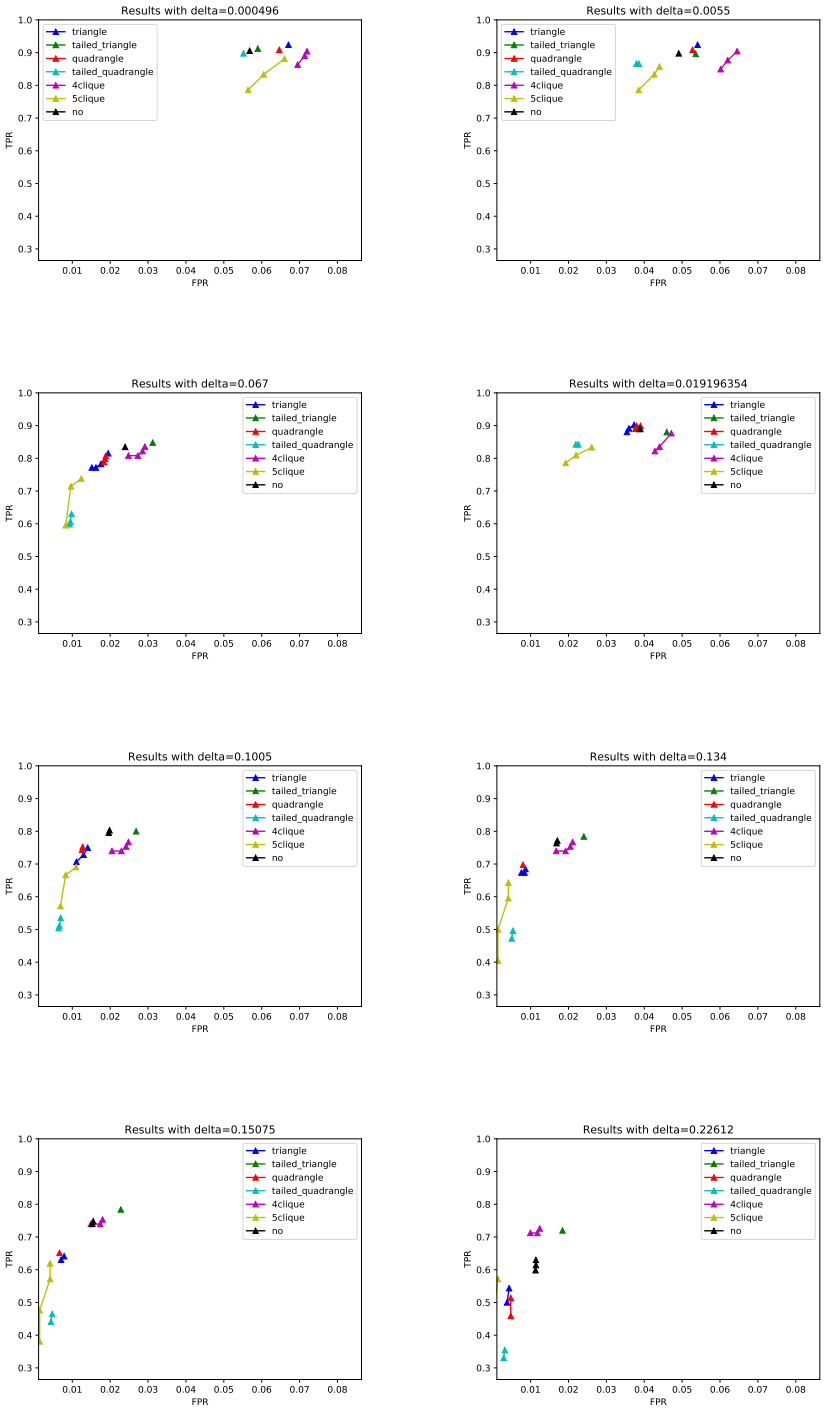


Figura 4.2: Curve ROC per ogni possibile δ scelto, al variare del motif analizzato.

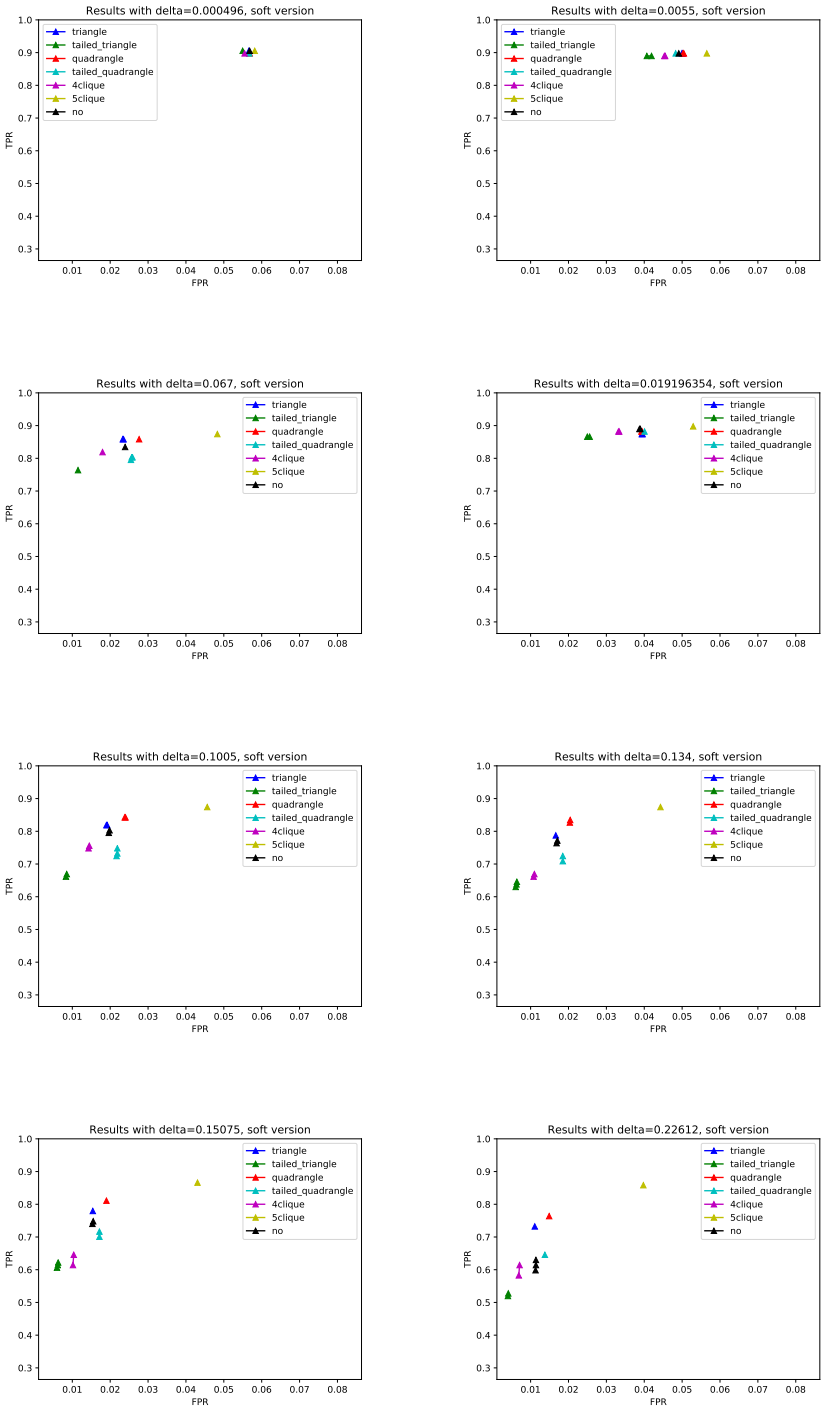


Figura 4.3: Curve ROC nella versione *soft* del problema.

Capitolo 5

Conclusioni

Con gli esperimenti mostrati in Sezione 4, la tesi indica una nuova tecnica per estrarre sotto-grafi mutati nei tumori, similmente a quanto fatto in *HotNet2*, ma tramite l'analisi di *network motifs*, ovvero strutture su grafi ad alto livello (e.g. triangoli), raggiungendo così l'obiettivo prefissato in Sezione 3.1.
Le curve ROC mostrate precedentemente permettono di fare un'immediata prima considerazione: l'algoritmo pensato e implementato in questa tesi estrae correttamente un buon numero di *cancer genes* dal database fornito in input. La maggior parte delle tecniche implementate, indipendentemente dai motif scelti per effettuare l'analisi, fa ottenere diversi *true positives* (TPR molto elevato), suggerendo che tali tecniche ritornano risultati affidabili.
Dai grafici delle *ROC curves* si evince un ruolo centrale nell'analisi del motif "triangolo con coda", che mantiene uno score elevato sia per TPR che per FPR; indipendentemente dal parametro δ, un alto *hit rate* conferma la bontà della tecnica, mentre un valore non particolarmente basso di FPR evidenzia una buona capacità di invidiuare nuovi geni.
Il parametro δ, quando elevato, mette in crisi l'algoritmo: diverse tecniche, e in maniera più evidente "*5-clique*" e "*tailed quadrangle*", falliscono nel ri-

tornare sotto-grafi con mutazioni utili alla comunità scientifica, né già noti. Quando viene implementata la versione *soft* del problema, i risultati cambiano sensibilmente; se l'analisi viene effettuata attraverso il motif "triangolo con coda", si ottengono risultati peggiori, con un basso TPR e un FPR praticamente nullo. Spicca invece, in questa versione, il ruolo delle *clique*; dove, se $k = 5$, si ottengono risultati migliori, i.e. coefficienti di valutazione più elevati.

5.1 Possibili sviluppi

I risultati ottenuti in Sezione 4.2 suggeriscono che l'analisi dei network motifs può essere di fondamentale importanza nel contesto della *cancer genomics*; tuttavia, come anticipato, nel lavoro svolto nella tesi non si prendono in considerazione ulteriori misure di qualità, in particolare per misurare la *significatività* statistica dei sotto-grafi mutati ritornati dall'algoritmo. Di fatto, un buon rapporto TPR/FPR non implica necessariamente una scoperta significativa nel contesto dell'onco-genomica (anzi); come già anticipato, FPR alti evidenzierebbero un buon numero di "nuovi" geni ritornati dall'algoritmo, ma senza un test statistico non è possibile validare tali risultati.
Un test d'ipotesi sui cluster ritornati non solo completerebbe il lavoro svolto, ma permetterebbe di svolgere una migliore parametrizzazione del problema. Infatti, gli autori di *HotNet2*, come già precedentemente citato, eseguono un'ampia analisi al fine di individuare una configurazione dei parametri ottimale, ovvero il *set-up* di δ e α; a tal scopo gli autori randomizzano più volte sia la disposizione dei lati all'interno dei grafi, sia lo score (*heat*) in input, scegliendo la coppia (α, δ) che ottimizza gli score (statistici e non). In un

lavoro futuro, sarà di fondamentale importanza impostare tali parametri con la stessa tecnica implementata, dove però la randomizzazione andrà ripetuta per ogni motif scelto, dato che si ottengono grafi pesati differenti.

Ringraziamenti

Quello che ho ricevuto in questi anni è fantastico e meraviglioso.
Mi sono sempre sentito una persona estremamente fortunata, e me ne rendo conto sempre di più: sono sempre circondato da cose e persone incredibili.
Ricevo, ogni giorno, occasioni di crescita e di lavoro per me stesso, doni unici, e per questo ringrazio profondamente.
Grazie a chi mi è stato vicino nei periodi più bui, ma grazie anche a chi mi ha messo in difficoltà, permettendomi di crescere. Grazie anche a chi mi ha permesso di osservare, di aprire un'altra prospettiva sul prossimo.
Grazie alla pazienza di chi mi ha accolto, sempre, nonostante gli spigoli.
Un ringraziamento particolare alle guide che mi hanno accompagnato fin qui.

E se per caso io non fossi altro che brezza?
E se per caso tu non mi ritenessi all'altezza?
Io ti direi che ho un grido che sfigura il mio volto
e che per ogni passo, io,
sarò la tua orma.

Luca Venir *Udine, 03 dicembre 2019*

Bibliografia

[1] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[2] C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2006.

[3] H. Yin, A. R. Benson, J. Leskovec, and D. F. Gleich, "Local higher-order graph clustering," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 555–564.

[4] M. D. Leiserson, F. Vandin, H.-T. Wu, J. R. Dobson, J. V. Eldridge, J. L. Thomas, A. Papoutsaki, Y. Kim, B. Niu, M. McLellan *et al.*, "Pan-cancer network analysis identifies combinations of rare somatic mutations across pathways and protein complexes," *Nature genetics*, vol. 47, no. 2, p. 106, 2015.

[5] N. Chiba and T. Nishizeki, "Arboricity and subgraph listing algorithms," *SIAM Journal on computing*, vol. 14, no. 1, pp. 210–223, 1985.

[6] M. Danisch, O. Balalau, and M. Sozio, "Listing k-cliques in sparse real-world graphs," in *Proceedings of the 2018 World Wide Web Conference*. International World Wide Web Conferences Steering Committee, 2018, pp. 589–598.

[7] R. Andersen, F. Chung, and K. Lang, "Local graph partitioning using pagerank vectors," in *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*. IEEE, 2006, pp. 475–486.

[8] H. Horn, M. S. Lawrence, C. R. Chouinard, Y. Shrestha, J. X. Hu, E. Worstell, E. Shea, N. Ilic, E. Kim, A. Kamburov *et al.*, "Netsig: network-based discovery from cancer genomes," *Nature methods*, vol. 15, no. 1, p. 61, 2018.