UNIVERSITÀ DI PADOVA **SCUOLA DI INGEGNERIA**

TESI DI LAUREA

Analisi e confronto di misure di distanza per uncertain graph

Laureando: Dennis Dosso

Relatori: Andrea Alberto Pietracaprina, Geppino Pucci
Correlatore: Matteo Ceccarello

Corso di Laurea Magistrale in Ingegneria Informatica

Data Laurea: 10 Ottobre 2016

Anno Accademico 2015/2016

Abstract

In questa tesi viene affrontato il problema della scelta di una metrica di distanza adatta al problema dei k-nearest neighbour nell'ambito degli uncertain graph, grafi con una distribuzione di probabilità sull'insieme dei lati. In particolare, vengono messe a confronto le misure di Median-Distance, utilizzata in [11] e una nuova misura, qui chiamata α-closeness.

Per l'α-closeness sono inoltre proposti degli algoritmi predisposti di stopping condition per velocizzare l'esecuzione e viene studiato l'impatto di tali condizioni sull'efficienza computazionale degli stessi.

Ringraziamenti

Alla mia famiglia per avermi permesso di arrivare a questo traguardo; ai miei professori che sono stati fonte di ispirazione e sostegno nel corso degli anni ed hanno insegnato più delle semplici nozioni accademico; agli amici per avermi ricordato quanto ognuno di noi sia importante a prescindere.

One equal temper of heoric hearts,
Made weak by time and fate, but strong in will
To strive, to seek, to find, and not to yield.

Lord Alfred Tennyson

Indice

1 Introduzione **1**
- 1.1 Obiettivi della tesi . . . 2
- 1.2 Risultati raggiunti . . . 3
- 1.3 Struttura della tesi . . . 4

2 Uncertain Graph **5**
- 2.1 Definizione . . . 5
- 2.2 Alcune applicazioni su Uncertain Graphs . . . 7
  - 2.2.1 Core Decomposition of Uncertain Graphs . . . 8
  - 2.2.2 Clustering Large Probabilistic Graphs . . . 9
  - 2.2.3 Distance-Constraint reachability Computation in Uncertain Graphs . . . 10
  - 2.2.4 The pursuit of a Good Possible World . . . 11

3 Metriche di distanza **13**
- 3.1 La Median-Distance . . . 13
  - 3.1.1 Calcolo della Median-Distance . . . 14
- 3.2 La metrica α-closeness . . . 16
  - 3.2.1 Le ragioni per una nuova metrica . . . 16
  - 3.2.2 α-closeness . . . 17
  - 3.2.3 Approssimare l'α-closeness . . . 18

4 Il problema dei k-NN negli Uncertain Graphs **21**
- 4.1 Il calcolo dei k-NN in base alla expetced α-closeness . . . 21
  - 4.1.1 Un primo semplice algoritmo di approssimazione . . . 23
- 4.2 Stopping condition per algoritmi più veloci . . . 24
  - 4.2.1 Una prima stopping condition . . . 25
  - 4.2.2 Una seconda stopping condition . . . 27
  - 4.2.3 Una terza stopping condition . . . 28

5 Esperimenti **33**
5.1 Setup degli esperimenti 33
5.1.1 Inizializzazione e lettura del grafo 34
5.1.2 Nodo sorgente e ground truth 34
5.1.3 Rendere il grafo probabilistico 35
5.1.4 Calcolo dell'α-closeness e della Median-Distance dei nodi . . 36
5.2 Primo esperimento 37
5.2.1 Variante con scelta del nodo sorgente 45
5.3 Secondo esperimento 49
5.4 Terzo Esperimento 52
5.4.1 Tempo impiegato nell'ordinamento 56

6 Osservazioni finali e sviluppi futuri **57**

Elenco delle figure

2.1 Un uncertain graphs con probabilità lievemente differenti ai lati . . 6

3.1 Esempio particolare di CDF. 16
3.2 Esempio di algoritmo basato su Median-Distance 17

5.1 I valori di true positive, true negative, false positive e false negative ottenuti dai due algoritmi sui quattro grafi per $\alpha = 1$ e $\beta = 0.3$. . . 38
5.2 I valori di true positive, true negative, false positive e false negative ottenuti dai due algoritmi sui quattro grafi per $\alpha = 1$ e $\beta = 0.5$. . . 39
5.3 I valori di true positive, true negative, false positive e false negative ottenuti dai due algoritmi sui quattro grafi per $\alpha = 1$ e $\beta = 0.8$. . . 40
5.4 Le precision e recall ottenute dagli esperimenti sulle 4 mesh con $\alpha = 1$ e $\beta = 0.3$ 42
5.5 Le precision e recall ottenute dagli esperimenti sulle 4 mesh con $\alpha = 1$ e $\beta = 0.5$ 43
5.6 Le precision e recall ottenute dagli esperimenti sulle 4 mesh con $\alpha = 1$ e $\beta = 0.8$ 44
5.7 I valori di tp, tn, fp e fn ottenuti dalle mesh utilizzando il nodo centrale come sorgente, $\alpha = 1$ e $\beta = 0.5$. 46
5.8 I valori di tp, tn, fp e fn ottenuti dalle mesh utilizzando un nodo nel mezzo di un lato come sorgente, $\alpha = 1$ e $\beta = 0.5$. 47
5.9 I valori di tp, tn, fp e fn ottenuti dalle mesh utilizzando il nodo in angolo 0 come sorgente, $\alpha = 1$ e $\beta = 0.5$. 48
5.10 I valori di precision e recall ottenuti dalle mesh utilizzando $\alpha = 0.3$ e $\beta = 0.5$ 49
5.11 I valori di precision e recall ottenuti dalle mesh utilizzando $\alpha = 0.5$ e $\beta = 0.5$ 50
5.12 Le precision e recall ottenute dagli esperimenti sulle 4 mesh utilizzando $\alpha = 2$ e $\beta = 0.5$. 51
5.13 Andamenti dei tempi sulla mesh10. 53
5.14 Andamenti dei tempi sulla mesh32. 54
5.15 Andamenti dei tempi sulla mesh64. 54

5.16 Andamenti dei tempi sul grafo rome99. 55
5.17 Impatto del sorting 56

Capitolo 1

Introduzione

Grafi complessi come quelli associati alle reti biologiche, sociali e di comunicazione spesso portano con sé dell'incertezza, di frequente derivante da misurazioni rumorose, modelli di inferenza o da processi di perturbazione introdotti appositamente per proteggere in qualche modo la privacy legata alle informazioni che si stanno manipolando. In molti domini di applicazione, come quelli legati alle reti citate, si ha che i grafi funzionano come modelli migliori rispetto ad esempio ai database relazionali [11]. Introdurre incertezza all'interno dei grafi porta ad ottenere i cosiddetti **uncertain (probabilistic) graphs**, ossia grafi sui cui lati è presente una distribuzione di probabilità. In questa tesi si fa uso della *possible world semantic* [11], che presuppone che gli eventi di esistenza dei singoli lati siano indipendenti tra di loro, e che vede un uncertain graph come l'insieme dei 2^m possibili grafi deterministici che possono generarsi dal sampling dei lati. Ognuno di questi viene chiamato *possible world* del grafo probabilistico.

Le reti biologiche costituiscono una delle loro maggiori applicazioni. In esse, i nodi rappresentano geni e proteine, e i lati le interazioni che si vengono a creare tra di loro e che sono state individuate tramite esperimenti. Questi ultimi possono essere soggetti a rumore ed errori di misurazione in laboratorio, conseguentemente ogni lato è associato con un valore di incertezza [2]. Nelle PPI network (*Protein to Protein Interaction networks*) sono state stabilite sperimentalmente delle interazioni per un numero limitato di coppie di proteine [9]. Identificare i vicini di una proteina in tali reti può essere determinante per individuare nuove interazioni [12]. Per questo motivo le query per i k-nearest neighbor (query k-NN, ossia la ricerca dell'insieme dei primi k vicini di un particolare nodo) possono essere utilizzate per individuare buoni candidati link, la cui validità può poi in un secondo momento essere ulteriormente controllata tramite ulteriori esperimenti [11].

Nelle reti sociali l'incertezza deriva da molte ragioni. La probabilità di un lato può rappresentare l'incertezza di una *link-prediction* che si intende fare, o l'influenza di una persona su di un altra, come si sperimenta nel *viral marketing*

[1]. In questo contesto, può essere interessante porre query che chiedano quali sono le prime k persone che possono essere influenzate da un altro soggetto.

Nelle reti mobile ad-hoc le connessioni tra i nodi variano come effetto della loro mobilità e, quindi, della variabilità delle relative istanze. La connettività tra i nodi può essere stimata utilizzando misurazioni e la nozione della *delivery probability* può essere utilizzata per quantificare la probabilità che un dato nodo possa riuscire a far arrivare un pacchetto ad un altro nodo [6]. In questo caso, le query k-NN possono essere utilizzate per affrontare il *probabilistic-routing problem.*

Si comprende bene quindi come query di questo tipo possano ben servire come operatori primitivi per task come la link prediction, il clustering, la classification e il graph mining.

1.1 Obiettivi della tesi

In questa tesi viene affrontato il problema della ricerca dei *k-nearest neighbors* in un uncertain graph con un focus particolare sulla scelta della migliore misura di distanza per affrontare con successo tale task. Il problema originale dei k-NN nei grafi deterministici è molto semplice: dato un nodo sorgente s, ci si chiede quale sia l'insieme dei primi k nodi ad esso più vicini (l'insieme potrebbe avere cardinalità maggiore di k nel caso gli ultimi nodi della lista presentino la stessa distanza dalla sorgente). Un simile problema è facilmente affrontabile tramite un qualsiasi algoritmo di esplorazione del grafo, come BFS o Dijkstra nel caso rispettivamente di grafi non pesati e pesati. Diversamente, in un grafo probabilistico, dove un lato non esiste con certezza ma possiede una certa probabilità di esistenza, il calcolo è più difficile in quanto non è più immediata la definizione di distanza tra due vertici. Non avendo un modo immediato ed intuitivo come nei grafi deterministici di definire la distanza tra una sorgente s ed un qualsiasi altro vertice del grafo, diventa più difficile definire il problema dei k-NN all'interno di un probabilistic graph. La query richiede di venire a conoscenza dell'insieme dei primi k vertici più vicini alla sorgente, ma che cosa significa a questo punto "essere vicini" ad un altro nodo all'interno di un uncertain graph?

Si capisce bene come sia di fondamentale importanza per affrontare il problema fissare prima di tutto la **metrica di distanza** che si vuole utilizzare. Per questa scelta è ovviamente necessario tenere in considerazione il tipo di task che si sta affrontando. Vi possono infatti essere diverse metriche per conteggiare la distanza tra nodi in un uncertain graph. In [11] vengono citate alcune semplici metriche di distanza qui riportate per dare qualche esempio. Considerati due nodi arbitrari s e t:

Most-Probable-Path-Distance È definita come *la lunghezza del cammino più probabile tra i nodi s e t.*

Reliability Una semplice alternativa è considerare la probabilità che ci sia un path tra s e t come un indicatore di closeness per i due.

Most-Probable-Distance, Median-Distance, Expected-Distance Si può considerare la distribuzione di probabilità sulle distanze che si possono trovare tra i due nodi (una distanza d avrà come probabilità la somma delle probabilità dei possible world nei quali i due nodi sono a distanza esattamente d) e, in base a questa distribuzione, si può trovare la distanza con la probabilità maggiore, la distanza mediana e la distanza attesa (escludendo nel calcolo di questa la distanza ∞).

Ovviamente tutte queste distanze hanno caratteristiche differenti e possono dare soluzioni differenti. Molte altre possono poi essere definite, ognuna con le sue caratteristiche peculiari. Fondamentale diventa perciò comprendere, a seconda del problema in esame, quale sia la metrica di misura più adatta e perché proprio quella rispetto ad un'altra.

In [11], gli autori si concentrano su tre metriche, cioè *Median-Distance*, *Majority-Distance* ed *Expected-Reliable-Distance*. Il loro obiettivo è mostrare come queste metriche (di fatto, nei loro test lo fanno principalmente per la Median-Distance) possano ottenere risultati migliori rispetto alle altre appena citate, come Most-Probable-Path e Reliability.

In questa tesi vengono messe in evidenza alcune caratteristiche critiche della Median-Distance, come il fatto di essere molto influenzata da piccole variazioni nella probabilità sui lati, oltre che presentare una discutibile capacità di poter approssimare, tramite metodo Monte Carlo, la distanza reale. Queste potrebbero portare la Median-Distance a non dimostrarsi la scelta migliore come metrica di distanza nell'approccio del problema dei k-NN,

Conseguentemente si definisce qui una nuova metrica chiamata α-*closeness*, per poi quindi applicarla al problema dei k-NN su grafi probabilistici e confrontarla con la Median-Distance sia da un punto di vista teorico che pratico, per scoprire quale delle due possa ritenersi più adeguata a identificare i vicini corretti di un nodo. Inoltre, si vuole studiare come la scelta di diversi parametri possa influenzare le prestazioni delle due misure. Infine, si vogliono cercare delle stopping condition che possano riuscire a ridurre il tempo di esecuzione dell'algoritmo basato su α-closeness.

1.2 Risultati raggiunti

Tra i risultati raggiunti in questa tesi vi è un solido framework teorico che permette di evidenziare i punti di forza dell'α-closeness rispetto alla Median-Distance. Tra questi, si ha che per la prima si può stabilire una relazione stretta tra numero di

sample e qualità dell'approssimazione ottenuta, mentre per la seconda la relazione è molto più lasca e non permette di scegliere facilmente il numero di sample per ottenere una garanzia di approssimazione. I risultati che si ottengono negli esperimenti sono alla pari della Median-Distance se non superiori, soprattutto nei casi in cui vi sia un grande noise sui lati del grafo.

Tramite test con diversi parametri si sono individuate le scelte di valori ottimali per raggiungere le migliori prestazioni con l'utilizzo della metrica α-closeness.

Infine, per accelerare i tempi di esecuzione, sono state cercate delle stopping condition per far terminare prima gli algoritmi. Si è quindi verificato come le stopping condition proposte possano efficacemente ridurre la durata della computazione e come al variare di parametri legati all'α-closeness esse possano diventare più efficaci, permettendo un trade-off tra velocità di esecuzione e precisione che si desidera raggiunge. In particolare, se il k in esame non è particolarmente alto, si possono ottenere ottimi risultati con tempi molto ridotti.

1.3 Struttura della tesi

La tesi procede nei seguenti capitoli in questo modo

- Nel Capitolo 2 viene presentato il framework teorico relativo agli uncertain graph, oltre ad alcuni dei risultati teorici presenti nella letteratura non direttamente collegati alla tesi che si sono comunque supervisionati nella fase di ricerca e approfondimento.
- Nel Capitolo 3 vengono descritte le metriche Median-Distance e α-closeness utilizzate per indirizzare il problema dei k-NN, spiegando inoltre i punti di debolezza della Median-Distance che hanno portato in questo lavoro a cercare di dimostrare come l'α-closeness possa rivelarsi più adatta per il tipo di query in esame.
- Nel Capitolo 4 viene affrontato specificatamente il problema delle query k-nearest neighbor, proponendo un algoritmo di risoluzione approssimato basato sulla α-closeness, oltre a delle stopping condition per renderlo più efficiente.
- Infine, nel Capitolo 5 vengono mostrati i risultati degli esperimenti eseguiti in varie situazioni e con diversi parametri mirati a testare sia le prestazioni in termini di precision, recall, tp, tn ecc., sia in termini di tempi di esecuzione.
- Nel Capitolo 6 vengono riassunti i risultati trovati e i possibili sviluppi futuri.

Capitolo 2

Uncertain Graph

Sempre maggiore importanza stanno ricevendo gli uncertain graphs grazie alla loro capacità di modellare delle situazioni in cui vi è dell'incertezza nelle relazioni tra gli elementi di un insieme [11]. In questo capitolo verrà descritto il modello teorico ed alcune delle applicazioni in cui tali grafi sono stati utilizzati in alcuni recenti studi.

2.1 Definizione

Si definisce uncertain graph un grafo $\mathcal{G} = (V, E, p, w)$ dove V è l'insieme dei vertici, con $|V| = n$; E l'insieme dei lati, con $|E| = m$; w è la funzione dei pesi sui lati, quindi $w : E \to \mathbb{R}^+$, con $w(e)$ che denota tale peso. La funzione $p : E \to [0, 1]$ assegna invece una probabilità di esistenza ad ogni lato. Gli eventi di esistenza dei lati sono considerati indipendenti. Si prendono in esame solo grafi non diretti e, per semplicità di trattazione, il caso particolare di grafi non pesati, in cui cioè $w(e) = 1 \ \forall e \in E$.

Si fa notare che un modo alternativo per definire gli uncertain graph è quello di considerare un grafo completo, nel quale i lati hanno una distribuzione di probabilità e, in particolare, i lati mancanti hanno probabilità di esistenza pari a 0.

Viene qui adottata la cosiddetta *possible world semantics*, descritta in [11], a sua volta derivata dalla letteratura sui database probabilistici [5], nella quale si assume indipendenza tra i lati del grafo. Un grafo probabilistico $\mathcal{G}$ ha ovviamente 2^m possibili realizzazioni, determinate dalla presenza ed assenza dei lati. In ogni realizzazione come grafo deterministico infatti si sceglie quali lati appaiono e quali no, portando quindi a 2^m possibilità. Ognuna di questa realizzazioni viene chiamata *possible world* del grafo $\mathcal{G}$, e si indica con $G \sqsubseteq \mathcal{G}$. Ognuno di questi, a seconda di quali lati compaiono, ha come probabilità di venire estratto:

$$Pr[G] = \prod_{e \in E_G} p(e) \prod_{e \in E \setminus E_G} (1 - p(e)) \tag{2.1}$$

dove E_G indica l'insieme dei lati nel grafo G. Si noti che l'insieme dei vertici è lo stesso per ogni *possible world*.

Inoltre, dato $G \sqsubseteq \mathcal{G}$, sia $d_G(s,t)$ la distanza del cammino minimo tra i nodi s e t. Definiamo, come in [11], la **distribuzione** $p_{s,t}$ della distanza del cammino minimo tra i due vertici

$$p_{s,t}(d) = \sum_{G | d_G(s,t)=d} Pr[G] \tag{2.2}$$

Ossia, $p_{s,t}(d)$ è la somma delle probabilità di tutti i *possible world* in cui la distanza minima tra s e t è esattamente d. Ovviamente potrebbero esserci dei casi in cui i due nodi risultano disconnessi e quindi la distanza infinita. In questa situazione si permette a d di assumere valore ∞, e conseguentemente $p_{s,t}(\infty)$ è definita come la somma delle probabilità di esistenza di tutti i *possible world* in cui s e t sono disconnessi.

Ora, ci sono diversi modi in cui si possono definire metriche che possano in qualche modo sintetizzare la distribuzione di probabilità delle distanze tra i vertici all'interno di un grafo probabilistico. In [11], gli autori definiscono diverse distanze tra cui la Median-Distance, la Majority-Distance e la Expected-Reliable-Distance. In questa nota si concentra l'attenzione sulla Median-Distance, mettendola in parallelo con una nuova misura di vicinanza, detta *α-closeness*. Prima di discutere le metriche però, si fa notare la relazione che intercorre tra quest'ultime e la semantica dell'applicazione in esame. Se si considera ad esempio la Figura 2.1, dove le probabilità sugli unici due lati differiscono di poco, ci si può chiedere quale sia il nodo più vicino ad u, se $c1$ o $c2$. Ora, potrebbe influire su questa scelta ciò che il grafo sta a rappresentare. Infatti, se fosse una rete di sensori, dove la probabilità sui lati misura la reliability del collegamento, mentre il peso misura la sua latenza, allora potrebbero esserci applicazioni, richiedenti la maggiore affidabilità possibile, che potrebbero preferire $c1$, nonostante la maggiore latenza.

Questo è solo un semplice esempio per dimostrare come non possa esistere una metrica di distanza adatta in ogni caso: ogni contesto potrebbe richiedere

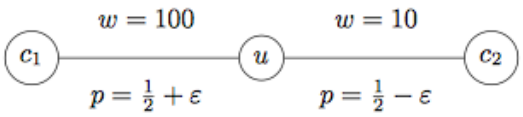


Figura 2.1: *Un uncertain graphs con probabilità lievemente differenti ai lati*

la sua metrica specifica. In particolare, in [11], gli autori scelgono di trattare in maggior dettaglio la Median-Distance, ma questo senza specificare perché, ossia senza descrivere la condizione in cui ritengono che questa metrica possa essere più utile rispetto ad un'altra.

Qui si ha il focus sia sulla distanza intesa in senso classico, cioè come numero minimo di archi da attraversare per raggiungere il nodo t dal nodo s, che sulla reliability delle connessioni tra i nodi del cammino. Si metteranno a confronto la Median-Distance con la α-closeness per scoprire come queste due possano permettere di rispondere alla query dei k-nearest neighbors in un uncertain graph.

2.2 Alcune applicazioni su Uncertain Graphs

Come spiegato nell'introduzione al Capitolo 1, gli Uncertain Graphs sono un paradigma di rappresentazione di dati piuttosto espressivo, utilizzati per descrivere entità e loro relazioni in una vasta gamma di applicazioni.

La presenza di incertezza in una relazione può essere dovuta a misurazioni soggette a rumore; l'utilizzo di modelli di predizione come nel Data Mining; o ancora per una esplicita manipolazione per proteggere la privacy degli utenti.

Si possono individuare tre principali direzioni di ricerca per quel che riguarda gli Uncertain Graphs [10]:

1. Query basate sulle distanze dei cammini minimi (k-NN, distance constant reachability computation).

2. Pattern mining (top-k maximal cliques, frequent subgraph pattern).

3. Subgraph (similarity) search.

Come osservato nella Sezione 2.1, servirebbe materializzare $2^{|E|}$ possible world per poter affrontare direttamente molti problemi, la qual cosa è impossibile. Conseguentemente, si è soliti applicare dei metodi Monte-Carlo. Tuttavia, spesso, nemmeno questa strada è spesso facilmente praticabile, dato che [10]:

1. Il sampling di un possible world spesso ha un costo non trascurabile.

2. In combinazione col primo punto, spesso è richiesto un grande numero di sample.

3. Non è sempre immediatamente chiaro come aggregare l'informazione che perviene dai diversi sample.

Ogni problema all'interno degli Uncertain Graph necessita di essere studiato e analizzato a parte, sviluppando metodi di soluzione specifici. Nel resto del paragrafo, verranno velocemente presentati quattro problemi descritti in altrettanti paper. Con riferimento alle direzioni di ricerca sopra citate, il primo, il secondo e il quarto problema riguardano il pattern mining, mentre il terzo è attinente alle query basate sulle distanze dei cammini minimi.

2.2.1 Core Decomposition of Uncertain Graphs

In [3] gli autori si concentrano sul problema della *core decomposition* applicata agli Uncertain Graphs. In generale, un *k-core* di un grafo è definito come il sottografo massimale nel quale ogni vertice è collegato ad almeno altri k vertici nello stesso sottografo. In maniera più formale:

Definizione. Il *k-core* (o core di ordine k) di un grafo deterministico G è un sottografo massimale $H = (C, E|C)$ (dove C è un sottoinsieme dei nodi di G, mentre $E|C$ è l'insieme dei lati indotto da C) tale che $\forall v \in C : deg_H(v) \geq k$. Il *core number* di un vertice v, indicato come $c(v)$, è l'ordine del core che contiene v con ordine maggiore. L'insieme di tutti i k-core di G, per ogni k, è detta **core decomposition** di G.

Si può osservare che la core decomposition di un grafo G è unica e determinata dal core number $c(v)$ di tutti i vertici v in G. L'algoritmo per calcolare la core decomposition di un un grafo G impiega tempo lineare.

Più difficile diventa la definizione in un grafo probabilistico.

Definizione. Data un uncertain graph $\mathcal{G} = (V, E, p)$ e una soglia $\eta \in [0, 1]$, si chiama **probabilistic** (k, η)**-core** di $\mathcal{G}$ il sottografo massimale $\mathcal{H} = (C, E|C, p)$ tale che $\forall v \in C$

$$Pr[deg_{\mathcal{H}}(v) \geq k] \geq \eta$$

Data questa definizione, il problema in cui ci si concentra in [3] è chiamato PROBCORES, e consiste, dato un uncertain graph $\mathcal{G}$ ed una soglia $\eta \in [0, 1]$, nel trovare la (k, η)-core decomposition di $\mathcal{G}$. Il paper quindi procede nel creare un algoritmo per trovare la core decomposition nel caso probabilistico basato su dynamic-programming.

Un tale problema, secondo [3], è stato utilizzato e quindi può tornare utile per analizzare la natura di un network e scoprirne sottostrutture dense. È stato applicato in campi come la bioinformatica, la software engineering e le reti sociali. È stato inoltre impiegato per portare uno speed-up nel calcolo di sottografi densi con definizioni più complicate, tra le quali ad esempio le clique massimali.

2.2.2 Clustering Large Probabilistic Graphs

In [8], gli autori presentano un approccio per il probabilistic graph clustering. Si parte considerando un grafo deterministico $G = (V, E)$, e una partizione C dei nodi in V. Una *clustering objective function* $D(G, C)$ è una funzione che quantifica il costo del clustering C rispetto a G. Nel caso di un grafo probabilistico $\mathcal{G}$, si ha che $D(\mathcal{G}, C) = E[D(G, C)]$, ossia il costo del clustering C sull'uncertain graph è il valore atteso di $D(G, C)$ su tutti i possible world G. Ovviamente, questa definizione è problematica in quanto richiederebbe la creazione di tutti i possible world a disposizione.

Gli autori nel paper utilizzano come funzione la *edit distance*, definita come segue:

Definizione. Dati due grafi deterministici $G = (V, E_G)$ e $Q = (V, E_Q)$, si definisce la *edit distance* tra G e Q come il numero di lati da aggiungere o eliminare da G per trasformarlo in Q.

$$D(G, Q) = |E_G \backslash E_Q| + |E_Q \backslash E_G|$$

Nel caso di grafi probabilistici invece, si utilizza l'*expected edit distance* tra Q ed ogni $G \sqsubseteq \mathcal{G}$

$$D(\mathcal{G}, Q) = \sum_{G \sqsubseteq \mathcal{G}} Pr[G] D(G, Q)$$

Proseguono definendo quindi il tipo di grafo di clustering a cui si ispirano:

Definizione. Un cluster graph $C = (V, E_C)$ è un grafo deterministico con le seguenti proprietà:

1. C definisce una partizione dei nodi di V in k parti, $V = \{V_1, \ldots, V_k\}$ $|\forall i, j, i \neq j \quad V_i \cap V_j = \emptyset$.
2. Per ogni $i \in \{1 \ldots, k\}$ e per ogni coppia di nodi $v, v' \in V_i$ si ha che $\{v, v'\} \in E_C$.
3. Per ogni $i, j \in \{1, \ldots, k\}$ con $i \neq j$ ed ogni coppia di nodi $v \in V_i, v' \in V_j$, si ha $\{v, v'\} \notin E_C$.

Il problema chiamato PCLUSTEREDIT consiste, dato un probabilistic graph $\mathcal{G} = (V, E, p)$, nel trovare il cluster graph $C = (V, E_C)$ tale che $D(\mathcal{G}, C)$ sia minimizzata. Questa non è che una generalizzazione del problema CLUSTEREDIT con grafi deterministici. Essendo quest'ultimo NP-Hard, anche PCLUSTEREDIT è NP-Hard.

Gli autori di questo paper non fanno altro che adattare altri algoritmi pensati per un altro problema dalla objective function simile a quella del PCLUSTEREDIT già esistenza, mostrandone la complessità e l'efficienza.

2.2.3 Distance-Constraint reachability Computation in Uncertain Graphs

In generale, il problema noto come *s-t reachability* nei grafi probabilistici calcola la probabilità totale dei possible world $G \sqsubseteq \mathcal{G}$ nei quali il vertice t sia raggiungibile dal vertice s. Quindi, mentre nei grafi deterministici il problema della reachability ha semplicemente come risposta sì o no, per gli uncertain graphs si ha una risposta in termini probabilistici.

In particolare, gli autori di [7] non si concentrano su questo problema, ma su di una sua generalizzazione, chiamata *distance-constraint reachablity (DCR)*, per la quale, dati due vertici s e t, ci si chiede quale sia la probabilità che la distanza tra i due sia minore o uguale ad una soglia fissata d. Non solo quindi ci si chiede la probabilità con cui i due sono connessi, ma che lo siano entro una certa distanza.

In particolare, il problema della s-t reachability è noto per essere #P-Complete[1] negli uncertain graph. Essendo DCR una sua estensione, sarà #P-Complete a sua volta. Ne consegue che in [7] gli autori si concentrano nello sviluppare un approccio che approssimi questo problema. In particolare, se definiamo

$$\mathbf{I}^{\mathbf{d}}_{\mathbf{s,t}}(\mathbf{G}) = \begin{cases} 1, & \text{se} \quad dis(s,t|G) \leq d \\ 0, & \text{altrimenti} \end{cases}$$

la variabile aleatoria Bernoulliana che indica se nel possible world $G \sqsubseteq \mathcal{G}$ la distanza tra s e t è minore o uguale a d, allora la s-t distance-constraint reachability nell'uncertain graph $\mathcal{G}$ rispetto al parametro d è definibile come

$$\mathbf{R}^{\mathbf{d}}_{\mathbf{s,t}}(\mathcal{G}) = \sum_{\mathbf{G} \sqsubseteq \mathcal{G}} \mathbf{I}^{\mathbf{d}}_{\mathbf{s,t}}(\mathbf{G}) \cdot \mathbf{Pr}[\mathbf{G}]$$

Calcolare direttamente $\mathbf{R}^{\mathbf{d}}_{\mathbf{s,t}}$ è #P-Complete, in quanto richiederebbe di esplorare tutti i possible world G dell'uncertain graph. Conseguentemente, in [7] si sviluppa un modo per calcolare un unbiased estimator $\hat{R}$ che abbia la varianza e il costo computazionale più piccoli possibile. Per far questo vengono proposte diverse soluzioni, tra cui un approccio diretto tramite sampling e un più complesso metodo che si basa sullo sviluppare un estimator partendo dall'albero di ricorsione generato da un algoritmo Divide & Conquer esatto.

[1] Secondo la teoria della complessità computazionale, un problema in NP è tipicamente espresso sottoforma di una domanda a cui si può rispondere sì/no. Per quel che riguarda i problemi #P, essi sono posti nella forma "quante istanze sono presenti?" e sono associati con i corrispettivi problemi in NP. I problemi #P-Complete costituiscono una classe di complessità nella teoria della complessità computazionale. Secondo definizione, un problema è #P-Complete se e solo se si trova in #P e se per ogni macchina di Turing non deterministica si può ridurre ad esso il problema di calcolare il numero dei path che essa accetta.

2.2.4 The pursuit of a Good Possible World

Come accennato precedentemente, spesso l'approccio ai problemi riguardanti gli Uncertain Graph tramite metodo Monte-Carlo risulta comunque molto pesante computazionalmente. L'idea in [10] è quella di individuare, tra tutti i possible world di $\mathcal{G}$, un grafo deterministico tale da mantenere in sé alcune caratteristiche particolari. Questo quindi potrà fungere da *rappresentante* e permettere di eseguire su di lui i classici algoritmi per grafi deterministici per risolvere problemi computazionali che richiederebbero altrimenti un elevato tempo di calcolo. Fissato il problema che si intende affrontare, gli algoritmi di estrazione di questo good possible world possono avere caratteristiche differenti. In particolare, per poter raggiungere una certa accuratezza nel risultato, il rappresentante dovrà preservare la struttura sottostante dell'uncertain graph. In [10] si osserva che il grado dei vertici è una delle proprietà fondamentali della struttura di un grafo, e viene quindi fatta la congettura per cui preservando il grado di ogni vertice si potrà catturare l'essenza dell'uncertain graph sottostante, e quindi così meglio approssimarne altre proprietà.

Pertanto, dato un uncertain graph $\mathcal{G}$ ed un vertice $u \in V$, viene chiamato *expected degree* di u in $\mathcal{G}$ il valore

$$[deg_u(\mathcal{G})] = \sum_{e=(u,v)\in E} p(e)$$

e la *discrepancy* di un vertice u in un possible world $G \sqsubseteq \mathcal{G}$ è definita come

$$dis_u(G) = deg_u(G) - [deg_u(\mathcal{G})]$$

A questo punto, si può definire la *discrepancy* di un possible world G come

$$\Delta(G) = \sum_{u\in V} |dis_u(G)|$$

Il problema, detto della REPRESENTATIVE INSTANCE, consiste dato l'uncertain graph $\mathcal{G} = (V, E, p)$ nel trovare il possible world $G^* \sqsubseteq \mathcal{G}$ tale che

$$G^* = \arg\min_{G\sqsubseteq\mathcal{G}} \Delta(G)$$

Nel paper sono proposti due algoritmi per risolvere REPRESENTATIVE INSTANCE. Curiosamente non si è ancora riusciti ad identificare la sua classe di complessità, sebbene la congettura fatta è che si tratti di un problema NP-Hard date le sue affinità con un problema relativamente simile detto *closest vector problem*.

Capitolo 3

Metriche di distanza

Un grafo probabilistico, come si è visto, non è che una generalizzazione di un grafo deterministico, in quanto possiede una distribuzione di probabilità sui suoi lati. Questo fa sì che in una sua possibile realizzazione, altrimenti detta *possible world*, un lato con probabilità non nulla possa non esistere. Questo rende più complessa la definizione di distanza tra due vertici.

È importante notare subito come possano essere definite diverse metriche di distanza a questo punto. In [11] gli autori ne citano tre e ne definiscono altre tre, per esempio. Non esiste una metrica universale che possa applicarsi in maniera ideale in ogni situazione. Ogni metrica pone importanza ad un aspetto del grafo piuttosto che ad altri, ed è quindi necessario scegliere la metrica più corretta in base al task che si è intenzionati ad affrontare.

In questo capitolo è presentata e ridefinita la Median-Distance utilizzata in [11] e si spiega come essa possa essere utilizzata nel problema dei k-NN in un grafo probabilistico. Successivamente,a partire da alcune critiche mosse alla Median, è definita la nuova metrica α-closeness e viene anche dato un suo approssimatore unbiased.

3.1 La Median-Distance

Definizione 1. *Dato un grafo probablistico $\mathcal{G} = (V, E, p, w)$ e i nodi $s, t \in V$, definiamo come **Median-Distance**, e indichiamo con $d_M(s,t)$, la mediana della distribuzione di probabilità sulle distanze tra s e t originata dalla distribuzione di probabilità di $\mathcal{G}$. Ossia:*

$$p(X \leq d_M(s,t)) \geq \frac{1}{2} \wedge p(X \geq d_M(s,t)) \geq \frac{1}{2}$$

Se vi è più di una mediana, si prende la più piccola.

Si noti peraltro come la definizione data qui è diversa dall'originale data in [11], che è la seguente:

Definizione. Dato un grafo probablistico $\mathcal{G} = (V, E, p, w)$ e i nodi $s, t \in V$, definiamo come **Median-Distance**, e indichiamo con $d_M(s,t)$, la distanza mediana del *shortest-path* tra tutti i *possible world*, cioè:

$$d_M(s,t) = \arg\max_D \left\{ \sum_{d=0}^{D} p_{s,t}(d) \leq \frac{1}{2} \right\}$$

Questa è una definizione mal posta, in quanto, se si considera la seguente distribuzione di distanza sull'arco $\{s,t\}$:

$$p_{s,t}(1) = \epsilon \qquad p_{s,t}(\infty) = 1 - \epsilon$$

ed applicando la definizione data in [11], ne risulta che la Median-Distance dovrebbe essere 1, mentre chiaramente dovrebbe essere ∞.

3.1.1 Calcolo della Median-Distance

Il modo più diretto per il calcolo della Median-Distance tra una coppia di nodi s e t è quello di individuare la loro distanza deterministica $d(s,t)$ in ogni $G \sqsubseteq \mathcal{G}$, da queste derivare la distribuzione di probabilità della distanza e quindi calcolare la mediana. Questo richiederebbe la creazione dei 2^m *possible world* di $\mathcal{G}$, su ognuno dei quali servirebbe utilizzare un algoritmo per il calcolo della distanza minima tra i due vertici e ciò è ovviamente inaccettabile come complessità.

Il seguente Lemma rappresenta uno strumento per la stima della Median-Distance tramite l'estrazione di sample.

Lemma 1. *Sia D una variabile aleatoria con PDF (Probability Distribution Function) $f_D(x)$ e mediana μ. Siano α e β i valori per cui $\sum_{x=-\infty}^{\alpha} f_D(x) = 1/2 - \varepsilon$ e $\sum_{x=\beta}^{\infty} f_D(x) = 1/2 + \varepsilon$. Si consideri un certo insieme di sample $\mathcal{S} = \{s_1, s_2, \ldots, s_r\}$, dove $r \geq \frac{c}{\varepsilon^2} log\frac{2}{\delta}$, estratti da D, e sia $\hat{\mu} = median(s_1, \ldots, s_r)$. Per un c adeguato abbiamo che:*

$$Pr[\hat{\mu} \in [\alpha, \beta]] > 1 - \delta$$

Dimostrazione. Partizioniamo l'insieme $\mathcal{S}$ in tre sottoinsiemi:

$$\mathcal{S}_L = \{s : s \in \mathcal{S} \wedge s < \alpha\}$$
$$\mathcal{S}_M = \{s : s \in \mathcal{S} \wedge s \in [\alpha, \beta]\}$$
$$\mathcal{S}_U = \{s : s \in \mathcal{S} \wedge s > \beta\}$$

Si mostra qui che, con probabilità almeno $1-\delta$, l'elemento mediano di $\mathcal{S}$ si trova in $\mathcal{S}_M$.

Si considerino le variabili aleatorie X_i, per $i \in [1, r]$, tali che $X_i = 1$ se $s_i \in \mathcal{S}_L$, e $X_i = 0$ altrimenti. Sia ora $X = \sum_{i=1}^{r} X_i$ la variabile aleatoria che conta il numero di elementi in $\mathcal{S}_L$. Abbiamo $E[X] = (1/2 - \varepsilon)r$. Utilizzando quindi il bound di Chernoff, possiamo dire che:

$$Pr[X \geq (1+\varepsilon)E[X]] \leq e^{\frac{-\varepsilon^2(1/2-\varepsilon)r}{3}} \leq \delta/2$$

Dove la seconda disuguaglianza segue dalla scelta fatta di r. Si osserva ora che:

$$(1+\varepsilon)E[X] = \left(\frac{1}{2} - \frac{\varepsilon}{2} - \varepsilon^2\right) \cdot r < \frac{r}{2}$$

Da questo possiamo dire che $X < r/2$ con probabilità almeno $1 - \delta/2$, il che significa che meno di $r/2$ elementi sono in $\mathcal{S}_L$ con questa stessa probabilità.

In maniera del tutto simile si può ragionare per $\mathcal{S}_U$, dimostrando che con probabilità almeno $1 - \delta/2$ in esso si trova un numero di elementi minore di $r/2$.

Applicando a questo punto lo union bound a questi due ultimi eventi, abbiamo che, con probabilità almeno $1 - \delta$, ci sono meno di $r/2$ elementi in $\mathcal{S}_L$ e lo stesso per $\mathcal{S}_U$.

Si assuma ora per assurdo che $\hat{\mu} \in \mathcal{S}_L$. Questo comporta che $\hat{\mu} < \alpha$. Ora, dato che $\hat{\mu}$ è la mediana di $\mathcal{S}$ per ipotesi, c'è un numero di elementi $\geq r/2$ che sono $\leq \hat{\mu}$. Ognuno di questi elementi, per costruzione, deve stare in $\mathcal{S}_L$, e quindi ne segue che $|\mathcal{S}_L| \geq r/2$. Questa è però una contraddizione con quanto detto prima, ossia che con probabilità almeno $1 - \delta$ $|\mathcal{S}_L| < r/2$. Ne segue che $\hat{\mu} \notin \mathcal{S}_L$. Ripetendo un ragionamento del tutto analogo, si dimostra che $\hat{\mu} \notin \mathcal{S}_U$, e che quindi, dato che i tre insiemi formano una partizione di $\mathcal{S}$ per costruzione, si ha che $\hat{\mu} \in \mathcal{S}_M$, ossia che $\hat{\mu} \in [\alpha, \beta]$, con probabilità almeno $1 - \delta$. □

Ora, questo lemma deve essere utilizzato con la giusta consapevolezza, in quanto esso garantisce che si riesce a raggiungere un errore relativo di ε sulle probabilità attorno alla mediana, non sulle distanze ottenute come stime. Ossia, si riesce ad ottenere un valore la cui probabilità di realizzazione si assesta molto vicino a quella della mediana, ma non è detto che esso, in sé e per sé, sia un valore vicino a quello effettivo della mediana. In particolare, se la Cumulative Distribution Function delle distanze presenta uno scalino attorno alla probabilità 1/2, possiamo ottenere delle stime della Median-Distance anche molto diverse, a seconda di quali sample vengono presi di volta in volta (figura 3.1).

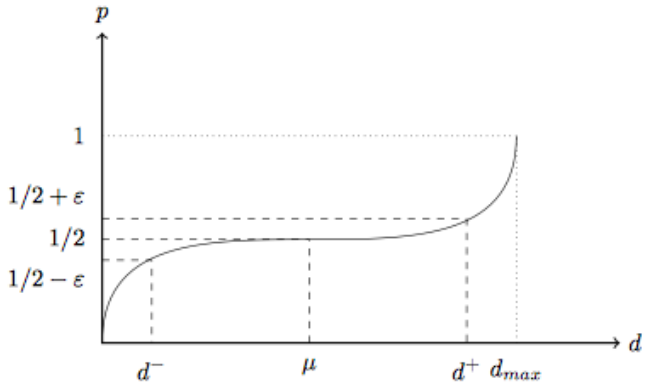


Figura 3.1: *Un particolare esempio di CDF dove la mediana che viene estrapolata dal calcolo dei sample potrebbe portare ad un valore anche molto distante dalla mediana originale, pur avendo le due una probabilità di uscire molto simile.*

3.2 La metrica α-closeness

3.2.1 Le ragioni per una nuova metrica

A questo punto, possiamo identificare due problemi per quel che riguarda la misura della Median-Distance. Il primo difetto è stato descritto alla fine della sezione 3.1, e deriva dal fatto che il sampling della Median-Distance non porta nessuna garanzia sull'errore commesso sulla stima della distanza stessa.

Il secondo problema riguarda il fatto che la Median-Distance è, di fatto, estremamente sensibile alle piccole variazioni sulle probabilità dei lati. Questo può portare a risposte molto diverse alla stessa query k-NN se solo si cambia leggermente le probabilità su qualche edge. Nell'esemio di figura 3.2 si può vedere come questo sia vero. In questo particolare grafo, si ha la source s nel centro, circondata da una stella di k nodi, gli a_i, ed una catena di altri k vertici, i b_i. Su questo grafo, vogliamo calcolare i k-nearest neighbors di s. Se si considera il caso in cui i lati della stella abbiano probabilità $\frac{1}{2} + \varepsilon$ si può facilmente vedere come i vicini di s siano i nodi della stella, ossia $b_1, a_1, \ldots, a_k$. Questi vertici infatti si trovano tutti alla distanza mediana 1.

Viceversa, se si variano leggermente le probabilità sui lati della stella, ponendole a $\frac{1}{2} - \varepsilon$, la loro distanza mediana da s diventa ∞, e in questo modo i k vicini sono i nodi della catena, $b_1, \ldots, b_k$.

Questo semplice esempio mostra che se usiamo la Median-Distance per il ranking dei vicini, una perturbazione arbitrariamente piccola delle probabilità sui lati del grafo potrebbe portare ad un risultato estremamente diverso alla stessa query

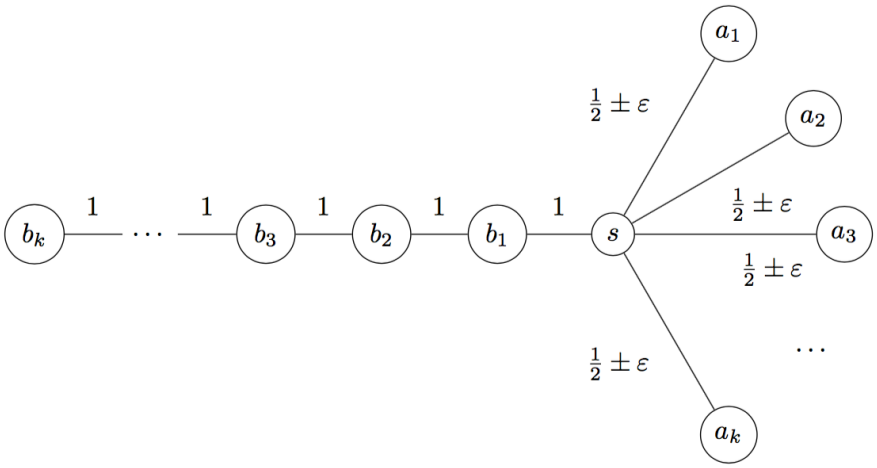


Figura 3.2: *L'algoritmo k-NN basato sulla Median-Distance dimostra in questo esempio la sua estrema sensibilità alle piccole variazioni sulla probabilità dei lati. Questa figura mostra un uncertain graph con lati non diretti e non pesati, etichettati con la loro probabilità, ed* $\varepsilon > 0$.

k-NN. Da questo deriva il bisogno di una misura più robusta per il ranking dei nodi in base alla loro distanza da s.

3.2.2 α-closeness

Andiamo a considerare quindi la seguente misura:

Definizione 2. *Dato un grafo probabilistico* $\mathcal{G} = (V, E, p, w)$ *e una coppia di nodi* $s, t \in V$*, e per un* $\alpha > 0$*, si definisce la loro* ***expected*** α***-closeness*** *come:*

$$c_\alpha(s,t) \triangleq \sum_{d=1}^{\infty} \frac{1}{d^\alpha} p_{s,t}(d)$$

e diciamo che $\frac{1}{\infty} = 0$ *(quando un nodo non è raggiungibile).*

Questa metrica cerca di catturare sia le distanze che le loro probabilità. In particolare, la metrica favorisce i nodi che sono vicini con una maggiore probabilità. Da ciò, i nodi più vicini alla sorgente s possono essere definiti come quei $t \in V$ che presentano la più alta α-closeness.

Il parametro α può essere usato per controllare l'influenza delle distanze e delle probabilità sullo score. Questo perché un α più grande favorirà le distanze rispetto alle probabilità (un nodo distante, anche se avrà una probabilità alta su quella distanza, vedrà quel contributo notevolmente diminuito. Un α alto favorisce i nodi

più vicini, abbattendo velocemente gli score dei nodi più distanti) e, viceversa, un minore α ridurrà l'influenza delle distanze nello score finale.

Rispetto alla Median-Distance, l'expected α-closeness è più robusta alle piccole perturbazioni sulle probabilità dei lati. Se si riconsidera l'esempio della figura 3.2, si vede, se si fissa $\alpha = 1$, che quando i lati della stella uscente da s hanno probabilità pari a $1/2 + \varepsilon$ si ha che $c_\alpha(s, a_i) = 1/2 + \varepsilon$ per tutti gli a_i della stella. Per quel che riguarda i nodi nella catena, si ha invece:

$$c_\alpha(s, b_1) = 1 \quad c_\alpha(s, b_2) = 1/2 \quad \dots \quad c_\alpha(s, b_k) = 1/k$$

Ne consegue che i k-nearest neighbours saranno $b_1, a_1, \dots, a_k$.

Se invece i lati della stella avessero probabilità $1/2-\varepsilon$, si avrebbe che $c_\alpha(s, a_i) = 1/2-\varepsilon$ per $i \in [1, k]$. In questo modo i k-nearest neighbours diventano $b_1, b_2, a_1, \dots, a_k$. Quindi, mentre l'$\alpha$-closeness è robusta a queste variazioni nelle probabilità dei lati, la Median-Distance non lo è.

Osservazione 1. *La expected α-closeness è pari al valore medio della variabile aleatoria $(1/d_{\mathcal{G}}(s,t))^\alpha$, la quale prende valori $\in [0, 1]$. Ne segue che per ogni coppia di nodi s e t si ha che $0 \leq c_\alpha(s, t) \leq 1$.*

3.2.3 Approssimare l'α-closeness

Ora, è necessario capire come calcolare l'expected α-closeness. Anche in questo caso, il metodo banale e non praticabile richiederebbe il calcolo di tutti i 2^m *possible world* da cui ottenere le distanze e le probabilità con cui esse compaiono per ognuno dei vertici. Si può tuttavia creare un unbiased estimator eseguendo il sampling di alcune istanze di *possible world* di $\mathcal{G}$, come mostrato nel Lemma 4.

Lemma 2. *[4] Sia X una variabile aleatoria discreta con PDF data da f_X, e sia $g(X)$ una funzione nella variabile aleatoria X. Il valore atteso di $g(X)$ è dato da:*

$$E[g(X)] = \sum_x g(x) f_X(x)$$

dove la somma è eseguita su tutti i valori x assumibili da X.

Lemma 3. ***(di Hoeffding)*** *Se $x_1, \dots, x_k$ sono delle variabili aleatorie indipendenti, con $a_i \leq x_i \leq b_i$, e $\mu = E[\sum_{i=1}^k x_i/k]$ è il valore atteso della media ottenuta dai sample, allora, per $\varepsilon > 0$, si ha:*

$$Pr\left[\left|\frac{1}{r}\sum_{i=1}^{r} x_i - \mu\right| \geq \varepsilon\right] \leq 2exp\left(\frac{-2r^2\varepsilon^2}{\sum_{i=1}^{r}(b_i - a_i)^2}\right)$$

Lemma 4. *Sia $\mathcal{G} = (V, E, w, p)$ un uncertain graph ed r un certo numero di possible world da esso estratti. inoltre, per $i \in [1, r]$, siano $G_i \sqsubseteq \mathcal{G}$ suddetti sample. Allora*

$$\hat{c}_\alpha(s,t) = \frac{1}{r}\sum_{i=1}^{r}\frac{1}{(d_{G_i}(s,t))^\alpha}$$

è un unbiased estimator di $c_\alpha(s,t)$

Dimostrazione. La distanza $d_{G_i}(s,t)$ in ogni sample $G_i \sqsubseteq \mathcal{G}$ è una variabile aleatoria la cui PDF è la $p_{s,t}(d)$ come data nell'equazione 2.2. Grazie al Lemma 2 abbiamo che, considerando la funzione $1/(d_{G_i}(s,t))^\alpha$, possiamo calcolare il suo valore atteso in questo modo:

$$E\left[\frac{1}{(d_{G_i}(s,t))^\alpha}\right] = \sum_{d=1}^{\infty}\frac{1}{d^\alpha}p_{s,t}(d)$$

A questo punto, possiamo calcolare la media dell'estimator come segue:

$$\begin{aligned} E[\hat{c}_\alpha(s,t)] &= E\left[\frac{1}{r}\sum_{i=1}^{r}\frac{1}{(d_{G_i}(s,t))^\alpha}\right] = \frac{1}{r}\sum_{i=1}^{r}E\left[\frac{1}{(d_{G_i}(s,t))^\alpha}\right] \\ &= \frac{1}{r}\sum_{i=1}^{r}\sum_{d=1}^{\infty}\frac{1}{d^\alpha}p_{s,t}(d) = \frac{1}{r}\sum_{i=1}^{r}c_\alpha(s,t) \\ &= c_\alpha(s,t) \end{aligned}$$

□

Capitolo 4

Il problema dei k-NN negli Uncertain Graphs

Si è già descritto nell'introduzione del capitolo 1 come il problema dei k-nearest neighbors sia di grande interesse e di utilità nei grafi. Mentre nei grafi deterministici calcolare i primi k vicini di un nodo è compito relativamente facile, facilmente effettuabile tramite algoritmi di attraversamento come BFS nel caso di grafi non pesati o Dijkstra altrimenti, nella situazione posta dai grafi probabilistici il problema è molto più complesso.

In questo capitolo si approccia il problema utilizzando la metrica di distanza α-closeness, è sviluppato un algoritmo di approssimazione per il suo calcolo e delle stopping condition per velocizzare i calcoli.

4.1 Il calcolo dei k-NN in base alla expetced α-closeness

In questa sezione si studierà come l'α-closeness possa essere applicata nel problema dei k-NN.

Definizione 3. *Dato l'uncertain graph $\mathcal{G} = (V, E, w, p)$ e un nodo sorgente $s \in V$, si consideri un ordinamento decrescente dei vertici $v \in V \backslash \{s\}$ secondo il loro $c_\alpha(s, v)$ e si mettano come nomi per questi vertici $v_1, \ldots, v_{n-1}$, in accordo a questo ordinamento. Scelto un k, con $1 \leq k \leq n$, si definisce per brevità $c_\alpha^{(k)} = c_\alpha(s, v_k)$. Viene chiamato* ***insieme dei k-nearest neighbors*** *rispetto alla sorgente s e in accordo alla misura della expected α-closeness il seguente insieme:*

$$k\text{-}NN(\mathcal{G}, V, s, k) \triangleq \{(v, c_\alpha(s, v)) : v \in V \backslash \{s\}, c_\alpha(s, v) \geq c_\alpha^{(k)}\}$$

La cardinalità di questo insieme è $\geq k$.

Dato che calcolare l'α-closeness per tutti i vertici richiederebbe un tempo esponenziale in m, si preferisce usare un algoritmo di approssimazione. Definiamo pertanto la seguente soluzione approssimata per l'insieme dei k-nearest neighbors:

Definizione 4. *Dato l'uncertain graph $\mathcal{G} = (V, E, w, p)$, il nodo $s \in V$, $k \in [1, n]$ intero e $\varepsilon \in (0, 1)$ reale, si definisce una* ***ε-approximation di k-NN***$(\mathcal{G}, V, s, k)$ *un insieme W di k o più coppie (v, d) tali che $v \in V$, $d \in [0, 1]$ e per il quale valga ognuna delle seguenti:*

P1 $\forall (v, d) \in W, c_\alpha(s, v) \geq c_\alpha^{(k)} - \varepsilon$

P2 $\forall (v, d) \notin W, c_\alpha(s, v) < c_\alpha^{(k)} + \varepsilon$

P3 $\forall (v, d) \in W, |d - c_\alpha(s, v)| < \varepsilon$

Prima di affrontare la ricerca di un algoritmo per trovare tale approssimazione, nel seguente lemma viene dato un modo per imporre un bound all'errore assoluto nella stima dell'expected α-closeness per tutti i nodi del grafo rispetto alla singola sorgente s. In particolare, $\hat{c}_\alpha(s, v)$ è usato come unbiased estimator per il valore $c_\alpha(s, v)$ come nel Lemma 4. L'errore assoluto viene resto piccolo quanto $\varepsilon/2$ a causa di come verrà utilizzato nelle dimostrazioni successive.

Lemma 5. *Sia $\mathcal{G} = (V, E, w, p)$ un uncertain graph, e $s \in V$ il nodo sorgente. Per dei fissati $\varepsilon, \delta \in (0, 1)$, sia*

$$r = \frac{2}{\varepsilon^2} ln \frac{2n}{\delta}$$

il numero di sample estrapolati indipendentemente da $\mathcal{G}$ con reinserimento. Allora, con probabilità almeno $1 - \delta$ e $\forall v \in V \backslash \{s\}$ si ha che

$$|c_\alpha(s, v) - \hat{c}_\alpha(s, v)| < \frac{\varepsilon}{2}$$

Dimostrazione. Si consideri un arbitrario $v \in V \backslash \{s\}$. Sia $G_i \sqsubseteq \mathcal{G}$, per $i \in [1, r]$, il *possible world* i-esimo estrapolato da $\mathcal{G}$. Per il Lemma 4, si ricorda che $E[\hat{c}_\alpha(s, v)] = c_\alpha(s, v)$. Si osserva inoltre che le quantità $1/(d_{G_i}(s, v))^\alpha$ sono delle variabili aleatorie indipendenti che assumono valori in $[0, 1]$. Per questo, si stanno soddisfacendo le ipotesi del Lemma 3, e quindi possiamo applicare la disuguaglianza di Hoeffding come segue:

$$Pr\left[|\hat{c}_\alpha(s, t) - c_\alpha(s, t)| \geq \frac{\varepsilon}{2}\right] \leq 2exp\left(\frac{-2r^2(\varepsilon/2)^2}{\sum_{i=1}^{r}(b_i - a_i)^2})\right) = 2exp\left(-\frac{\varepsilon^2}{2}r\right)$$

Se ora sostituiamo al posto di r il numero dato nelle ipotesi si ottiene:

$$Pr\left[|\hat{c}_\alpha(s, t) - c_\alpha(s, t)| \geq \frac{\varepsilon}{2}\right] \leq \frac{\delta}{n}$$

La tesi del lemma segue applicando lo union bound su tutti i nodi $v \in V \backslash \{s\}$ $\square$

4.1.1 Un primo semplice algoritmo di approssimazione

Siamo a questo punto pronti a dare un primo semplice algoritmo di approssimazione per il problema dei k-NN. Si tratta dell'algoritmo 1, dove viene semplicemente fatto per r volte un sampling dei *possible world* e in ognuno di essi è eseguito un attraversamento BFS. Durante ogni attraversamento viene progressivamente calcolato l'approssimatore $\hat{c}_\alpha(s,v)\ \forall v \in V\backslash\{s\}$. Vengono quindi ordinati i nodi in ordine decrescente di $\hat{c}_\alpha$, e vengono date delle label $\hat{v}_1,\ldots,\hat{v}_{n-1}$ a tale ordinamento di nodi. A questo punto, si chiama $\hat{c}_\alpha^{(k)} \triangleq \hat{c}_\alpha(s,\hat{v}_k)$, e si ritorna quindi come risultato dell'algoritmo il set seguente:

$$W = \{(v,\hat{c}_\alpha(s,v)) : v \in V\backslash\{s\}, \hat{c}_\alpha(s,v) \geq \hat{c}_\alpha^{(k)}\}$$

La domanda a questo punto è se questo è realtemente una ε-approximation di k-NN. Nel lemma 6 viene dimostrato che l'algoritmo 1 ritorna effettivamente una ε-approximation di k-NN$(\mathcal{G},V,s,k)$ con probabilità $1-\delta$.

Lemma 6. *Sia $\mathcal{G}$ un uncertain graph. Per un dato $s \in V$ e $k > 0$ e per dei fissati $\varepsilon,\delta \in (0,1)$, l'algoritmo 1 restituisce una ε-approximation di k-NN($\mathcal{G},V,s,k$), con probabilità almeno $1-\delta$.*

Dimostrazione. L'insieme W deve soddisfare le proprietà **P1, P2** e **P3** della definizione di ε-approximation di k-NN$(\mathcal{G},V,s,k)$. L'algoritmo utilizza un numero di sample pari a $r = \frac{2}{\epsilon^2} ln(\frac{2n}{\delta})$ per il calcolo delle approssimazioni, quindi è facile vedere che la proprietà **P3** deriva direttamente dal Lemma 5.

Si considera quindi un ordinamento dei vertici $v \in V\backslash\{s\}$ per valori decrescti di $c_\alpha(s,v)$ con i pareggi tra vertici risolti in maniera arbitraria. Sia quindi $v_1,\ldots,v_{n-1}$ il nome dato ai vertici in tale ordinamento. Sia $c_\alpha^{(k)}$ lo score del nodo v_k ($c_\alpha^{(k)} = c_\alpha(s,v_k)$). Si consideri un qualsiasi vertice v_l, con $l \in [1,k]$. Abbiamo che:

$$\hat{c}_\alpha(s,v_l) \geq c_\alpha(s,v_l) - \frac{\varepsilon}{2} \geq c_\alpha^{(k)} - \frac{\varepsilon}{2}$$

Dove la prima disuguaglianza deriva dal Lemma 5 con probabilità almeno $1-\delta$, e la seconda è vera per costruzione, in quando i vertici v_l sono stati presi tra i primi k, quindi con $c_\alpha(s,v_l) \geq c_\alpha^{(k)}$.

Si hanno quindi almeno k vertici per cui lo score stimato è tale che $\hat{c}_\alpha(s,v) \geq c_\alpha^{(k)} - \varepsilon/2$. Ne segue che, in particolare, per il k-esimo nodo dell'ordinamento secondo il valore $\hat{c}_\alpha(s,v)$ vale che

$$\hat{c}_\alpha^{(k)} \geq c_\alpha^{(k)} - \frac{\varepsilon}{2}$$

A questo punto, $\forall$ coppia $(v,\hat{c}_\alpha(s,v)) \in W$ si ha

$$c_\alpha(s,v) \geq \hat{c}_\alpha(s,v) - \frac{\varepsilon}{2} \geq \hat{c}_\alpha^{(k)} - \frac{\varepsilon}{2} \geq c_\alpha^{(k)} - \varepsilon$$

dove la prima disuguaglianza deriva sempre dal Lemma 5, la seconda dal fatto che la coppia deve essere tale che $\hat{c}_\alpha(s,v) \geq \hat{c}_\alpha^{(k)}$, altrimenti non si troverebbe in W. La terza disequazione, infine, deriva da quanto detto subito sopra. Questo dimostra **P1**.

Per **P2** ora si osserva che o W contiene tutte e sole le coppie (v, $\hat{c}_\alpha(s,v)$) tali per cui $c_\alpha(s,v) \geq c_\alpha^{(k)}$, oppure contiene almeno una coppia tale che $c_\alpha(s,v) < c_\alpha^{(k)}$. Quindi per costruzione W deve contenere almeno una coppia $(z, \hat{c}_\alpha(s,z))$ tale che $c_\alpha^{(k)} \geq c_\alpha(s,z)$. Possiamo dire di questa coppia che:

$$\hat{c}_\alpha^{(k)} \leq \hat{c}_\alpha(s,z) \leq c_\alpha(s,z) + \frac{\varepsilon}{2} \leq c_\alpha^{(k)} + \frac{\varepsilon}{2}$$

Dove la prima disequazione è vera altrimenti la coppia non si troverebbe in W, la seconda è vera per il Lemma 5 con probabilità $1-\delta$ e la terza per l'osservazione fatta precedentemente.

Si può quindi concludere che vale la seguente relazione:

$$\hat{c}_\alpha^{(k)} \leq c_\alpha^{(k)} + \frac{\epsilon}{2} \tag{4.1}$$

Ora si consideri una coppia $(u, \hat{c}_\alpha(s,u)) \notin W$. Abbiamo quindi:

$$c_\alpha(s,u) \leq \hat{c}_\alpha(s,u) + \frac{\varepsilon}{2} < \hat{c}_\alpha^{(k)} + \frac{\epsilon}{2} \leq c_\alpha^{(k)} + \epsilon$$

Dove la prima disuguaglianza è vera per il Lemma 5, la seconda è vera altrimenti la coppia sarebbe all'interno di W, e l'ultima deriva dalla disuguaglianza 4.1 trovata più sopra. Questo verifica **P2** e conclude la dimostrazione. □

Sebbene questo algoritmo permetta senz'altro di calcolare la una ε-approssimazione di k-NN come dato dalla Definizione 4, resta il fatto che potrebbe richiedere molto tempo, in quanto è necessaria la visita completa di r grafi. Nella prossima sezione vengono proposte delle stopping conditions, via via meno stringenti, per permettere una computazione più rapida.

4.2 Stopping condition per algoritmi più veloci

Quando si calcolano i k-nearest neighbours in un grafo deterministico, non è necessario calcolare esplicitamente l'intero *shortes path tree* con radice al nodo s, bensì si può far fermare l'algoritmo non appena siano stati raggiunti tutti i nodi che distano da s non più del k-esimo.

L'algoritmo 1 richiede di calcolare esplicitamente questi *shortest path tree* in tutti gli r *possible worlds* a disposizione in quanto ciò è necessario per creare l'estimatore $\hat{c}_\alpha(s,v)$ per ogni nodo del grafo.

In questa sezione vengono definite delle stopping condition per permettere di calcolare questo stesso estimator, o un estimator altrettanto buono, senza necessariamente visitare tutti gli *shortest path tree* per ogni sample. Prima di introdurle però servono alcune definizioni.

Definizione 5. *Chiamiamo $\mathcal{S}$ l'insieme dei sampled graph, ossia gli r possible word estratti dall'uncertain graph $\mathcal{G}$.*

Definizione 6. *Con T_G si indica il* shortest path tree *con radice in s nel grafo $G \in \mathcal{S}$. Si suppone che per ognuno di questi G si sia esplorato T_G solo fino alla distanza D_G inclusa, dove D_G è peculiare di ogni G.*

Definizione 7. *Considerato un qualsiasi vertice $v \in V$, si definisce $\mathcal{R}_v \subseteq \mathcal{S}$ come insieme dei grafi $G \in \mathcal{S}$ in cui v è stato raggiunto da s tramite il* shortest path tree*. Similmente, viene definito con $\mathcal{U}_v = \mathcal{S} \backslash \mathcal{R}$ l'insieme dei grafi G in cui v non è stato ancora raggiunto da s.*

A questo punto, possiamo definire i seguenti due score:

Definizione 8. *Per ogni vertice $v \in V\{s\}$, in qualsiasi momento dello sviluppo degli* shortest path tree*, sono definiti i seguenti due score:*

$$\bar{c}_\alpha(s,v) = \frac{1}{r} \sum_{G \in \mathcal{R}_v} \frac{1}{(d_G(s,v))^\alpha} \quad \tilde{c}_\alpha(s,v) = \frac{1}{r} \sum_{G \in \mathcal{U}_v} \frac{1}{(D_G)^\alpha}$$

Si osservi come questi due score rappresentano rispettivamente, per così dire, quello che si sa "deterministicamente" della expected α-closeness del nodo v, ossia quello che si può dire con la computazione eseguita fino ad un certo punto, e l'incertezza che si continua ad avere nei suoi riguardi.

Inoltre, questi due valori sono tali che, in ogni momento, ossia in ogni punto dello sviluppo degli *shortest path tree*, si ha

$$\bar{c}_\alpha(s,v) \leq \hat{c}_\alpha(s,v) \leq \bar{c}_\alpha(s,v) + \tilde{c}_\alpha(s,v) \quad \forall v \in V \backslash \{s\}$$

4.2.1 Una prima stopping condition

Viene quindi proposta questa prima stopping condition:

$$\begin{gathered} \tilde{c}_\alpha(s,v) = 0 \quad \forall v \in V \backslash \{s\} : \bar{c}_\alpha(s,v) \geq \bar{c}_\alpha^{(k)} \\ \text{and} \quad \bar{c}_\alpha(s,v) + \tilde{c}_\alpha(s,v) < \bar{c}_\alpha^{(k)} \quad \forall v \in V \backslash \{s\} : \bar{c}_\alpha(s,v) < \bar{c}_\alpha^{(k)} \end{gathered} \tag{4.2}$$

Intuitivamente, questa stopping condition dice che ci si può fermare non appena l'incertezza, rappresentata dal valore $\tilde{c}_\alpha(s,v)$. raggiunge 0 per i nodi più alti nel ranking e, inoltre, quando i nodi che ricadono fuori dai k-NN sono in condizione

di non riuscire in alcun modo a risalire la classifica e scalzare coloro che stanno già in cima.

L'insieme restituito è quindi

$$W' = \{(v, \bar{c}_\alpha(s,v)) | \bar{c}_\alpha(s,v) \geq \bar{c}_\alpha^{(k)} \wedge \tilde{c}_\alpha(s,v) = 0\} \setminus \{(v, \bar{c}_\alpha(s,v)) | \bar{c}_\alpha(s,v) < \bar{c}_\alpha^{(k)} \wedge \bar{c}_\alpha(s,v) + \tilde{c}_\alpha(s,v) < \bar{c}_\alpha^{(k)}\}$$

La correttezza dell'algoritmo che fa uso di questa stopping condition è presentata nel Lemma 7.

Prima di introdurlo, si osservi che, in ogni caso, si dovranno visitare almeno k nodi in ognuno degli alberi T_G, per ognuno dei $G \in \mathcal{S}$. Quindi, si può pensare di costruire un algoritmo costituito da una prima fase di inizializzazione e successivamente da una fase iterativa. Nella prima fase, vengono calcolati i k-nearest neighbors in ogni grafo $G \in \mathcal{S}$, mentre nella fase iterativa l'algoritmo fa crescere gli alberi man mano, fino a quando non viene incontrata la stopping condition. Ogni volta che un nuovo nodo viene visitato in uno dei *possible world* di $\mathcal{S}$, si ha che la componente $\tilde{c}_\alpha(s,v)$ diminuisce, mentre la componente $\bar{c}_\alpha(s,v)$ aumenta. Tuttavia l'incremento di cui gode la seconda è sempre minore o uguale al decremento subito dalla la prima, per loro definizione.

Si osserva inoltre che nella fase iterativa potrebbero essere provate delle euristiche per la scelta dei cosiddetti *most promising tree*, ossia degli alberi T_G tra tutti quelli a disposizione che potrebbero portare, se esplorati per primi, ad un più veloce approssimarsi alla stopping condition.

Lemma 7. *Sia $\mathcal{G}$ un uncertain graph. Per un dato $s \in V$ e $k > 0$ e per dei fissati $\varepsilon, \delta \in (0,1)$, l'algoritmo 2 restituisce una ε-approximation di K-NN($\mathcal{G}, V, s, k$), con probabilità almeno $1 - \delta$.*

Dimostrazione. Un modo per dimostrare che l'insieme restituito dall'algoritmo 2 sia effettivamente una ε-approximation di k-NN è mostrare che esso presenti tutte e tre le proprietà **P1, P2** e **P3** della definizione 4. Tuttavia, esiste un modo più semplice ed immediato. Sia W l'insieme restituito dall'algoritmo 1 e W' l'insieme ritornato dall'algoritmo 2. Si dimostra che i due insiemi sono lo stesso e quindi, conseguentemente, che W' è una ε-approximation di k-NN per il Lemma 6.

Si consideri un $u \in W'$. Si ha che $\bar{c}_\alpha(s,u) = \hat{c}_\alpha(s,u)$ in quanto, per definizione di W', l'incertezza su u è nulla. Inoltre, si ha anche che $\bar{c}_\alpha^{(k)} = \hat{c}_\alpha^{(k)}$ in quanto tutti i nodi $v \notin W'$ sono tali che $\bar{c}_\alpha(s,v) + \tilde{c}_\alpha(s,v) < \bar{c}_\alpha^{(k)}$, e quindi nessuno di loro potrebbe superare nel ranking il nodo k-esimo. Questo significa che $\hat{c}_\alpha(s,u) \geq \hat{c}_\alpha^{(k)}$, e quindi $W' \subseteq W$.

Si consideri ora un $u \in W$. Si ha che $\hat{c}_\alpha(s,u) \geq \hat{c}_\alpha^{(k)}$ e anche qui $\bar{c}_\alpha(s,u) = \hat{c}_\alpha(s,u)$. In generale $\bar{c}_\alpha^{(k)} \leq \hat{c}_\alpha^{(k)}$ quindi a maggior ragione $\hat{c}_\alpha(s,u) \geq \bar{c}_\alpha^{(k)}$. Ne segue che $W \subseteq W'$.

□

Osservazione 2. *Si fa notare come questa stopping condition non necessariamente porti ad un decremento significativo dei tempi, in quanto è molto restrittiva. Infatti, nel worst case scenario, potrebbe capitare che un nodo che pure è destinato a finire in W' perché in $r-1$ sample appare vicino alla source s, nell'r-esimo sample sia raggiungibile solo attraverso un lungo path, che obblighi a dover esplorare tutto il grafo. Se questo avvenisse per tutti i vertici v destinati a finire in W', o per una gran parte di essi, i tempi di esecuzione non sarebbero diversi da quelli dell'algoritmo senza stopping condition. Questa osservazione porta alla prossima stopping condition, che contempla la possibilità di ammettere la presenza di piccoli valori $\tilde{c}_\alpha(s,v) \neq 0$ nell'output.*

4.2.2 Una seconda stopping condition

Viene proposta questa seconda stopping condition:

$$\begin{gathered}(|W'| = k \quad \text{or} \quad \tilde{c}_\alpha(s,v) = 0 \quad \forall v \in V \backslash \{s\} : \bar{c}_\alpha(s,v) \geq \bar{c}_\alpha^{(k)}) \\ \text{and} \\ (\bar{c}_\alpha(s,v) + \tilde{c}_\alpha(s,v) < \bar{c}_\alpha^{(k)} \quad \forall v \in V \backslash \{s\} : \bar{c}_\alpha(s,v) < \bar{c}_\alpha^{(k)})\end{gathered} \tag{4.3}$$

Come si nota, essa non è che un raffinamento della precedente, in quanto si prende in considerazione, anche se in maniera ancora molto restrittiva, il fatto che, nell'insieme soluzione restituito, ci possano essere nodi in cui la componente di incertezza $\tilde{c}_\alpha(s,v)$ non è stata portata a 0. L'algoritmo 2 può fare uso di quest'ultima esattamente nello stesso modo in cui faceva uso della condizione 4.2. Il lemma seguente dimostra la correttezza di tale condizione.

Lemma 8. *Sia $\mathcal{G}$ un uncertain graph. Per un dato $s \in V$ e $k > 0$ e per dei fissati $\varepsilon, \delta \in (0,1)$, l'algoritmo 2 che faccia uso della stopping condition 4.3 restituisce una ε-approximation di K-NN($\mathcal{G}, V, s, k$), con probabilità almeno $1-\delta$.*

Dimostrazione. Anche in questo caso, si chiama W' l'insieme risultato dell'algoritmo 2, mentre W quello di 1. Si dimostra che $W = W'$ e che quindi W', per il Lemma 6, è una ε-approximation di k-NN.

Si procede in maniera del tutto analoga al Lemma 7. Si considera un $u \in W'$. Quindi ci possono essere due casi di interesse. O $\bar{c}_\alpha(s,u) = \hat{c}_\alpha(s,u)$ in quanto, per definizione di W', l'incertezza su ogni u è nulla. In questo caso si ha anche che $\bar{c}_\alpha^{(k)} = \hat{c}_\alpha^{(k)}$ in quanto tutti i nodi $v \notin W'$ sono tali che $\bar{c}_\alpha(s,v) + \tilde{c}_\alpha(s,v) < \bar{c}_\alpha^{(k)}$, e quindi nessuno di loro potrebbe superare nel ranking il nodo k-esimo. Questo significa che $\hat{c}_\alpha(s,u) \geq \hat{c}_\alpha^{(k)}$. Ne segue che $W' \subseteq W$. Altrimenti $|W'| = k$, e in generale $\tilde{c}_\alpha(s,v) \geq 0$. Si ha tuttavia ancora che tutti i $v \notin W'$ sono tali che $\bar{c}_\alpha(s,v) + \tilde{c}_\alpha(s,v) < \bar{c}_\alpha^{(k)}$, e quindi nessuno di loro potrebbe superare nel ranking il nodo k-esimo, ed essendo già esattamente k i candidati k-NN, nessuno di essi, se

l'algoritmo continuasse ad espandere i T_{G_i}, uscirebbe dall'insieme. Ne segue che $W' \subseteq W$[1].

Si consideri ora un $u \in W$. Si ha che $\hat{c}_\alpha(s,u) \geq \hat{c}_\alpha^{(k)}$ e $\bar{c}_\alpha(s,u) = \hat{c}_\alpha(s,u)$. In generale $\bar{c}_\alpha^{(k)} \leq \hat{c}_\alpha^{(k)}$ quindi a maggior ragione $\hat{c}_\alpha(s,u) \geq \bar{c}_\alpha^{(k)}$. Ne segue che $W \subseteq W'$. □

Questa seconda condizione permette di introdurre dell'incertezza anche all'interno dei nodi presenti in W'. Tuttavia, risulta ancora molto rigida, il che porta a proporre la terza ed ultima stopping condition, che permetta di porre un upper bound al valore dell'incertezza $\tilde{c}_\alpha(s,v)\ \forall v \in V \backslash \{s\}$.

4.2.3 Una terza stopping condition

L'ultima stopping condition è la seguente:

$$\tilde{c}_\alpha(s,v) < \varepsilon'/2 \quad \forall v \in V \backslash \{s\}, \varepsilon' \in (0,1) \tag{4.4}$$

Essa pone un limite superiore all'incertezza che ci si permette di avere sui singoli nodi. L'algoritmo che ne fa utilizzo è del tutto analogo ancora una volta all'Algoritmo 2. Si osserva come questo algoritmo ha a disposizione solo il valore $\bar{c}_\alpha(s,v)$ per ogni v per creare il ranking, e quindi si sta usando questo, e non più $\hat{c}_\alpha(s,v)$, per approssimare l'expected α-closeness. Serve quindi, per dimostrare che questo algoritmo ritorni effettivamente una ε-approximation di k-NN, mostrare prima un modo per imporre un bound assoluto nella stima dell'α-closeness per tutti i nodi del grafo rispetto alla sorgente s.

Lemma 9. *Sia $\mathcal{G} = (V, E, w, p)$ un uncertain graph, e $s \in V$ il nodo sorgente. Sia inoltre $\tilde{c}_\alpha(s,v) < \varepsilon'/2 \quad \forall v \in V \backslash \{s\}, \varepsilon' \in (0,1)$. Per dei fissati $\delta, \varepsilon'' \in (0,1)$, con $\varepsilon'' > \varepsilon'$, e $\varepsilon = \varepsilon'' - \varepsilon'$, sia*

$$r = \frac{2}{\varepsilon^2} ln \frac{2n}{\delta}$$

il numero di sample estrapolati indipendentemente da $\mathcal{G}$ con reinserimento. Allora, con probabilità almeno $1 - \delta$ e $\forall v \in V \backslash \{s\}$ si ha che

$$|c_\alpha(s,v) - \bar{c}_\alpha(s,v)| < \frac{\varepsilon''}{2}$$

[1]Nel momento in cui si consente ai nodi in W' di avere una incertezza non nulla, il ranking è esclusivamente basato sui valori di $\bar{c}_\alpha(s,u)$. Il rischio è che ci sia più di un vertice che condivida il valore $\bar{c}_\alpha(s,v) = \bar{c}_\alpha^{(k)}$. In questo caso, nello sviluppare i T_{G_i} non c'è modo di garantire che gli score non varino in maniera da far uscire alcuni di questi nodi dall'insieme W' se esso è tale che $|W'| > k$.

Dimostrazione. Si consideri un arbitrario $v \in V \backslash \{s\}$. Siano $G_i \sqsubseteq \mathcal{G}$, per $i \in [1, r]$, i sample estratti da $\mathcal{G}$. Si ha che:

$$\hat{c}_\alpha(s,v) = \bar{c}_\alpha(s,v) + c^R_\alpha(s,v) \leq \bar{c}_\alpha(s,v) + \tilde{c}_\alpha(s,v) < \bar{c}_\alpha(s,v) + \varepsilon'/2$$

Dove si definisce $c^R_\alpha(s,v)$ come *residuo* che separa $\bar{c}_\alpha(s,v)$ da $\hat{c}_\alpha(s,v)$, a noi ignoto perché derivato dai nodi non ancora visitati nei $G \in \mathcal{S}$. Ora, si consideri l'evento:

$$|\bar{c}_\alpha(s,v) - c_\alpha(s,v)| \geq \frac{\varepsilon''}{2}$$

Questo è equivalente a scrivere

$$\begin{array}{ccc}
\bar{c}_\alpha(s,v) - c_\alpha(s,v) \geq \frac{\varepsilon''}{2} & \vee & \bar{c}_\alpha(s,v) - c_\alpha(s,v) \leq -\frac{\varepsilon''}{2} \\
\hat{c}_\alpha(s,v) - c^R_\alpha(s,v) - c_\alpha(s,v) \geq \frac{\varepsilon''}{2} & \vee & \hat{c}_\alpha(s,v) - c^R_\alpha(s,v) - c_\alpha(s,v) \leq -\frac{\varepsilon''}{2} \\
\hat{c}_\alpha(s,v) - c_\alpha(s,v) \geq \frac{\varepsilon''}{2} + c^R_\alpha(s,v) & \vee & \hat{c}_\alpha(s,v) - c_\alpha(s,v) \leq -\frac{\varepsilon''}{2} + c^R_\alpha(s,v)
\end{array}$$

Questo evento è contenuto in quello che segue:

$$\hat{c}_\alpha(s,v) - c_\alpha(s,v) \geq \frac{\varepsilon''}{2} - \frac{\varepsilon'}{2} \quad \vee \quad \hat{c}_\alpha(s,v) - c_\alpha(s,v) \leq -\frac{\varepsilon''}{2} + \frac{\varepsilon'}{2}$$

ossia:

$$|\hat{c}_\alpha(s,v) - c_\alpha(s,v)| \geq \frac{\varepsilon'' - \varepsilon'}{2}$$

Essendo un evento che contiene il precedente, la probabilità che esso avvenga ne è $\geq$. Per il lemma 4, si ricorda che $E[\hat{c}_\alpha(s,v)] = c_\alpha(s,v)$. Si osserva inoltre che le quantità $1/(d_{G_i}(s,v))^\alpha$ sono delle variabili aleatorie indipendenti che assumono valori in $[0,1]$. Per questo, si stanno soddisfando le ipotesi del lemma 3, e quindi, se chiamiamo $\varepsilon = \varepsilon'' - \varepsilon'$, possiamo applicare la disuguaglianza di Hoeffding come segue:

$$\begin{aligned}
Pr\left[|\bar{c}_\alpha(s,v) - c_\alpha(s,v)| \geq \frac{\varepsilon''}{2}\right] &\leq Pr\left[|\hat{c}_\alpha(s,v) - c_\alpha(s,v)| \geq \frac{\varepsilon}{2}\right] \\
&\leq 2exp\left(\frac{-2r^2(\varepsilon/2)^2}{\sum_{i=1}^r (b_i - a_i)^2})\right) \\
&= 2exp\left(-\frac{\varepsilon^2}{2}r\right)
\end{aligned}$$

Se ora sostituiamo al posto di r il numero dato nelle ipotesi si ottiene:

$$Pr\left[|\bar{c}_\alpha(s,t) - c_\alpha(s,t)| \geq \frac{\varepsilon''}{2}\right] \leq \frac{\delta}{n}$$

Il lemma segue applicando l'union bound su tutti i $v \in V \backslash \{s\}$ □

Lemma 10. *Sia $\mathcal{G}$ un uncertain graph. Per un dato $s \in V$ e $k > 0$ e per dei fissati $\varepsilon', \varepsilon''$, $\delta \in (0,1)$, con $\varepsilon'' > \varepsilon'$ e $\varepsilon = \varepsilon'' - \varepsilon'$, l'algoritmo 2 che faccia uso della stopping condition 4.4 restituisce una ε''-approximation di K-NN($\mathcal{G}, V, s, k$), con probabilità almeno $1 - \delta$.*

Dimostrazione. L'insieme W deve soddisfare le proprietà **P1, P2** e **P3** della definizione di ε-approximation di k-NN$(\mathcal{G}, V, s, k)$. L'algoritmo utilizza un numero di sample pari a $r = \frac{2}{\varepsilon^2} ln(\frac{2n}{\delta})$ per il calcolo delle approssimazioni, quindi è facile vedere che la proprietà **P3** deriva direttamente dal Lemma 9.

Si considera quindi un ordinamento dei vertici $v \in V \backslash \{s\}$ per valori decrescenti di $c_\alpha(s, v)$ con i pareggi tra vertici risolti in maniera arbitraria. Sia quindi $v_1, \ldots, v_{n-1}$ il nome dato ai vertici in tale ordinamento. Sia $c_\alpha^{(k)}$ lo score del nodo v_k ($c_\alpha^{(k)} = c_\alpha(s, v_k)$). Si consideri un qualsiasi vertice v_l, con $l \in [1, k]$. Abbiamo che:

$$\bar{c}_\alpha(s, v_l) \geq c_\alpha(s, v_l) - \frac{\varepsilon''}{2} \geq c_\alpha^{(k)} - \frac{\varepsilon''}{2}$$

Dove la prima disuguaglianza deriva dal Lemma 9 con probabilità almeno $1-\delta$, e la seconda è vera per costruzione, in quando i vertici v_l sono stati presi tra i primi k, quindi con $c_\alpha(s, v_l) \geq c_\alpha^{(k)}$.

Si hanno quindi almeno k vertici per cui lo score stimato è tale che $\bar{c}_\alpha(s, v) \geq c_\alpha^{(k)} - \varepsilon''/2$. Ne segue che, in particolare, per il k-esimo nodo dell'ordinamento secondo il valore $\bar{c}_\alpha(s, v)$ vale che:

$$\bar{c}_\alpha^{(k)} \geq c_\alpha^{(k)} - \frac{\varepsilon''}{2}$$

A questo punto, $\forall$ coppia $(v, \hat{c}_\alpha(s, v)) \in W$ si ha:

$$c_\alpha(s, v) \geq \bar{c}_\alpha(s, v) - \frac{\varepsilon''}{2} \geq \bar{c}_\alpha^{(k)} - \frac{\varepsilon''}{2} \geq c_\alpha^{(k)} - \varepsilon''$$

dove la prima disuguaglianza deriva sempre dal Lemma 9, la seconda dal fatto che la coppia deve essere tale che $\hat{c}_\alpha(s, v) \geq \hat{c}_\alpha^{(k)}$, altrimenti non si troverebbe in W. La terza inuguaglianza, infine, deriva da quanto detto subito sopra. Questo dimostra **P1**.

Per **P2** ora si osserva che o W contiene tutte e sole le coppie (v, $\bar{c}_\alpha(s, v)$) tali per cui $c_\alpha(s, v) \geq c_\alpha^{(v)}$, oppure contiene almeno una coppia tale che $c_\alpha(s, v) < c_\alpha^{(k)}$. Quindi per costruzione W deve contenere almeno una coppia $(z, \bar{c}_\alpha(s, z))$ tale che $c_\alpha^{(k)} \geq c_\alpha(s, z)$. Possiamo dire di questa coppia che:

$$\bar{c}_\alpha^{(k)} \leq \bar{c}_\alpha(s, z) \leq c_\alpha(s, z) + \frac{\varepsilon''}{2} \leq c_\alpha^{(k)} + \frac{\varepsilon''}{2}$$

Dove la prima disequazione è vera altrimenti la coppia non si troverebbe in W, la seconda è vera per il Lemma 9 con probabilità $1 - \delta$ e la terza per l'osservazione fatta precedentemente.

Si può quindi concludere che vale la seguente relazione:

$$\bar{c}_\alpha^{(k)} \leq c_\alpha^{(k)} + \frac{\varepsilon''}{2} \tag{4.5}$$

Ora si consideri una coppia $(u, \bar{c}_\alpha(s,u)) \notin W$. Abbiamo quindi:

$$c_\alpha(s,u) \leq \bar{c}_\alpha(s,u) + \frac{\varepsilon''}{2} < \bar{c}_\alpha^{(k)} + \frac{\varepsilon''}{2} \leq c_\alpha^{(k)} + \varepsilon''$$

Dove la prima disuguaglianza è vera per il Lemma 5, la seconda è vera altrimenti la coppia sarebbe all'interno di W, e l'ultima deriva dalla disuguaglianza 4.5 trovata più sopra. Questo verifica **P2** e conclude la dimostrazione. □

Capitolo 5

Esperimenti

In questo capitolo vengono eseguiti tre tipi di esperimenti per comparare l'α-closeness con la Median-Distance e testare l'efficacia delle stopping condition. Nella sezione 5.1 sono descritte per prima cosa le procedure utilizzate per preparare l'ambiente in cui si sono svolti gli esperimenti. Nella Sezione 5.2 vengono comparate la Median-Distance e la α-closeness utilizzando α=1. Successivamente, nella Sezione 5.3 vengono utilizzati altri valori per il parametro α per verificare come la sua variazione influisca sulle prestazioni della misura. Infine, nella Sezione 5.4 viene sperimentato come l'utilizzo delle stopping condition possa avere un impatto nei tempi di esecuzione dell'algoritmo.

5.1 Setup degli esperimenti

Nei seguenti esperimenti verranno utilizzati quattro grafi come dataset. Tre di questi sono delle mesh, il quarto è la rete stradale di Roma. Le loro caratteristiche sono riportate nella Tabella 5.1. Questi grafi sono dati in dei file contenenti la lista delle coppie di vertici che costituiscono i lati del grafo, uno per ogni riga. Nel caso del grafo di Roma, ogni riga ha tre valori, dove i primi due sono i vertici costituenti il lato e il terzo è il suo peso. In questa tesi in ogni caso nessuno di questi viene considerato come grafo pesato.

Si fa notare che tutti questi grafi sono deterministici, ossia i lati che presentano hanno tutti probabilità di esistenza pari ad 1. Si rivela quindi indispensabile in qualche modo "perturbare" suddetti grafi, ossia renderli probabilistici introducendo del rumore sui lati, in maniera simile a come si potrebbe avere se essi fossero in qualche modo letti da degli strumenti di misurazione che possono commettere un certo errore nelle loro rilevazioni.

Viene utilizzato il linguaggio C++, facendo uso della Boost Graph Library come supporto per la gestione dei grafi. Nelle sezioni seguenti è sinteticamente

nome grafo	numero vertici	numero archi
mesh10	100	180
mesh32	1024	1984
mesh64	4096	8064
rome99	3354	4831

Tabella 5.1: *Caratteristiche dei grafi utilizzati negli esperimenti. Nel conteggio dei lati del grafo rome99 si sono considerati i lati non diretti.*

descritto come è stata realizzata la perturbazione dei grafi e i criteri seguiti per preparare gli esperimenti.

5.1.1 Inizializzazione e lettura del grafo

Vengono per prima cosa letti i parametri necessari per lo sviluppo dell'esperimento. Oltre al path del file contenente le coppie di lati che vanno a formare il grafo, viene data la possibilità di indicare i parametri dell'esperimento come l'α della closeness; il *noise*, ossia il rumore gaussiano β utilizzato per perturbare il grafo deterministico a disposizione, e l'errore assoluto ε che si vuole compiere nell'approssimazione delle closeness, stesso parametro visto nella Definizione 4 di ε-approximation. Questi parametri hanno comunque un valore di default che viene loro assegnato in caso di mancata indicazione. Per α è 1, per il noise è 0.1 e per ϵ è 0.1. Questi sono i valori con cui si porteranno avanti gli esperimenti qui mostrati a parte per il noise, per il quale si useranno i valori 0.3, 0.5 e 0.8 per avere una maggiore incertezza sui rami dei grafi e mettere quindi maggiormente alla prova le misure. Per quel che riguarda il numero di sample r, si segue ovviamente la formula data dal Lemma 5, utilizzando $\delta = 1/n$. Lo stesso r viene usato sia per il calcolo della Median-Distance che per la α-closeness essendo abbastanza grande per entrambe.

La prima cosa che si esegue è la lettura dei lati del grafo attraverso i quali viene generata la mappa dei lati e dei vertici dello stesso. Sebbene si sia prevista la possibilità che questi vertici possano avere un peso, in questa tesi si considerano vertici non pesati e lati non orientati. Questo perché inserire dei pesi sui lati porterebbe ad un nuovo ulteriore grado di libertà che per il momento non si desidera dover gestire. Durante il caricamento del grafo in memoria, la probabilità di tutti i lati è posta ad 1, e quindi, inizialmente, si presenta un grafo deterministico.

5.1.2 Nodo sorgente e ground truth

Prima di rendere il grafo probabilistico bisogna ottenere una ground truth di riferimento per i calcoli successivi. Serve cioè un ordinamento dei vertici nel reale ordine di distanza che essi hanno dal nodo sorgente. Per questo basta eseguire

l'algoritmo di visita dei nodi (BFS o Dijkstra, ad esempio, ma in questo caso ha più senso il BFS dato che il grafo non è pesato) e tenere traccia delle loro distanze dal nodo sorgente.

Dato che i test vengono eseguiti su grafi diversi dalle migliaia di nodi ciascuno, si sceglie ogni volta una sorgente diversa. Per far questo si fa uso di una distribuzione di numeri interi uniformemente distribuita tra 0 e il numero di vertici nel grafo -1. Questa distribuzione permetterà di ottenere un numero in quest'intervallo, che sarà scelto come vertice s dell'esperimento di volta in volta.

Data la sorgente, si esegue l'algoritmo di visita per calcolare le distanze dei vertici e disporli in ordine. Da queste si possono conoscere i k-NN per ogni k desiderato, e quindi in base a questi potranno essere calcolati i vari parametri tra cui precision e recall degli algoritmi fondati sulle due misure in esame. Si ricordano le formule con cui si calcolano queste due misure:

$$\text{precision} = \frac{tp}{tp + fp} \qquad \text{recall} = \frac{tp}{tp + fn}$$

Dove ovviamente si sta intendendo *tp: true positive*, *fp: false positive*, *fn: false negative*.

5.1.3 Rendere il grafo probabilistico

Il passo successivo consiste nel rendere probabilistico il grafo, ossia introdurre una distribuzione di probabilità sui suoi lati. Si è scelto di realizzare questa distribuzione sottraendo randomicamente della probabilità a quei lati che possiedono probabilità 1 di esistenza e viceversa aggiungendo della probabilità ad alcuni dei lati che non esistono. Non si può dare probabilità di esistenza a tutti i lati del grafo in quanto ciò lo renderebbe completo e quindi poco significativo per il calcolo delle distanze di un vertice dagli altri. Pertanto, la probabilità per cui un nuovo lato viene creato e gli viene data una certa probabilità di esistenza è determinata dalla formula

$$\text{new edge probability} = \frac{2n}{m^2}$$

La quale garantisce un numero di lati aggiunti circa pari a quelli già presenti nel grafo.

A questo punto, ad ogni arco esistente viene assegnata una probabilità pari ad $1 - x$, dove x è un valore generato con distribuzione esponenziale di parametro $\lambda = 1/\beta$. Per i lati che non esistono si decide prima di tutto se aggiungerli o meno. Per questo si crea un apposito *functor* che faccia da generatore di valori come quelli dati da una variabile aleatoria uniformemente distribuita nell'intervallo $[0, 1]$. Viene eseguito un doppio ciclo da ogni possibile nodo sorgente ad ogni possibile nodo destinazione. Dato che in questo caso il grafo non è diretto, per non contare

due volte lo stesso lato ci si assicura di partire sempre da un nodo di indice inferiore e di avere a che fare con un lato che non esista già. Ora, tramite il *functor* creato precedente, si ottiene un numero in $[0, 1]$ e se questo è $\leq$ alla probabilità indicata di creare un nuovo lato, lo si genera dandogli come probabilità di esistenza quella ottenuta dal *functor* di perturbazione.

Una volta reso il grafo probabilistico in questo modo, si può procedere con i calcoli.

5.1.4 Calcolo dell'α-closeness e della Median-Distance dei nodi

Data la natura delle misure che si stanno prendendo, è necessario utilizzare un metodo Monte-Carlo per approssimare le reali α-closeness e Median-Distance. Per questo, si procede al calcolo del numero r di *possible world* da estrarre dall'uncertain graph $\mathcal{G}$ creato. Per quel che riguarda il numero di sample, coe già accennato, si utilizza il valore $r = 2/\varepsilon^2 \ln(2n/\delta)$ (dove si è scelto di porre $\delta = 1/n$), dato dal Lemma 5. Questo numero di sample è usato per entrambe le misure dato che, richiedendo la Median-Distance un numero pari a $c/\varepsilon^2 \ln(2/\delta)$, minore di r se $c = 3$, può andare bene lo stesso per il Lemma 1.

A questo punto viene eseguito un semplice calcolo delle distanze di ogni vertice dalla sorgente designata s in ognuno degli r possible world. Per ragioni di efficienza sia in termini di tempo che di memoria, non si creano r grafi deterministici estratti da $\mathcal{G}$, bensì si esegue l'algoritmo di attraversamento per r volte sul grafo probabilistico e, ogni volta che si dovrebbe visitare un lato, che comunque è presente nel grafo con la sua probabilità di esistenza, si esegue prima una funzione che permetta di decidere se quel lato esiste o meno. In pratica, si esegue *on the fly* il sampling del *possible world* in cui ci si trova al momento. Per gestire la conoscenza dei lati e dei nodi visitati, oltre che la loro distanza dalla sorgente s, vengono utilizzate delle apposite mappe messe a disposizione dalla libreria boost.

Durante questi r attraversamenti viene popolata una mappa che mette in relazione ogni vertice v del grafo con una lista, nella quale sono elencate tutte le distanze che il vertice in questione ha assunto in ognuno dei *possible world* che si sono generati.

Ottenuta questa lista, abbiamo tutti i dati necessari per calcolare le misure di α-closeness e Median-Distance dei vari nodi.

Per quel che riguarda le α-closeness non si fa che utilizzare la formula per il suo unbiased estimator $\hat{c}_\alpha(s, v) = \frac{1}{r} \sum_{i=1}^{r} \frac{1}{d(s,v)^\alpha} \ \forall v \in V \backslash \{s\}$. Successivamente si pone questa lista in ordine decrescente.

Anche per la Median-Distance non si fa che seguire la definizione e il Lemma 1 da essa sviluppato nella parte teorica, e quindi per ogni vertice si pone in ordine

crescente la lista delle distanze in cui appare negli r sample sviluppati, dopodiché si prende la distanza che compare all'indice $r/2$ come Median-Distance. A questo punto si è ottenuta una mappatura della Median-Distance per tutti i nodi del grafo, che possono essere messi a loro volta in ordine crescente di distanza dalla sorgente.

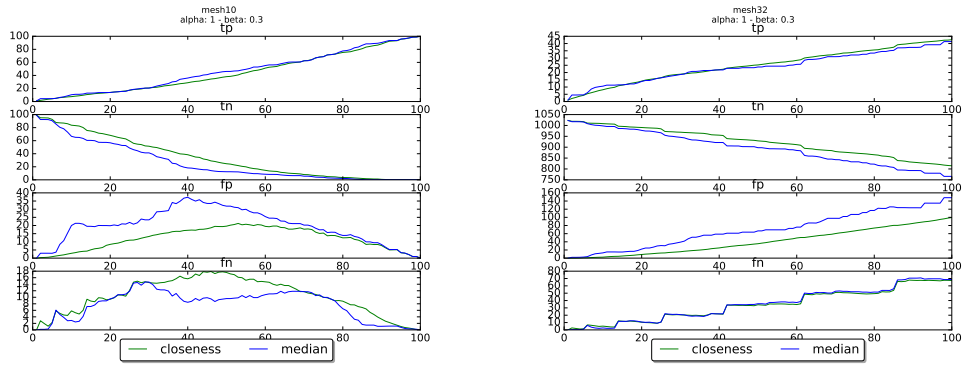
Questi due ordinamenti rappresentano il modo in cui gli algoritmi eseguono il ranking dei nodi in base alla sorgente scelta. Scegliendo qualsiasi k si può quindi facilmente estrapolare da queste liste l'insieme dei k-NN desiderati. Sarà utilizzando queste che si otterranno i risultati seguenti.

5.2 Primo esperimento

Per il calcolo dell'α-closeness si fa utilizzo dell'algoritmo esaustivo che esplora tutti i sample generati. In questo primo esperimento viene utilizzato $\alpha = 1$ e vengono eseguite 30 run per ogni caso, in maniera da mediare sui risultati. Inoltre, nei grafi che si vedranno più avanti, il computo dei nodi ammessi nell'insieme dei k-NN da parte dell'α-closeness fa uso di una certa tolleranza, data dall'errore assoluto moltiplicato per 0.1, in maniera da non risultare *unfair* nei confronti della misura, in quanto a valori reali. L'errore assoluto da solo forniva una tolleranza eccessivamente lasca, in quanto, soprattutto per i grafi più grandi, i valori di closeness scendevano velocemente a valori molto bassi e venivano conseguentemente considerati quasi subito come positivi, aumentando il numero di falsi positivi e quindi facendo calare drasticamente la precision e salire ad 1 il recall. Il coefficiente 0.1 è stato scelto dopo vari tentativi per cercare di ottenere una raffigurazione quanto più possibile *fair* per l'α-closeness.

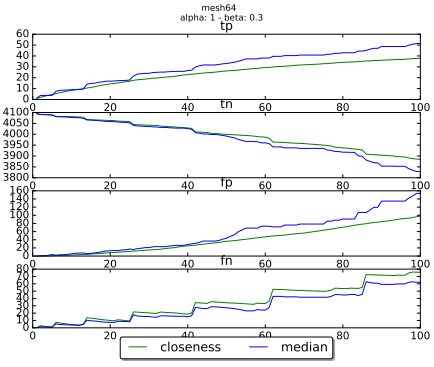
Nei seguenti grafici verranno mostrati i plot delle misure ottenute confrontando i due algoritmi e utilizzando come ground truth il grafo deterministico che è stato reso probabilistico con la procedura sopra indicata. Si considera in questo primo esperimento il valore di α fissato ad 1 mentre, come già spiegato, per il noise β verranno usati in sequenza i valori 0.3, 0.5 e 0.8. Si fa notare come questi valori non sono altro che la media del rumore introdotto nei lati. Quindi, ad esempio, quando si utilizza il valore $\beta = 0.5$ singifica che si sta togliendo dai lati che esistono già una probabilità di esistenza data da una variabile aleatoria esponenziale di media 0.5, e si aggiunge ai lati che si stanno creando una quantità di probabilità di esistenza dello stesso tipo. Si comprende quindi come per il caso 0.3 si abbia un'aggiunta di noise limitata, andando via via ad aumentare fino ad introdurre una perturbazione molto forte con 0.8.

Si inizia riportando nelle Figure 5.1, 5.2 e 5.3 i valori grezzi tp, tn, fp e fn per i quattro grafi. Si noti come i grafici mostrino il variare di questi valori man mano che il parametro k aumenta da 1 a 100.

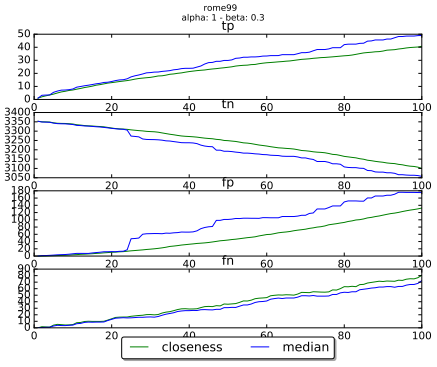


(a) *mesh10.*

(b) *mesh32.*

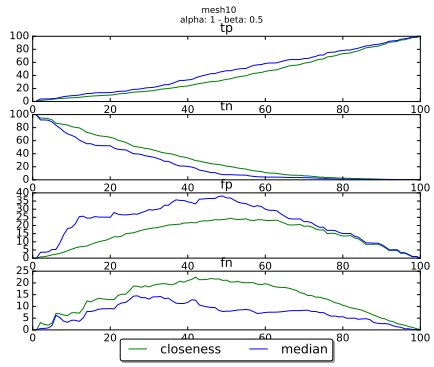


(c) *mesh64.*

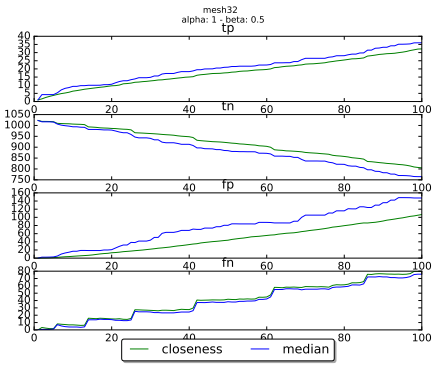


(d) *rome99.*

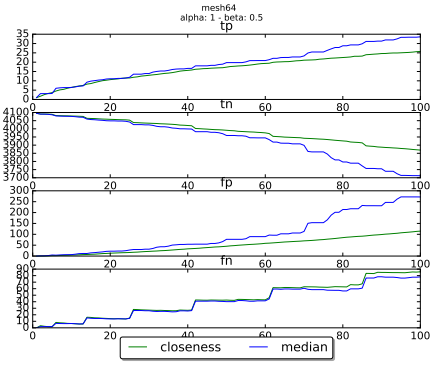
Figura 5.1: *I valori di true positive, true negative, false positive e false negative ottenuti dai due algoritmi sui quattro grafi per* $\alpha = 1$ *e* $\beta = 0.3$.



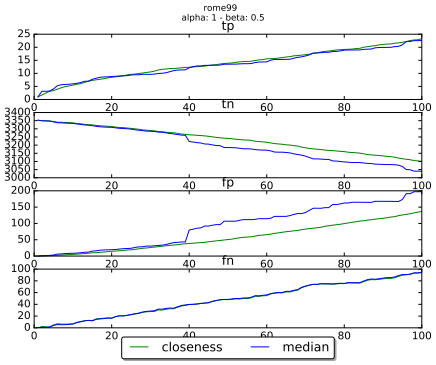
(a) *mesh10.*



(b) *mesh32.*

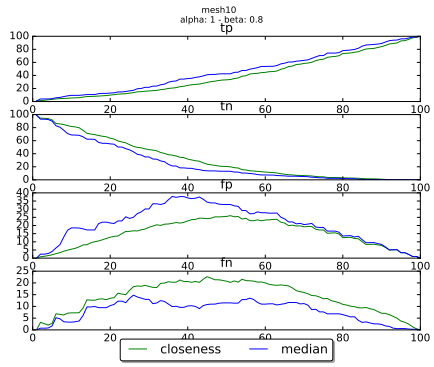


(c) *mesh64.*

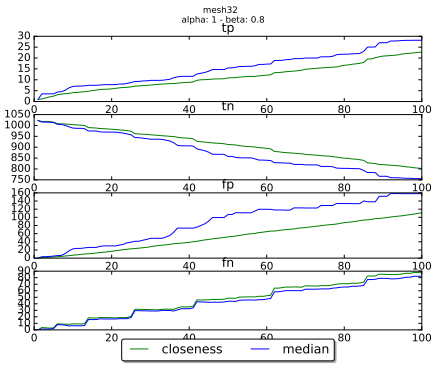


(d) *rome99.*

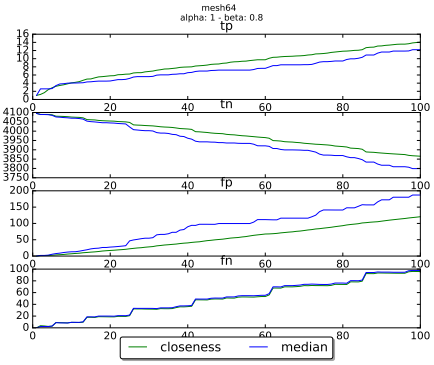
Figura 5.2: *I valori di true positive, true negative, false positive e false negative ottenuti dai due algoritmi sui quattro grafi per* $\alpha = 1$ *e* $\beta = 0.5$.



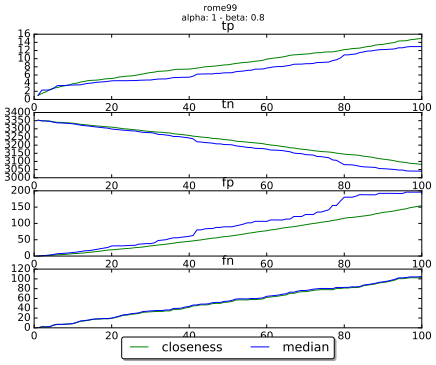
(a) *mesh10.*



(b) *mesh32.*



(c) *mesh64.*



(d) *rome99.*

Figura 5.3: *I valori di true positive, true negative, false positive e false negative ottenuti dai due algoritmi sui quattro grafi per* $\alpha = 1$ *e* $\beta = 0.8$.

Partendo dalla Figura 5.1, il caso con rumore più contenuto, il numero di true positive è maggiore per la Median-Distance rispetto alla closeness per i grafi più grandi, mentre è comparabile per le due mesh più piccole. Di convesso, la closeness riesce ad avere un numero di true negative maggiore, e di false positive minore. Questo può essere dovuto al fatto che, essendo una misura a valori reali, include in generale meno positivi della Median-Distance, che quindi in generale avrà un maggior numero di true positive pagando il costo di includere anche più falsi positivi. Un comportamento simile si ha alla Figura 5.2, dove si può vedere come la Median-Distance si dimostra lievemente migliore alla closeness per quel che riguarda la capacità di individuare true positive (sebbene nel caso di rome99 le due siano equivalenti, con la closeness che però riesce ad avere sempre degli andamenti migliori per i true negative e i false positive). Tuttavia, quando il rumore diventa più predominante, nel caso della Figura 5.3, l'α-closeness si dimostra migliore della Median-Distance per quel che riguarda i grafi più grandi, riuscendo ad avere un maggior numero di true positive e true negative, mantenendo più basso il valore di false positive. Si comprende quindi che, all'aumentare del rumore, la capacità della Median-Distance di individuare true positive si riduce. Questo è facilmente spiegabile considerando il fatto che la Median è una misura più prettamente basata sulla topologia del grafo, mentre la closeness prende maggiormente in considerazione la probabilità dei cammini minimi tra i vertici, oltre che la loro distanza.

Si nota ad ogni modo come, man mano che aumenti il noise, cali il numero assoluto di true positive che riescono ad essere individuati, mostrando che più cresce β e più diventa difficile trovare i corretti vicini dei nodi (si passa ad esempio da un valore assoluto di 50 nodi per rome99 a $\beta = 0.3$ a 16 per $\beta = 0.8$).

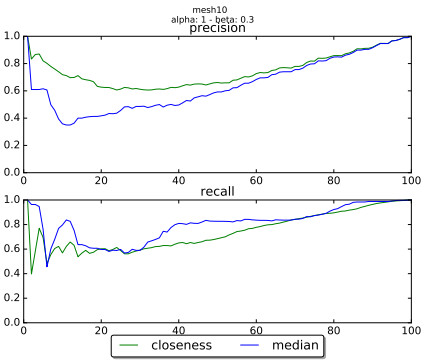
Una volta ottenuti tutti questi parametri è immediato calcolare quelli derivati di *precision* e *recall*. Per dare un riferimento, nelle Figure 5.4, 5.5 5.6 è riportato l'andamento per $\beta = 0.3, 0.5$ e 0.8 rispettivamente, mentre α è sempre fissato ad 1.

Come si può vedere, specialmente per la mesh10 e la mesh32, i valori di precision dati dall'α-closeness sono migliori rispetto quelli della Median-Distance.

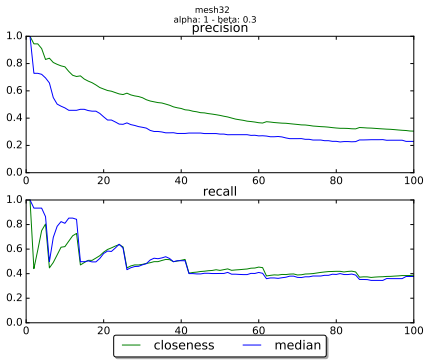
Non solo, mentre con rumore più piccolo dato da $\beta = 0.3$ le due misure si dimostrano equivalenti per quel che riguarda la precision, all'aumentare del noise si evidenzia non solo un ovvio decremento delle prestazioni per entrambe, ma soprattutto il fatto che l'α-closeness assume un vantaggio sulla Median-Distance, dimostrandosi ad essa superiore quando il grafo è fortemente perturbato.

Questo, accompagnato al fatto che come mostrato nel capitolo 3 la closeness è più affidabile e stabile per quanto riguarda l'approssimazione rispetto alla Median, mostra come la prima possa essere utilizzata con più successo e stabilità nei grafi probabilistici, soprattutto nel caso in cui in essi il noise sia più presente.

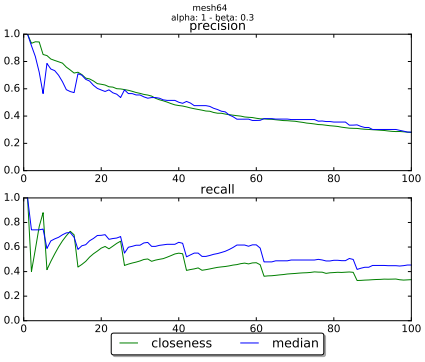
Si mette in evidenza come l'andamento della recall, rispetto a quello della pre-



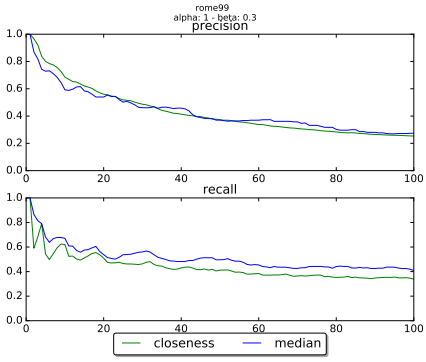
(a) *mesh10.*



(b) *mesh32.*

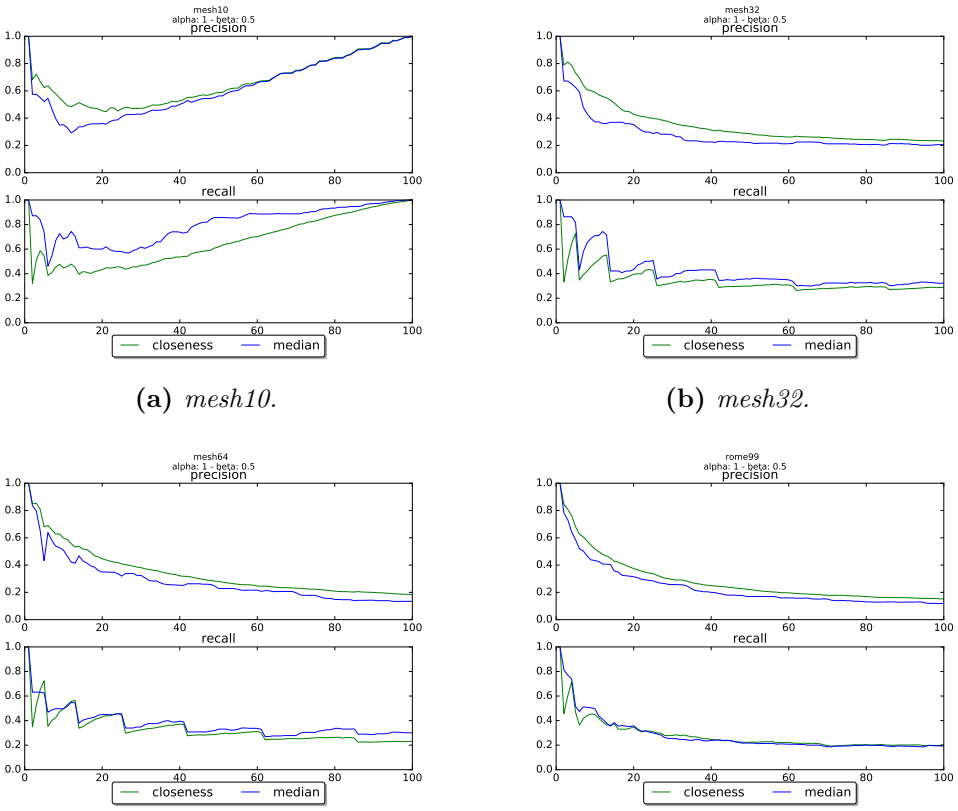


(c) *mesh64.*



(d) *rome99.*

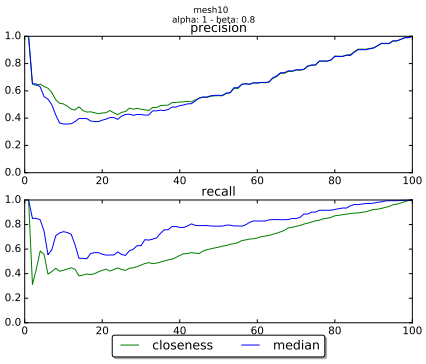
Figura 5.4: *Le precision e recall ottenute dagli esperimenti sulle 4 mesh con* $\alpha = 1$ *e* $\beta = 0.3$



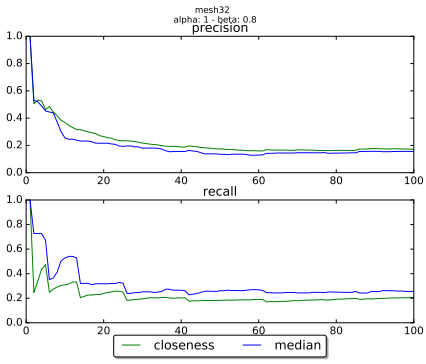
(a) *mesh10.* **(b)** *mesh32.*

(c) *mesh64.* **(d)** *rome99.*

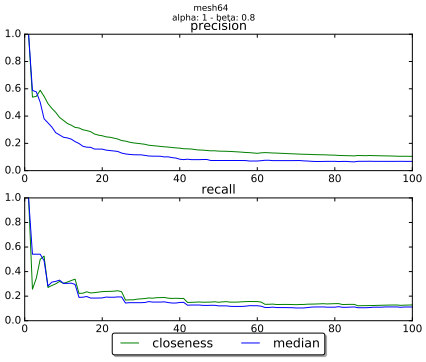
Figura 5.5: *Le precision e recall ottenute dagli esperimenti sulle 4 mesh con* $\alpha = 1$ *e* $\beta = 0.5$



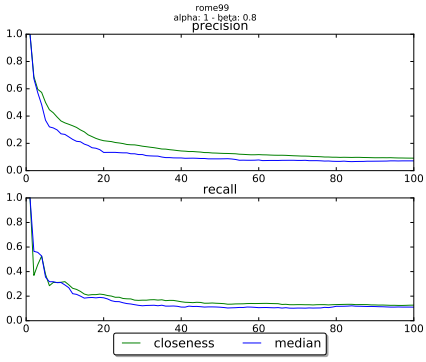
(a) *mesh10.*



(b) *mesh32.*



(c) *mesh64.*



(d) *rome99.*

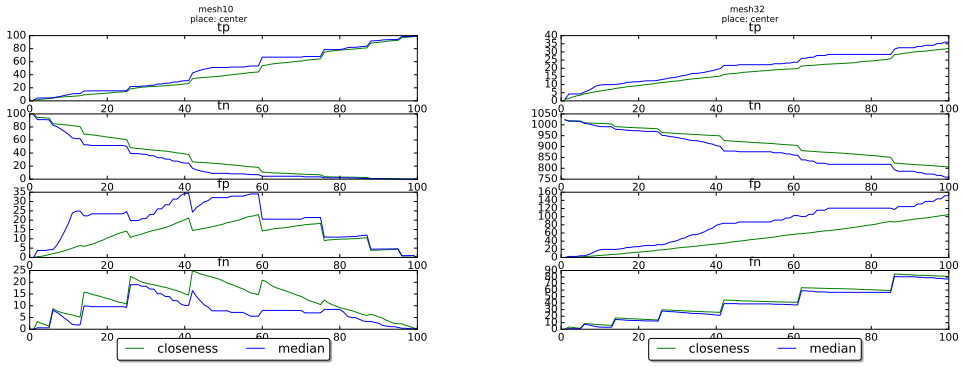
Figura 5.6: *Le precision e recall ottenute dagli esperimenti sulle 4 mesh con* $\alpha = 1$ *e* $\beta = 0.8$

cision, sia più altalenante nei vari casi. Appare in tutti i grafici come il numero di nodi ritrovati cali velocemente e poi risalga con un nuovo andamento a dente di sega, per quanto mitigato dalla scelta di includere comunque dei nodi tramite la tolleranza data dall'errore assoluto come spiegato precedentemente. Questo è ancora facilmente spiegabile tramite la natura del problema dei k-NN. La ground truth presenta, al variare del parametro k, un insieme dei k-NN che quasi mai è esattamente di cardinalità k, in quanto vi sono spesso alcuni nodi con la stessa distanza dalla sorgente del k-esimo. Differentemente, la misura di α-closeness è a valori reali tra 0 ed 1 e concede ai nodi uno score che è anche influenzato dalle probabilità, oltre che dalle distanze, e pertanto è molto improbabile che si presenti un gruppo di nodi con lo stesso valore di closeness del k-esimo. Questo porta al caratteristico andamento a dente di sega del parametro di recall che si vede soprattutto nelle mesh più popolose, come 5.4b e 5.4c. Ogniqualvolta vi è un brusco decremento del valore di recall, esso è succeduto da un graduale reinnalzamento, dovuto al fatto che si permette a nuovi nodi di entrare nell'insieme di k-NN restituito dall'algoritmo che fa uso della misura di α-closeness, mentre la ground truth resta stabile. Questo significa che anche quando il recall della closeness cala, è per il fatto che vengono tenuti fuori i nodi della "coda" dei k-NN, in ogni caso tra i più distanti. L'effetto a dente di sega ad ogni modo decresce di intensità man mano che il numero di nodi aumenta, dato che concorrentemente nella ground truth diventa meno probabile trovare un gruppo di vertici che condividano la stessa distanza al k-esimo posto della classifica.

Inoltre, questo stesso esperimento è stato provato anche con valori di β pari a 0.1 e 0.2, e in questi casi la Median si mostrava particolarmente performante. Ma questo è piuttosto prevedibile dato che, con valori di rumore così bassi, il grafo di fatto è quasi come se fosse deterministico. Infatti la perturbazione è quasi ininfluente, e basterebbe un semplice controllo sui lati per discernere quelli reali da quelli aggiunti, eliminando così l'incertezza. Le misure sono messe veramente alla prova quando il parametro β viene fatto aumentare in maniera più decisiva, rendendo impossibile individuare il grafo deterministico reale dietro alle probabilità introdotte dalla perturbazione. Anche sotto la luce di questa osservazione appare come la closeness, in questi casi, si dimostri più affidabile della Median.

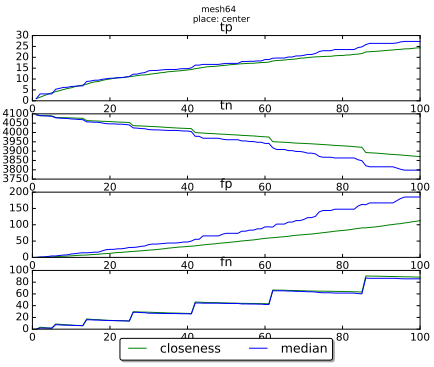
5.2.1 Variante con scelta del nodo sorgente

Gli esperimenti finora visti hanno mediato su 30 run, ognuna delle quali faceva uso di nodi source estratti a random dall'insieme V. Data la struttura di mesh dei primi tre grafi utilizzati, può essere interessante osservare il rapporto tra le due misure scegliendo il nodo sorgente tra i seguenti tre casi: nodo centrale della mesh (quattro vicini nella versione deterministica), nodo su di un lato (tre vicini) e nodo sull'angolo (due vicini). I grafici che seguono sono stati ottenuti mediando



(a) *mesh10.*

(b) *mesh32.*



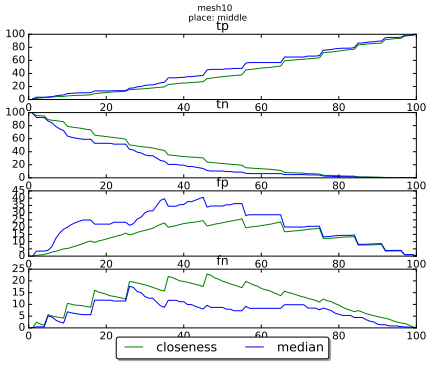
(c) *mesh64.*

Figura 5.7: *I valori di tp, tn, fp e fn ottenuti dalle mesh utilizzando il nodo centrale come sorgente,* $\alpha = 1$ *e* $\beta = 0.5$.

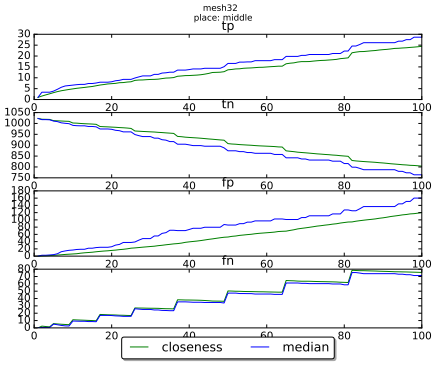
30 esperimenti eseguiti partendo dallo stesso nodo sorgente, utilizzando $\alpha = 1$ e $\beta = 0.5$.

Come si può notare alle Figure 5.7, 5.8 e 5.9, gli andamenti nei tre casi restano comunque simili, nonostante la differenza di posizione. Si nota che le misure sono paragonabili, con un maggior numero di tp per la Median quando il numero k tende ad aumentare. Non si discostano troppo dai risultati già visti precedentemente.

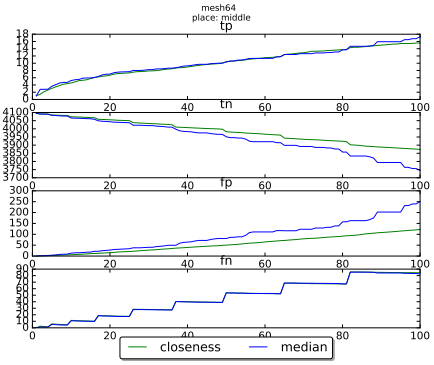
Ne emerge tuttavia il fatto interessante che, per quanto riguarda le due mesh più grandi, il numero assoluto di true positive che si riesce a recuperare è più grande nel caso di nodo sorgente centrale (dove si arriva rispettivamente a 40 e 30 circa) calando poi spostandosi nella periferia (sui 18 per il nodo a metà di un lato e 16 per il nodo 0 in angolo nel grafo rome99), indice di come la posizione della sorgente abbia una influenza non trascurabile sul risultato finale.



(a) *mesh10.*

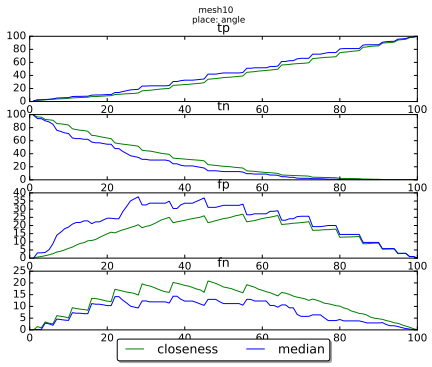


(b) *mesh32.*

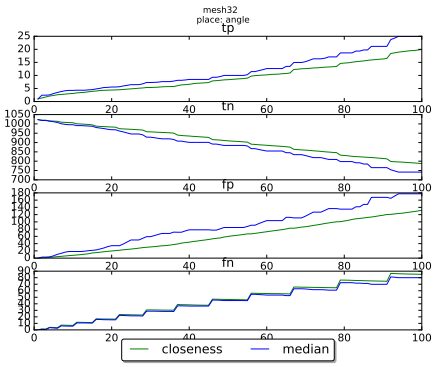


(c) *mesh64.*

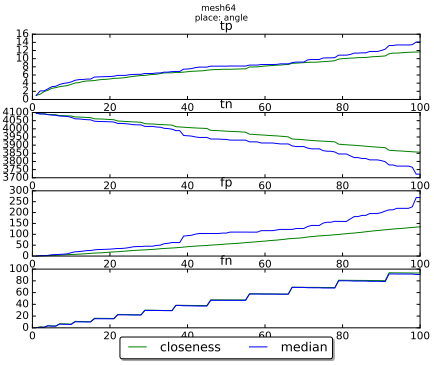
Figura 5.8: *I valori di tp, tn, fp e fn ottenuti dalle mesh utilizzando un nodo nel mezzo di un lato come sorgente,* $\alpha = 1$ *e* $\beta = 0.5$.



(a) *mesh10.*

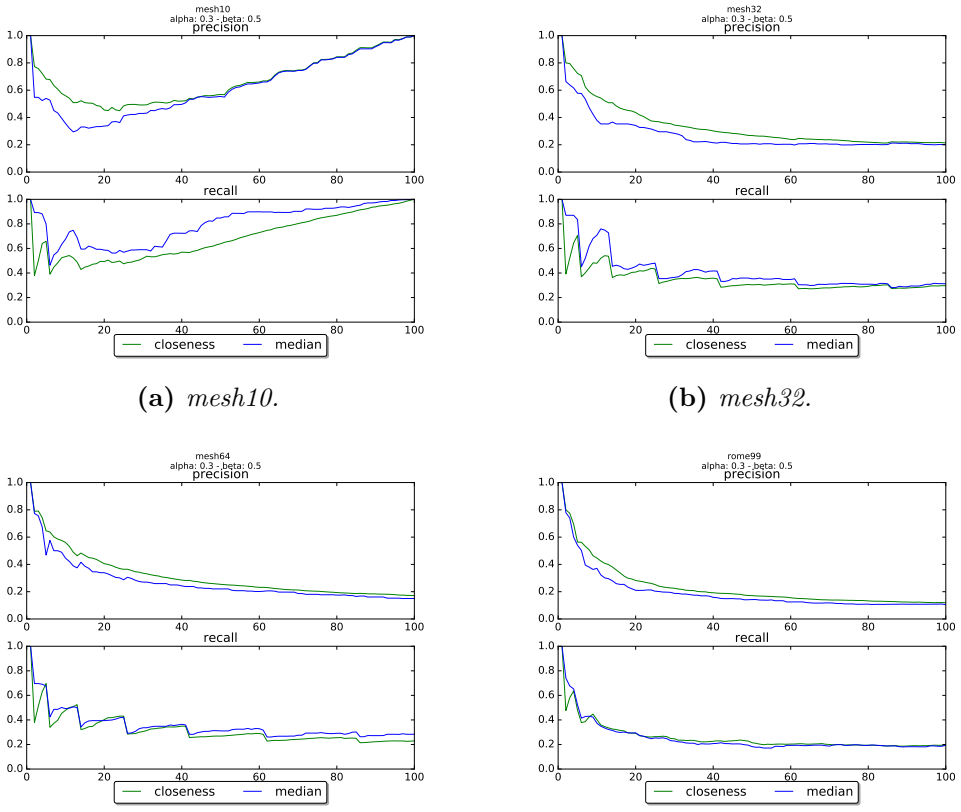


(b) *mesh32.*



(c) *mesh64.*

Figura 5.9: *I valori di tp, tn, fp e fn ottenuti dalle mesh utilizzando il nodo in angolo 0 come sorgente,* $\alpha = 1$ *e* $\beta = 0.5$.



(a) *mesh10.* **(b)** *mesh32.*

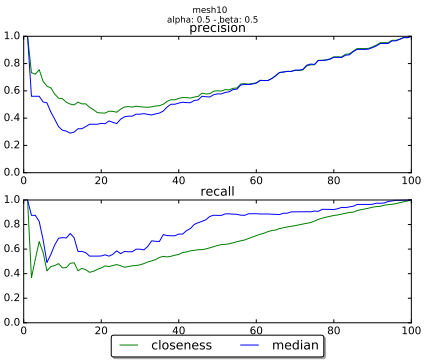
(c) *mesh64.* **(d)** *rome99.*

Figura 5.10: *I valori di precision e recall ottenuti dalle mesh utilizzando* $\alpha = 0.3$ *e* $\beta = 0.5$

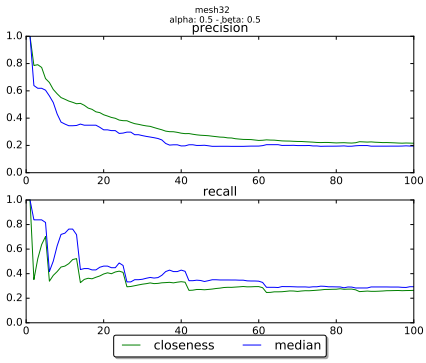
5.3 Secondo esperimento

In questo secondo esperimento si confrontano ancora le misure di precision e recall dell'algoritmo che fa uso dell'α-closeness rispetto alla Median-Distance, solo che questa volta si modificano gli α per studiarne l'impatto sulle performance. Si è già descritto precedentemente come un α più grande favorisca i nodi più vicini rispetto a quelli più lontani. In questa sezione sono provati come valori per α $0.3, 0.5$ e 2 mentre per β si tiene fisso 0.5.

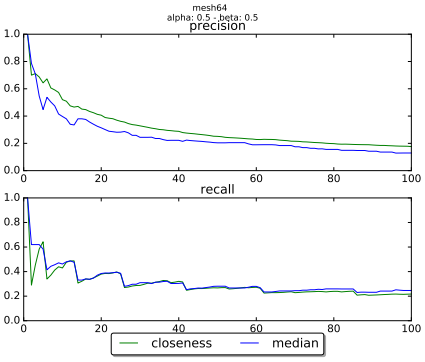
Il paragone può essere fatto con la Figura 5.5 e riguarda le Figure 5.10, 5.11 e 5.12. L'introduzione di un α lievemente minore come 0.3 o 0.5 non porta a delle variazioni particolarmente significative, soprattutto per quel che riguarda i grafi con un maggiore numero di nodi, che sono quelli a cui siamo più interessati. Si può notare peraltro che per k non troppo alti, fino ad esempio pari a 20, i grafici



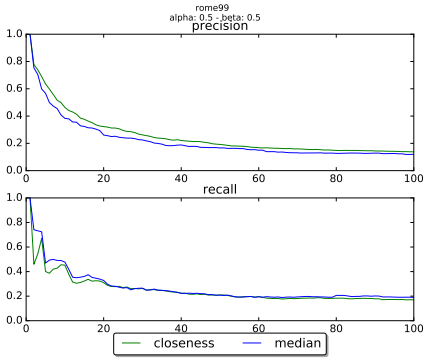
(a) *mesh10.*



(b) *mesh32.*

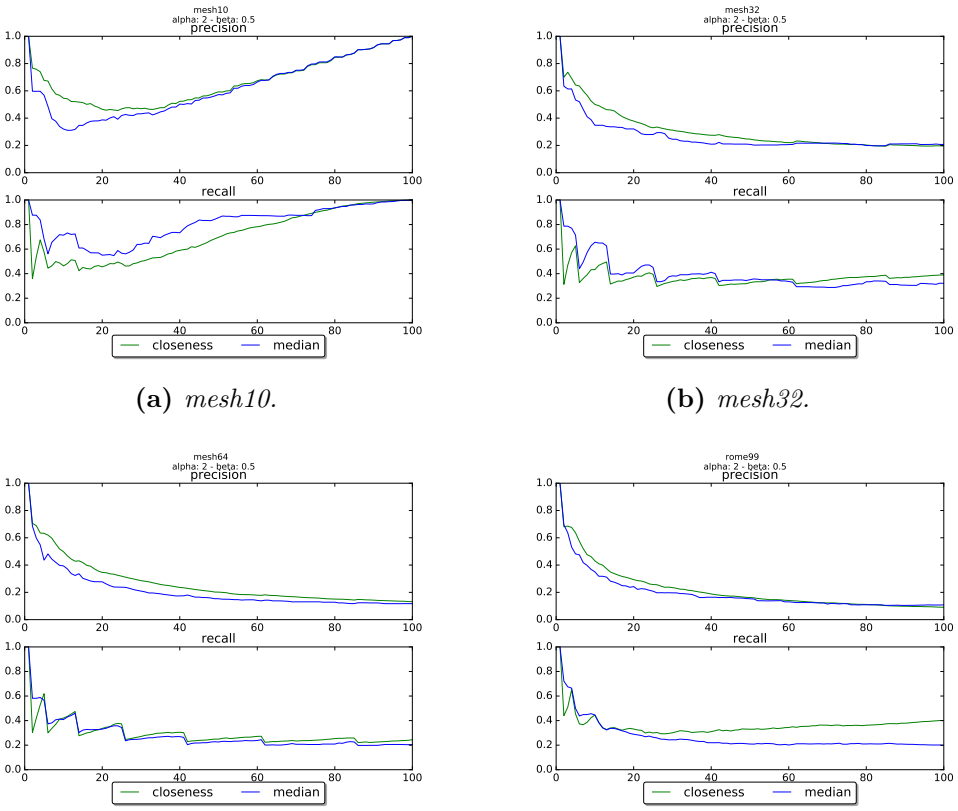


(c) *mesh64.*



(d) *rome99.*

Figura 5.11: *I valori di precision e recall ottenuti dalle mesh utilizzando* $\alpha = 0.5$ *e* $\beta = 0.5$



(a) *mesh10.* **(b)** *mesh32.*

(c) *mesh64.* **(d)** *rome99.*

Figura 5.12: *Le precision e recall ottenute dagli esperimenti sulle 4 mesh utilizzando* $\alpha = 2$ *e* $\beta = 0.5$.

si ripresentano pressoché identici, segno che per le prime posizioni non risentono particolarmente del decremento di α. Essendo che molte query probabilmente chiederanno un numero di vicini non eccessivamente elevato, si evidenzia il fatto che diminuire α sotto 1 non porta incentivi particolari.

Viceversa, l'utilizzo di un α pari a 2 ha portato ad un certo decremento nella precision dei grafi. Questo si può vedere soprattutto nella mesh32 e nel grafo rome99, dove un maggior numero di nodi vengono riconosciuti come positivi dall'α-closeness, con conseguente abbassamento della precision ed aumento del recall. Questo si spiega col fatto per cui maggiore il valore di α e maggiore è la velocità di decrescita dei valori di closeness per i nodi che si trovano più distanti dalla sorgente. Questo fa sì che se l'α è abbastanza grande, i nodi particolarmente lontani ottengono valori di closeness che differiscono tra di loro su piccole cifre decimali. Avanzando nella lista quindi, i valori di closeness dei nodi decrescono più velocemente rispetto ai casi nelle Figure 5.10 e 5.11 dove α è minore, e quindi sono più facilmente catalogati come positivi dall'algoritmo, anche in luce del meccanismo di tolleranza spiegato precedentemente. Inoltre l'appiattimento delle closeness verso valori molto bassi va a produrre maggiore noise nella classifica dei nodi, con conseguente decremento delle prestazioni. Aumentare ulteriormente l'α può quindi risultare pericoloso, almeno per quel che riguarda i k da una certa misura in poi. Ma, come si vedrà più avanti, potrebbe comunque tornare utile per sfruttare la maggiore velocità di calcolo data dall'implementazione delle stopping condition, soprattutto nel caso in cui il numero di vicini desiderato sia contenuto.

Si può notare come, anche modificando il parametro α, se è presente una perturbazione piuttosto forte nel grafo come in questi casi, con media pari a 0.5, l'α-closeness continua ad essere migliore della Median-Distance per i primi valori di k, per poi mantenersi almeno paragonabile.

5.4 Terzo Esperimento

Nel terzo esperimento si mette alla prova la capacità delle stopping condition di velocizzare la ricerca dei valori di α-closeness approssimati. Negli istogrammi riportati si vede come le varie stopping condition, indicate come prima, seconda e terza, abbiano un impatto nei tempi di esecuzione rispetto all'algoritmo che esplora completamente tutti i *possible world*. In questi esperimenti si è inizialmente reso probabilistico il grafo, poi si è misurato il tempo eseguito dall'algoritmo standard per avere una base di riferimento. Dopodiché, per ogni α e per ogni stopping condition sono state eseguite 30 run (dato che in questo esperimento si è interessati al tempo di esecuzione più che ai risultati quali precision e recall, il parametro β si è sempre mantenuto al valore 0.3). Inoltre si è fissato $k = 20$, in quanto era necessario avere una soglia ben specifica per poter verificare le condizioni.

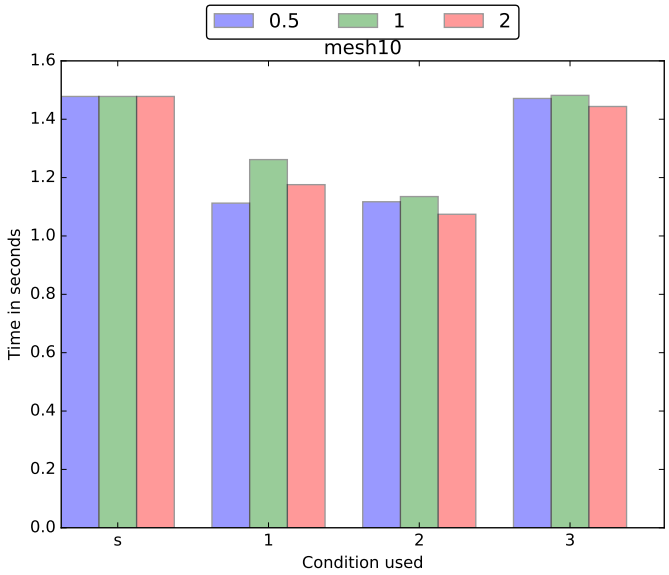


Figura 5.13: *Andamenti dei tempi sulla mesh10.*

Si può notare come in tutti i casi le stopping condition 1 e 2 riescano a portare un abbassamento dei tempi di esecuzione, più o meno apprezzabile. Esso si riscontra soprattutto nelle mesh più grandi, dove il maggior numero di nodi da esplorare e i conseguenti minori valori di α-closeness assunti dai nodi più distanti fanno sì che la seconda parte delle prime due stopping condition, quella che considera i nodi da escludere, sia incontrata più velocemente. Nel caso della mesh10, più piccola, l'impatto è meno significativo, ma riguarda soprattutto il fatto che il numero di iterazioni della parte iterativa dell'algoritmo è limitato e meno determinante rispetto a quello degli altri grafi.

Un'attenzione particolare va data alla condizione numero 3 che, per sua stessa natura, richiede un numero r di sample maggiore rispetto a quelli del semplice algoritmo esaustivo. Nel caso della mesh10 tende quindi perfino ad aumentare leggermente in media i tempi di esecuzione. Lo stesso fenomeno si ripresenta anche sulle altre mesh, almeno per quello che riguarda il valore di α pari ad 1.Per il grafo rome99, l'unico del dataset a non presentare una struttura a mesh è invece curiosamente più veloce, e per il valore $\alpha = 1$ paragonabile alle altre stopping condition.

Si nota come la condizione 2, un lieve miglioramento della condizione 1, riesca, a parità di α, ad abbassare sempre di qualche secondo i tempi di calcolo. Inoltre, osservando i dati grezzi ottenuti dalle esecuzioni degli algoritmi, si è potuto constatare come, almeno per le posizioni più elevate nella classifica, l'incertezza $\tilde{c}_\alpha$ lasciata dai nodi è molto piccola, spesso ancora minore di quella certificata dal-

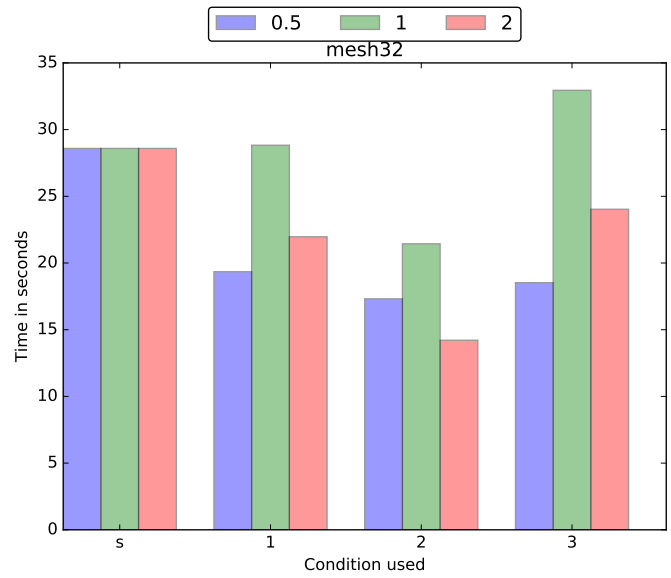


Figura 5.14: *Andamenti dei tempi sulla mesh32.*

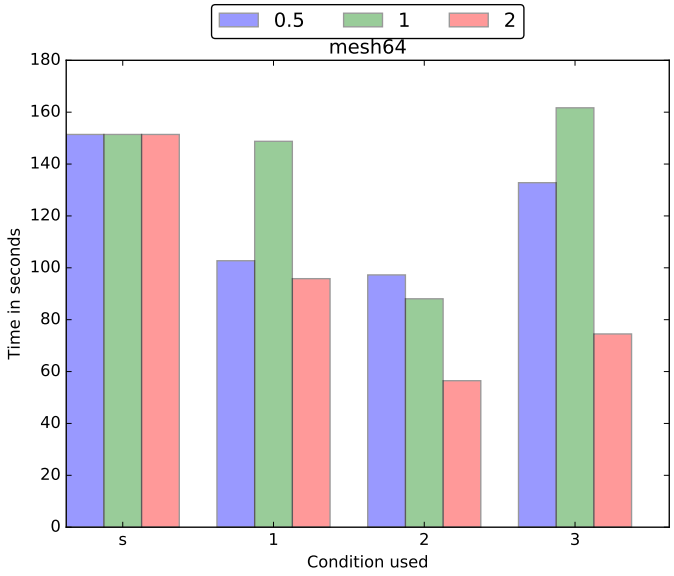


Figura 5.15: *Andamenti dei tempi sulla mesh64.*

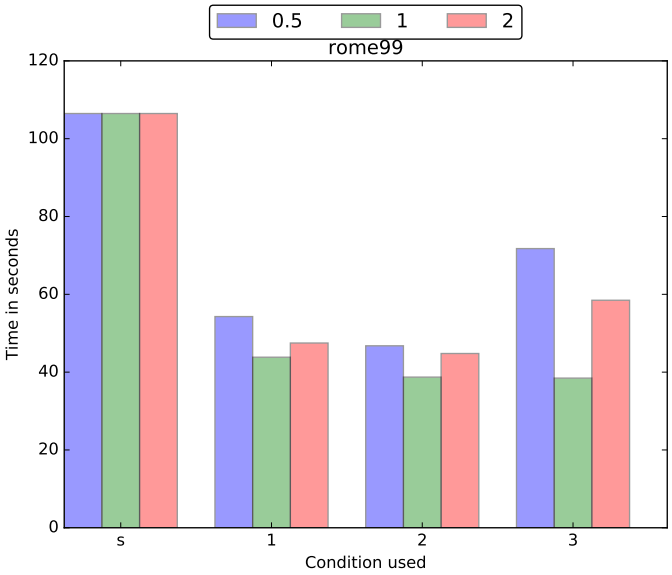


Figura 5.16: *Andamenti dei tempi sul grafo rome99.*

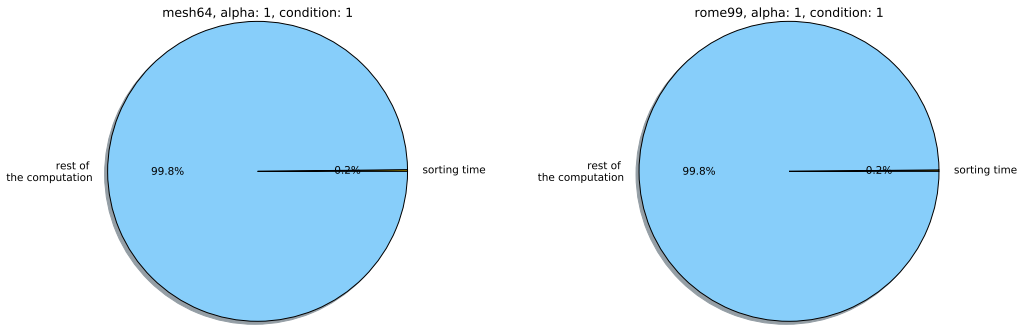
la condizione 3, permettendo un ranking accurato dei nodi nelle prime posizioni, come se fosse stato calcolato dall'algoritmo esaustivo.

È particolarmente curioso inoltre come, mentre per il grafo rome99 i valori di α non sembrino influenzare eccessivamente i tempi di esecuzione, per le mesh più grandi, 32 e 64, invece si abbia un picco nel tempo per $\alpha = 1$. Come si nota, un α pari a 2 riduce in questi casi i tempi di esecuzione e, ricordando i risultati della sezione precedente, permette di ottenere della prestazioni in termini di precision e recall paragonabili almeno per quel che riguarda le prime posizioni, con una loro riduzione all'aumentare però del parametro e di k. Il trade-off che si presenta quindi in questo caso riguarda la possibilità di velocizzare l'esecuzione dell'algoritmo tramite un α più grande a spese di una minore qualità del risultato finale. Nel caso in cui il k sia piccolo (20 o 30 ad esempio) si può procedere nell'utilizzare un α più elevato di 1 per godere dello speed-up consentito dalle stopping condition senza inficiare sul risultato.

Ne emerge che la più utile, almeno sotto queste condizioni, è la stopping condition numero 2, mentre la 1 e la 3 sono più adatte quando il valore α è alto, o quantomeno su grafi più adatti come rome99, permettendo alle volte di raggiungere la stopping condition fin da subito senza nemmeno dover espandere gli *shortest path tree* eccessivamente.

5.4.1 Tempo impiegato nell'ordinamento

Un punto critico nell'implementazione del codice utilizzato per la fase iterativa negli algoritmi che fanno uso di stopping condition è ovviamente l'ordinamento dei nodi, almeno parziale, necessario per rintracciare ogni volta la misura di closeness del k-esimo nodo. In questa tesi si è utilizzato l'algoritmo di sorting offerto dalla standard template library, riordinando tutta la lista di nodi nella sua interezza.



(a) *mesh64.* **(b)** *rome99.*

Figura 5.17: *L'impatto del tempo di sorting sull'algoritmo complessivo nei casi delle mesh più grandi, con* $\alpha = 1$ *e con la condizione 1.*

Vengono riportati in figura 5.17 i grafici che riassumono l'impatto dell'algoritmo di ordinamento sul tempo complessivo totale dell'algoritmo (in media) sulle mesh più grandi utilizzando la condizione 1. Grafici del tutto identici compaiono anche con la condizione 2 (la condizione 3 non richiede in verità un ordinamento). Come si può vedere, in questo caso si tratta veramente di una percentuale insignificante di tempo, dimostrando come il sorting fortunatamente non si stia rivelando un collo di bottiglia in questa computazione.

Capitolo 6

Osservazioni finali e sviluppi futuri

In questa tesi si è affrontato il problema dei k-NN nel contesto degli uncertain graph, comparando le due misure di distanza Median-Distance ed α-closeness. Per eseguire gli esperimenti, si sono presi grafi mesh ed un grafo reale inizialmente deterministici e sono stati perturbati utilizzando una distribuzione esponenziale, variandone la media per osservare le conseguenze. È apparso evidente come, mentre per perturbazioni relativamente contenute le due misure sono pressoché equivalenti, se non per una lieve capacità della Median di riuscire ad individuare un maggior numero di true positive, quando il noise viene aumentato la closeness riesce ad ottenere prestazioni migliori, grazie alla sua capacità di prendere in considerazione la probabilità sui lati in maniera migliore.

Sono state testate inoltre delle stopping condition per velocizzare gli algoritmi che fanno uso di α-closeness, mostrando come queste possano effettivamente portare ad una riduzione dei tempi tanto più determinante quanto più alto è il valore α. Nasce quindi la possibilità di un trade-off tra la velocità di esecuzione che si vuole raggiungere e la qualità nei risultati, che degrada velocemente all'aumentare di α, soprattutto quando si sale con i valori k. Nel caso quindi in cui si desiderino solo i primi vicini, si possono usare con grande vantaggio le stopping condition aumentando l'α.

Come si può vedere, nell'algoritmo 3, gli shortest path tree vengono fatti crescere di esattamente 1 nodo in ogni iterazione per semplicità. Ovviamente, su questa parte si può lavorare con delle euristiche, ad esempio scegliendo di aumentare suddetti alberi ad ogni iterazione di un certo numero variabile di nodi, una frazione di n ad esempio. Inoltre, come accennato precedentemente, si potrebbe preferire espandere solamente i *most promising trees*, piuttosto che tutti gli alberi insieme.

Non si è definito una volta per tutte cosa si intenda con *most promising* tree, e quindi in questo frangente c'è la possibilità di testare delle euristiche. Ad esempio, ogni albero ha una istanza di coda dove compare il nodo da visitare successivamente e quindi si può considerare come *most promising tree* quello che presenta come nodo da visitare come successivo quello che promette di ridurre maggiormente l'incertezza relativa al nodo in questione. La logica dietro tutto questo sarebbe quella di ridurre il più velocemente l'incertezza sui nodi che più verosimilmente appariranno in cima alla lista dei vicini nella soluzione finale soddisfacendo prima la stopping condition.

Algoritmi

Algorithm 1: ϵ-approximation of K-NN

input : Uncertain Graph $\mathcal{G}$,
source vertex s,
parameters k, δ, ε

$r \leftarrow \frac{2}{\epsilon^2} ln(\frac{2n}{\delta})$
foreach $v \in V$ **do**
  $\hat{c}_\alpha(s, v) \leftarrow 0$
for $i \leftarrow 1$ **to** r **do**
  sample a possible world G_i from $\mathcal{G}$
  foreach *vertex* $v \in V$ **do**
    $dist[v] \leftarrow \infty$
  $Q \leftarrow \emptyset$ `// a FIFO queue`
  $Q.enqueue(s)$
  $dist[s] \leftarrow 0$
  while $Q \neq \emptyset$ **do**
    $u \leftarrow Q.dequeue()$
    foreach *neighbour* v *of* u **do**
      if $dist[v] = \infty$ **then**
        $dist[v] \leftarrow dist[u] + 1$
        $Q.enqueue(v)$
  foreach $v \in V$ **do**
    $\hat{c}_\alpha(s, v) \leftarrow \hat{c}_\alpha(s, v) + \frac{1}{r} \frac{1}{(dist[v])^\alpha}$
sort the nodes accordingly to their $\hat{c}_\alpha(s, v)$. Be $\hat{v}_1, \ldots, \hat{v}_{n-1}$ the labeling of such ordering
for $i \leftarrow k$ **to** k **do**
  $W \leftarrow W \cup \{\hat{v}_i\}$
$i \leftarrow i + 1$
while $\hat{c}_\alpha(s, \hat{v}_k) = \hat{c}_\alpha(s, \hat{v}_i)$ **do**
  $W \leftarrow W \cup \{\hat{v}_i\}$
  $i \leftarrow i + 1$
return W

Algorithm 2: stopped ϵ-approximation of K-NN: initialization phase

input : $\mathcal{G}$, uncertain graph
k, the number of nearest neighbours
s, source node
r, integer

```
1  foreach v ∈ V do
2      c̄_α(s,v), c̃_α(s,v) ← 0
3  for i ← 1 to r do
4      *sample one possible world G_i of 𝒢*
5      foreach v ∈ V do
6          dist_i[v] ← ∞
7      dist_i[s], D_{G_i}, counter ← 0
8      Q_i, T_i ← ∅
9      Q_i.enqueue(s)
10     while (Q_i ≠ ∅) and (counter < k) do
11         u ← Q_i.dequeue()
12         counter ← counter + 1
13         T_i ← T_i ∪ {u}
14         if dist_i[u] > D_{G_i} then
               // have to update D_{G_i}
15             D_{G_i} ← dist_i[u]
16         foreach neighbour v of u do
17             if dist_i[v] = ∞ then
18                 dist_i[v] ← dist_i[u] + 1
19                 Q_i.enqueue(v)
20     while dist_i[Q_i.front()] = D_{G_i} do
21         u ← Q_i.dequeue()
22         T_i ← T_i ∪ {u}
23         foreach neighbour v of u do
24             if dist_i[v] = ∞ then
25                 dist_i[v] ← dist_i[u] + 1
26                 Q_i.enqueue(v)
27     foreach v ∈ T_i do
28         c̄_α(s,v) ← c̄_α(s,v) + (1/r) 1/(dist_i[v])^α
29     foreach v ∉ T_i do
30         c̃_α(s,v) ← c̃_α(s,v) + (1/r) 1/(D_{G_i})^α
31 return iteration_phase(r, G, D_G, T, Q, dist)
```

Algorithm 3: stopped ϵ-approximation of K-NN: iteration phase

input : r, number of samples
G, possible worlds of $\mathcal{G}$
$\mathbf{D_G}$ for each of these possible worlds
T, sets of nodes already visited for each G_i
Q, the priority queues used in the initialization phase
dist, arrays of distances of the nodes in the various possible worlds

Let $\bar{v}_1, \ldots, \bar{v}_{n-1}$ the ordering of the nodes based on their $\bar{c}_\alpha(s, v)$ and be $\bar{c}_\alpha^{(k)} = \bar{c}_\alpha(s, \bar{v}_k)$
while *stopping condition is not met* **do**
  for $i \leftarrow 1$ **to** r **do**
    if $Q_i \neq \emptyset$ **then**
      `// grow the tree` T_{G_i}
      $u \leftarrow Q_i.dequeue()$
      $\bar{c}_\alpha(s, u) \leftarrow \bar{c}_\alpha(s, u) + \frac{1}{r}\frac{1}{(dist_i[u])^\alpha}$
      $\tilde{c}_\alpha(s, u) \leftarrow \tilde{c}_\alpha(s, u) - \frac{1}{r}\frac{1}{(D_{G_i})^\alpha}$
      $T_i \leftarrow T_i \cup \{u\}$
      foreach *neighbor* v *of* u **do**
        if $dist_i[v] = \infty$ **then**
          $dist_i[v] \leftarrow dist_i[u] + 1$
          $Q.enqueue(v)$
      if $dist_i[u] > D_{G_i}$ **then**
        $oldD_{G_i} \leftarrow D_{G_i}$
        $D_{G_i} \leftarrow dist_i[u]$
        foreach $v \notin T_i$ **do**
          $\tilde{c}_\alpha(s, v) \leftarrow \tilde{c}_\alpha(s, v) - \frac{1}{r}\frac{1}{(oldD_{G_i})^\alpha} + \frac{1}{r}\frac{1}{(D_{G_i})^\alpha}$
  sort the nodes accordingly to their $\bar{c}_\alpha(s, v)$ in decreasing order and update $\bar{c}_\alpha^{(k)}$, both if necessary
return W' $= \{(v, \bar{c}_\alpha(s, v)) : v \in V \backslash \{s\}, \bar{c}_\alpha(s, v) \geq \bar{c}_\alpha^{(k)}\}$

Bibliografia

[1] Eytan Adar and Christopher Re. Managing uncertainty in social networks. *IEEE Data Eng. Bull.*, 30(2):15–22, 2007.

[2] Saurabh Asthana, Oliver D King, Francis D Gibbons, and Frederick P Roth. Predicting protein complex membership using probabilistic network reliability. *Genome research*, 14(6):1170–1175, 2004.

[3] Francesco Bonchi, Francesco Gullo, Andreas Kaltenbrunner, and Yana Volkovich. Core decomposition of uncertain graphs. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1316–1325. ACM, 2014.

[4] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.

[5] Nilesh Dalvi and Dan Suciu. Efficient query evaluation on probabilistic databases. *The VLDB Journal*, 16(4):523–544, 2007.

[6] Joy Ghosh, Hung Q Ngo, Seokhoon Yoon, and Chunming Qiao. On a routing problem within probabilistic graphs and its application to intermittently connected networks. In *IEEE INFOCOM 2007-26th IEEE International Conference on Computer Communications*, pages 1721–1729. IEEE, 2007.

[7] Ruoming Jin, Lin Liu, Bolin Ding, and Haixun Wang. Distance-constraint reachability computation in uncertain graphs. *Proceedings of the VLDB Endowment*, 4(9):551–562, 2011.

[8] George Kollios, Michalis Potamias, and Evimaria Terzi. Clustering large probabilistic graphs. *IEEE Transactions on Knowledge and Data Engineering*, 25(2):325–336, 2013.

[9] Nevan J Krogan, Gerard Cagney, Haiyuan Yu, Gouqing Zhong, Xinghua Guo, Alexandr Ignatchenko, Joyce Li, Shuye Pu, Nira Datta, Aaron P Tikuisis,

et al. Global landscape of protein complexes in the yeast saccharomyces cerevisiae. *Nature*, 440(7084):637–643, 2006.

[10] Panos Parchas, Francesco Gullo, Dimitris Papadias, and Franceseco Bonchi. The pursuit of a good possible world: extracting representative instances of uncertain graphs. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 967–978. ACM, 2014.

[11] Michalis Potamias, Francesco Bonchi, Aristides Gionis, and George Kollios. K-nearest neighbors in uncertain graphs. *Proceedings of the VLDB Endowment*, 3(1-2):997–1008, 2010.

[12] Petteri Sevon, Lauri Eronen, Petteri Hintsanen, Kimmo Kulovesi, and Hannu Toivonen. Link discovery in graphs derived from biological databases. In *International Workshop on Data Integration in the Life Sciences*, pages 35–49. Springer Berlin Heidelberg, 2006.