UNIVERSITÀ DI PISA

Facoltà di Scienze Matematiche, Fisiche e Naturali
Facoltà di Economia
Corso di Laurea Specialistica in
Informatica per l'Economia e l'Azienda

TESI DI LAUREA

Tecniche di *graph mining* per l'analisi di reti sociali

Candidato
Salvatore Corvasce

Relatori:
Dino Pedreschi
Ernesto Lastres
Michele Coscia

Controrelatore:
Fosca Giannotti

Anno Accademico 2008/09

Alla mia famiglia

Introduzione

Negli ultimi anni, il tema dell'innovazione, soprattutto dell'information technology, denominato IT, è diventato dominante nello scenario mondiale, nelle grandi e medie organizzazioni.

Oggi, le aziende dispongono di un'ingente quantità di dati, ad un livello di dettaglio e con qualità variabile, la cui natura è strettamente connessa all'attività gestionale o altre forme di attività. L'analisi di questi dati viene eseguita attraverso metodi, appartenenti alla statistica classica, che lasciano il posto alle esigenze di interpretazione dei risultati.

I dati considerati in questo lavoro sono rappresentati da entità come ad esempio persone. Ad esempio, si possono considerare, a livello mondiale, tutte le persone coinvolte in attività illecite, collegate tra loro da relazioni: amicizia, traffico di denaro, raccolta del pizzo ecc…. Inoltre, si possono analizzare questi dati eseguendo delle analisi utili al GIP (Giudice istruttore penale). Da questi dati, ad esempio, si potrebbe esplorare e analizzare la figura del criminale di spicco oppure si potrebbe selezionare un gruppo di persone di maggiore importanza. Oltre l'entità persona, si potrebbero considerare altri tipi di entità come: telefoni cellulari, luoghi, organizzazioni e dispositivi di comunicazione.

In questo scenario il Data Mining (DM) è lo strumento per l'analisi dei dati: un'attività di dati semi-automatica, esplorativa, basata sull'apprendimento e finalizzata all'estrazione di conoscenza sotto forma di modelli utili ai decisori per formulare ipotesi di azioni [Albano 06].

Il percorso evolutivo di questo lavoro inizia con lo studio del problema e la ricerca di modelli, come nell'esempio precedente, di un insieme di persone, che sono rappresentate da una *rete sociale*. Con il termine rete, s'intende un insieme di sistemi o entità interconnessi tra loro. L'*analisi delle reti sociali* prende in considerazione questo tipo di struttura per raccogliere informazioni utili. Discuteremo in dettaglio nel capitolo successivo di reti sociali e le analisi delle reti sociali.

Con questo lavoro tenteremo, inoltre, in fase del tutto sperimentale, di costruire un indice di centralità (ad esempio persone più importanti) e di individuare un pattern all'interno della rete sociale (ad esempio un gruppo di persone). La struttura dati utilizzata è una rete rappresentata da un grafo. Quindi, parleremo di GM, un caso speciale della struttura dati del DM. In particolare discuteremo del *link mining* LM il quale analizza i collegamenti tra entità.

1 Stato dell'arte

In questo capitolo prenderemo in considerazione le reti sociali e descriveremo come sono analizzate e rappresentate (grafo). Parleremo del DM, del GM e delle diverse misure/metriche che esistono in letteratura. Infine, parleremo della complessità nell'affrontare questo tipo di struttura con quantità di dati da analizzare elevata.

1.1 Social Network

L'analisi delle reti sociali è una metodologia con sempre maggiori applicazioni all'interno della scienza sociale. Essa è applicata ad aree tanto diverse: medicina, psicologia, organizzazioni, business, comunicazioni, elettronica e criminologia (terrorismo). Il suo uso è ancora relativamente recente. Con tale metodologia si cerca di creare delle linee guida, che organizzano e gestiscono lo strumento per renderlo più accessibile ad operatori del settore. Infatti, come mostra la Figura 1.1 [Borgatti 03], su questo argomento, la crescita della letteratura, aumenta esponenzialmente.

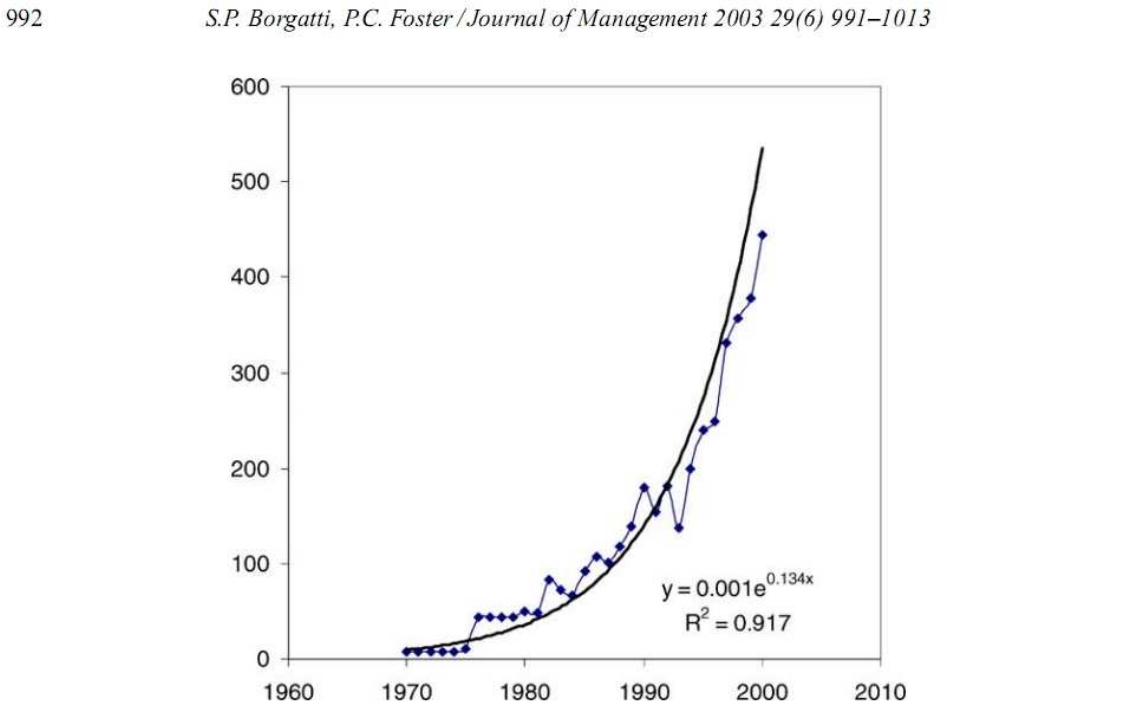


Figure 1. Exponential growth of publications indexed by Sociological Abstracts containing "social network" in the abstract or title.

Figura 1.1 grafo della letteratura delle reti sociali

L'analisi delle reti sociali ha due obiettivi principali: gli attori e le relazioni tra di loro nello specifico contesto sociale. Questi obiettivi aiutano a comprendere come la posizione dell'attore nella rete, influenza il loro accesso alle risorse costituite dal capitale d'informazione.

Nell'ambito economico l'analisi delle reti sociali non rappresenta una "soluzione" dei problemi. Tuttavia, essa fornisce una chiave diagnostica delle componenti del progetto, della gestione di iniziative e del cambio di programmi, che coinvolgono un elemento sociale di comunicazione collaborazione o coordinazione.

INSNA "International Network for Social Network Analysis" è un'associazione professionale di ricercatori interessati all'analisi delle reti sociali. Si tratta di un'organizzazione no-profit fondata nel 1978 da Barry Wellman.

È possibile distinguere la rete sociale dal punto di vista della *sociologia*, *matematico-fisico* e *data mining*. Prima di introdurre il concetto della rete sociale nei vari campi, è bene introdurre dei concetti basilari di tale strumento.

1.1.1 Introduzione al grafo

Prima di parlare del grafo definiamo la terminologia riferita da [Newman 03] con una definizione di vertice, arco, arco diretto o indiretto.

Vertice: è l'unità fondamentale della rete, chiamato anche: area (fisica), nodo (informatica), o attore (sociologia).

Archi: è la linea che connette due vertici, chiamata anche unione (fisica), collegamento (informatica), o legame (sociologia).

Diretto/Indiretto: un lato è diretto se si muove in una sola direzione: un verso della strada tra due punti; indiretto se si muove in entrambe le direzioni. I lati diretti, sono a volte chiamati *archi*. Un grafo è diretto se tutti i suoi archi sono diretti. Un grafo indiretto può essere rappresentato da uno diretto, avendo due archi tra ogni coppia di vertici connessi: uno per ogni direzione.

La struttura più utilizzata dalla rete sociale è il *grafo*. Tale struttura consente una visualizzazione di facile e semplice lettura (anche da chi non è esperto di conoscenze matematiche e statistiche). Nella Figura 1.2 sottostante, mostriamo un grafo semplice, i cui nodi sono collegati con archi senza frecce e ciascuno a sua volta è connesso ad un altro nodo.

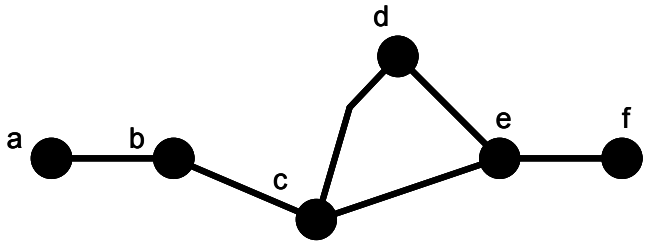


Figura 1.2 esempio di grafo con nodi etichettati

Più formalmente un grafo è definito come segue. Dato un nodo etichettato L_V e un arco etichettato L_E, definiamo un grafo *g* di quattro tuple *g* = (*V*, *E*, μ, ν), dove:

- *V* denota un insieme finito di *nodi*.
- $E \subseteq V \, x \, V$ denota un insieme di *archi*.
- $\mu : V \rightarrow L_V$ denota una funzione di *etichetta* del nodo.
- $\nu : E \rightarrow L_E$ denota una funzione di *etichetta* degli archi.

Mentre V definisce i nodi, l'insieme degli archi *E* rappresenta la struttura del grafo. Questo è, un nodo $u \in V$ è connesso a un nodo $v \in V$ da un arco $e = (u, v)$ è $(u, v) \in E$. La funzione etichetta può essere usata per integrare informazioni di nodi e archi all'interno del grafo, assegnando attributi a L_V e L_E cioè rispettivamente a nodi e archi.

Il grafo può essere *orientato* o *non orientato*. Il grafo del primo tipo è costituito da archi con direzione: dal nodo A al nodo B (quindi l'arco ha una sola freccia). Il grafo del secondo tipo non ha direzioni (quindi il nodo da A a B e il nodo da B a A hanno un arco senza freccia). Nella discussione del nostro lavoro parleremo dei grafi orientati.

Il grafo può avere più relazioni. In tal caso si parla di un multigrafo.

Un multigrafo è una coppia (*V*, *E*), dove *V* e *E* sono due insiemi disgiunti, con il dato di una mappa $E \rightarrow V \cup [V]^2$, che associa a ogni lato $e \in E$ uno, oppure due vertici in *V*, detti le estremità di *e*. In altri termini, un multigrafo è un grafo diretto, privato delle informazioni sulle orientazioni dei lati.

Un sottografo è definito come: sia $g_1=(V_1,\ E_1,\ \mu_1,\ \nu_1)$ e $g_2=(V_2,\ E_2,\ \mu_2,\ \nu_2)$ grafi. Il grafo g_1 è *sottografo* di g_2, scritto $g_1 \subseteq g_2$, se:

- $V_1 \subseteq V_2$.
- $E_1 = E_2 \cap (V_1 \text{ x } V_1)$.
- $\mu_1(u) = \mu_2(u)$ per tutti gli $u \in V_1$.
- $\nu_1(u, v) = \nu_2(u, v)$ per tutti gli $(u, v) \in E_1$.

Al contrario, il grafo g_2 è chiamato *supergrafo* di g_1 se è un sottografo di g_2. La notazione di sottografo viene usata per approcci a problemi più complessi come grandi parti comuni di diversi grafi. Il sottografo è molto utile per la ricerca di patterns.

Un grafo isomorfo è definito: sia $g_1=(V_1,\ E_1,\ \mu_1,\ \nu_1)$ e $g_2=(V_2,\ E_2,\ \mu_2,\ \nu_2)$ grafi. Un isomorfismo di grafo tra g_1 e g_2, è una funzione bigettiva che soddisfa $f : V_1 \to V_2$:

- $\mu_1(u) = \mu_2(f(u))$ per tutti i nodi $u \in V_1$.
- Per tutti gli archi $e_1=(u, v) \in E_1$, esiste un arco $e_2=(f(u), f(v)) \in E_2$ tale che $\nu_1(e_1) = \nu_2(e_2)$.
- Per tutti gli archi $e_2=(u, v) \in E_2$, esiste un arco $e_1=(f^{-1}(u), f^{-1}(v)) \in E_1$ tale che $\nu_1(e_1) = \nu_2(e_2)$.

Due grafi g_1 e g_2 sono chiamati isomorfi se esiste un isomorfismo di grafo tra loro.

Dall'ultima definizione deduciamo che grafi isomorfi sono identici in termini di struttura ed etichette. Per stabilire un isomorfismo, occorre mappare ogni nodo dal primo grafo ai nodi del secondo grafo. Ciò in modo tale che la struttura degli archi sia conservata e i nodi e le etichette siano consistenti. In Figura 1.3 è mostrato un'illustrazione di due grafi isomorfi senza etichette.

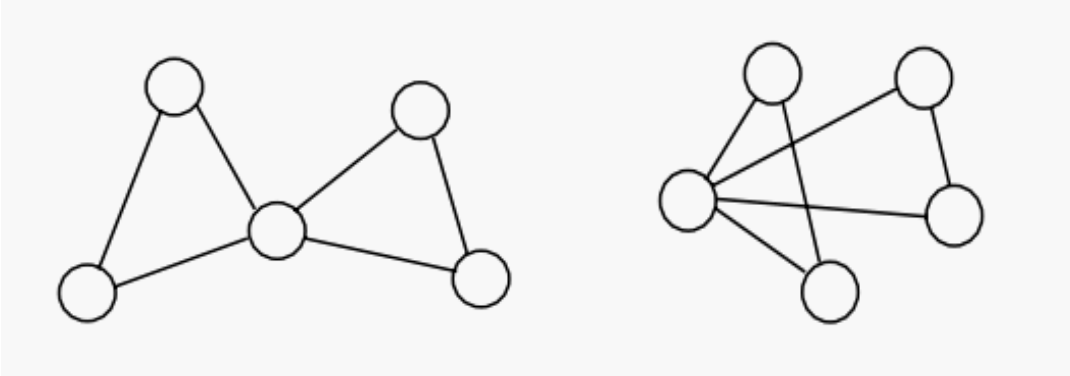


Figura 1.3 grafi isomorfi

L'approccio più semplice, per verificare l'isomorfismo tra due grafi, è quello di attraversarlo completamente, considerando un albero di ricerca, tutte le possibili corrispondenze da nodo a nodo. Se i nodi e gli archi sono dotati di etichetta, il matching di nodi e di archi deve anche essere consistente in termini delle loro etichette. Il raggiungimento di un nodo - foglia nell'albero di ricerca equivale ad avere trovato un matching: un grafo isomorfo. In generale, la complessità dell'esecuzione di questa procedura è esponenziale al numero dei nodi di entrambi i grafi.

Relativo all'isomorfismo di grafi è il problema di rilevare se un grafo più piccolo è presente in un grafo più grande. Se l'isomorfismo è costituito da un grafo uguaglianza, sottografo isomorfo può essere visto come un'uguaglianza di un sottografo.

Un sottografo isomorfo è definito come: sia $g_1=(V_1, E_1, \mu_1, \nu_1)$ e $g_2=(V_2, E_2, \mu_2, \nu_2)$ grafi. Una funzione ingettiva $f : V_1 \to V_2$ è chiamata sottografo isomorfismo da g_1 a g_2, se esiste un sottografo $g \subseteq g_2$ tale che f è un grafo isomorfo tra g_1 e g.

Un sottografo isomorfo esiste da g_1 a g_2 se il grafo più grande g_2 può trasformarsi ad un grafo che è isomorfo al grafo più piccolo g_1 rimuovendo alcuni nodi e archi. Il sottografo isomorfo può essere determinato con la procedura descritta sopra per grafi isomorfi. È noto che i sottografi isomorfi rientrano nella classe dei problemi NP-complete. In Figura 1.4 è mostrato un'illustrazione di un sottografo isomorfo senza etichette.

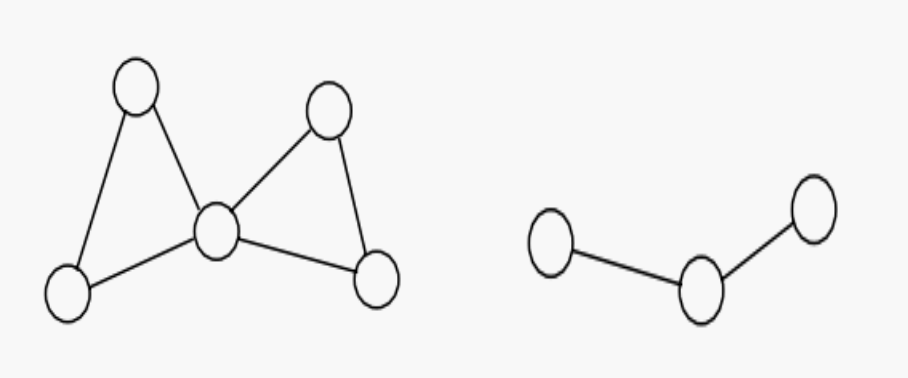


Figura 1.4 grafo-sottografo

1.1.1.1 Misure nell'analisi delle reti sociali

Esistono diverse metriche e indici che analizzano la rete (il grafo), queste misure servono appunto per l'analisi della rete anche le reti sociali.

Una rete sociale è un grafo rappresentato da una *matrice di adiacenza* quadrata *n x n*, dove n è il numero degli attori che costituiscono il network in esame. È l'usuale modalità di rappresentazione di una relazione fra attori. Per convenzione, le righe (*i*) contengono gli attori da cui la relazione "parte" e le colonne (*j*) gli attori a cui la relazione è rivolta. Una relazione è simmetrica quando $X_{ij} = X_{ji}$. Normalmente la diagonale principale della matrice è vuota e non è resa in conto nelle computazioni. Inoltre, per rappresentare di una rete sociale è possibile usare una *sociomatrice*. Una sociomatrice è costituita da una tabella alle cui righe corrispondono gli attori "emittenti" ed alle colonne gli attori "riceventi". Le entrate della sociomatrice registrano quali nodi sono adiacenti tra loro: se tra il vertice n_i e il vertice n_j esiste un arco, allora nella sociomatrice ci sarà un 1 nella cella (*i*, *j*); viceversa, avremo nella stessa cella uno *0*. Quindi, se i nodi n_i e n_j sono adiacenti, allora $x_{ij} = 1$, se non lo sono, allora $x_{ij} = 0$. La sociomatrice per una relazione dicotomica (dove si rileva solo la presenza o l'assenza di una relazione) corrisponde esattamente alla matrice di adiacenza.

Le *rappresentazioni grafiche* sono uno strumento particolarmente potente di analisi. Esso è basato sulle "proprietà intuitive" delle immagini, opportunamente guidate e migliorate dall'applicazione di misure statistiche in fase di scelta dei dati (nel caso di reti complesse) e dei layout (disposizioni dei nodi e dei legami nello spazio).

Una modalità sofisticata di rappresentazione è il *MDS* (Multi Dimensional Scaling), che dispone la rete nello spazio X-Y in modo che le loro distanze esprimano quanto i nodi siano fra loro "simili" o meno. In questo caso, le distanze assumono un significato, che richiede, comunque, di essere accuratamente interpretato.

A volte, può essere utile semplificare volontariamente la struttura della rete, in modo da renderla più facilmente interpretabile in via grafica. Ad esempio, è possibile scegliere di eliminare i nodi "isolati" (quelli privi di legami) ed i nodi "pendenti" (quelli uniti alla rete da un solo legame). Inoltre, possono essere visualizzate in modo specifico parti di rete che godono della proprietà di essere "sub-gruppi", ovvero possono presentare nodi dotati di caratteristiche simili in termini di struttura (componenti, blocchi, k-cores, fazioni, cluster).

Infine, è sempre possibile rappresentare una rete dal punto di vista dei singoli attori (in termini delle diverse ego-network che la costituiscono). Questi ultimi sono più semplici del network complessivo, soprattutto se questo si presenta molto ricco di attori e legami.

Non esiste a priori un modo "giusto" di rappresentare un network: bisogna provare diverse configurazioni, avendo ben chiaro cosa si vuole mettere in evidenza.

Normalmente, come si è visto, una rete è costituita da relazioni fra attori, ognuno dei quali dotato di attributi. Si può anche immaginare di "rovesciare" la rete, ridefinendo le relazioni a partire dal rapporto fra gli attributi degli attori. Se, ad esempio, due attori condividono lo stesso attributo, per esempio, hanno entrambi sex="M", la relazione fra essi assume il valore 1 o altrimenti 0. Esistono differenti funzioni di confronto fra attributi (somma, prodotto, differenza, differenza assoluta, ecc.), che vanno scelte in ragione del senso dell'analisi.

Gli indici che analizziamo sono: gradi, distanza geodetica, diametro, densità, componente.

Gradi: questi indici sono rappresentati dal numero di archi connesso al vertice. Nota che i gradi non sono necessariamente uguali al numero dei vertici adiacenti. In articoli recenti, il termine grado è riferito alla "connettività" di un vertice. Ma noi evitiamo di usare questo termine, perché connettività, nella teoria dei grafi, già ha un altro significato. Un grafo diretto ha entrambi i casi (in-gradi e out-gradi) per ogni vertice, i quali rappresentano rispettivamente i numeri di arrivo e di uscita degli archi.

Distanza geodetica: una distanza geodetica è il più breve percorso attraverso la rete da un vertice ad un altro. Nota che ci potrebbero spesso essere più di una distanza geodetica tra due vertici.

Diametro: il diametro di una rete è la lunghezza (in numero di archi) della più lunga distanza geodetica tra due qualsiasi vertici. Alcuni autori hanno usato anche questo termine per evidenziare la *media* della distanza geodetica nel grafo, sebbene strettamente le due quantità siano abbastanza distinte.

Densità: Nel caso di una rete a legame binario la densità è il rapporto fra il numero dei legami effettivi ed il numero delle diadi (coppie), ovvero di tutti i possibili legami diadici teoricamente esistenti. Se i legami sono espressi con variabili scalari (come la forza o la probabilità della relazione), la densità è espressa come rapporto fra la somma dei valori di tutti i legami e il numero dei legami possibili ovvero è la "forza media" dei legami teorici.

Componente: la componente, del quale un vertice fa parte è l'insieme di vertici che può essere raggiunto da esso da un percorso attraverso gli archi del grafo. In un grafo diretto un vertice

ha entrambi un in-componente e un out-componente, i quali rappresentano l'insieme dei vertici dal quale il vertice può raggiungere e essere raggiunto da esso.

Altre misure possono essere: raggiungibilità, connettività, cliques, n-cliques, n-clans, k-plexes, k-cores, blocchi e cutpoints, lambda set.

Raggiungibilità: questa misura indica se un vertice è raggiungibile da un altro, indipendentemente dalla lunghezza del percorso e tenendo conto dell'orientamento dei legami. Naturalmente, in caso di matrici non simmetriche si può dare il caso che "A" sia raggiungibile da "B", ma non il contrario. Sociologicamente, la raggiungibilità, che è una proprietà di connessione, va interpretata in ragione del significato del tipo di relazione che si osserva, in termini di potere.

Connettività: questa è una misura complementare della raggiungibilità. Essa esprime il numero di nodi che dovrebbe essere cancellato dalla rete per disconnettere due attori. In generale, maggiore è il numero di nodi da cancellare, maggiore sarà la possibilità dell'attore di ricevere/trasmettere informazioni; operare scambi, ecc., soprattutto se i nodi sono dati dalla sommatoria di diversi percorsi.

Cliques: una clique è un insieme di attori fra loro connessi in modo più "stretto" di quanto lo sia in generale il network complessivo di cui essi fanno parte. In termini più formali, una clique è il massimo numero di attori che hanno tutti i possibili legami, che possono teoricamente esistere fra loro stessi (un sottografo completo e massimo). Tale vincolo, piuttosto rigido, è, di solito, affrontato, dal punto di vista del calcolo, simmetrizzando preventivamente, la matrice di adiacenza, riferendosi ad un concetto di relazione "presente", al di là del suo orientamento.

N-cliques: rappresentano una generalizzazione delle cliques. Ciò avviene, attraverso un percorso di lunghezza massima N, sostituendo al vincolo forte del sottografo completo e massimo, l'esistenza di una relazione fra tutti gli attori. Di solito, in sociologia, si usano N-cliques con N posto a 2, ottenendo oggetti del tipo "amici degli amici". Visto che rilassiamo il vincolo della lunghezza del legame, può essere utile esplorare diverse ipotesi di cliques, agendo in aumento sul vincolo del numero minimo di membri.

N-clans: costituiscono una restrizione di N-cliques, attraverso il vincolo che il diametro sia minore o eguale a N. Corregge un difetto di N-cliques, quello di formare dei gruppi spuri, ovvero composti "casualmente" da membri appartenenti ognuno a gruppi fra loro differenti.

K-plexes: Un altro approccio rivolto a rendere meno forte il vincolo originario delle cliques è quello di accettare come membro del gruppo ogni nodo che abbia almeno n – k legami con gli altri nodi, dove n è il numero complessivo dei nodi che costituiscono il gruppo. Altrimenti detto, il nodo A fa parte di un 2-plex, costituito dai nodi B, C e D quando p.e. ha un legame sia con B che

con C, ma non con D, essendo D a sua volta, legato sia a B che a C. K-plexes genera ovviamente molti più gruppi di minore dimensione rispetto ai metodi precedenti.

K-cores: È costituito da tutti gli attori connessi con almeno K altri attori nell'ambito del medesimo gruppo o, se si preferisce, è un sottografo completo il cui grado sia eguale o maggiore di K.

Blocchi e cutpoints: un altro approccio top/down è rivolto alla ricerca dei cutpoints, cioè dei nodi che se rimossi, portano alla creazione di sottografi disconnessi, detti "blocchi" o "bi-componenti". Normalmente, questa misura è preceduta, per i networks orientati, dalla simmetrizzazione dei legami.

Dal punto di vista sociologico, questa misura esprime, nei suoi limiti, la vulnerabilità della rete.

Lambda sets: in alternativa alla ricerca dei cutpoints, è possibile cercare i legami "ponte", cioè quelli che, una volta interrotti, portano alla disconnessione del network in sottografi. Una modalità più sofisticata di analisi è rivolta alla valutazione della riduzione del flusso massimo che deriverebbe dallo scioglimento di ogni legame, in modo da determinare quelli maggiormente "importanti" rispetto alla capacità di comunicazione complessiva della rete. Anche in questo caso, normalmente, il network in esame è precedentemente simmetrizzato. Il risultato, denominato lambda set, esprime per ogni legame il grado di minima connettività interna, ed è rappresentato come cluster analysis gerarchica.

Eigenvector: il significato di questa misura in ogni caso risiede nel considerare l'importanza (ovvero la qualità) del vicinato. In questo modo, un vertice è tanto più centrale quanto più è collegato a vertici centrali: una misura indubbiamente adatta a valutare la posizione del soggetto per gruppi di importanza nel grafo (è un fatto che i valori più alti di eigenvector vengono raggiunti dai vertici inseriti nelle cliques più ampie, ovvero da quei vertici che sono collegati a molti altri, che a loro volta sono collegati a molti altri...).

Non possiamo dire quale vertice sia più centrale nella rete, fino a che non specifichiamo quale concetto di centralità intendiamo usare. La scelta del tipo di centralità da usare è dettata dalla tipologia della ricerca. Quindi, per ogni concetto esiste un indice adeguato.

1.1.2 SN: sociologia

Considerando, nell'ambito della sociologia, una rete sociale. Dice Boissevain J. che "l'uomo è un essere sociale interagente, capace di manipolare gli altri, così come di essere manipolato da loro [...]. Il postulato fondamentale dell'approccio della rete sta nel fatto che le persone sono viste come interagenti con altre persone, alcune delle quali, a loro volta, interagiscono fra di loro a più livelli, di modo che il network totale di relazioni, che così si forma, è in uno stato di fluidità" [Boissevain 73].

Una rete sociale è in un insieme di *entità* e *relazioni* definite su di essa. Gli attori possono essere individui discreti e collettivi, oppure possono essere collezioni di unità sociali, anche chiamati nodi. Le relazioni, invece, costituiscono un insieme di archi di un tipo specifico tra membri di un gruppo.

La struttura dati utilizzata nella rete sociale è quella tramite matrice o grafico. Noi utilizzeremo entrambi i metodi a seconda del problema affrontato. Qui, in Figura 1.5, è mostrato un grafo della Tabella like - not like.

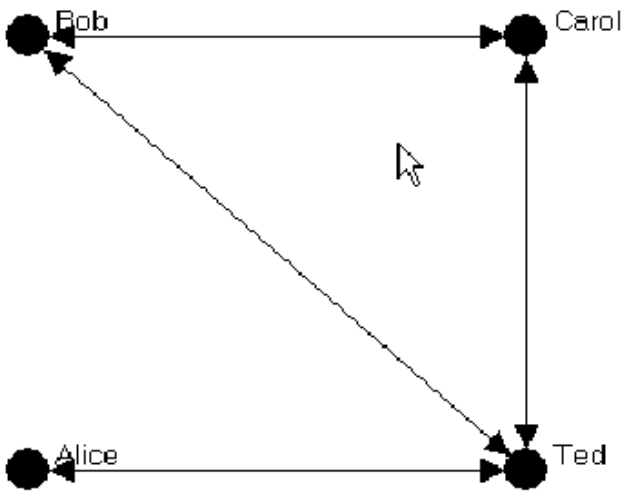


Figura 1.5 esempio di un grafo

Jacob L. Moreno è stato il pioniere dell'analisi della rete sociale per la sua famosa terapia "psychodrama". Il *sociogramma* di Moreno, anche chiamato rilevazione sociometrica, è un metodo di osservazione indiretta usato particolarmente nelle scienze dell'educazione e nelle analisi sociali. Il questionario sociometrico serve per analizzare, all'interno di un gruppo, la posizione di un individuo; serve a fornire informazioni sulla situazione del gruppo e ad individuare i leaders e gli emarginati. Mostriamo un esempio in Figura 1.6. La tecnica, quindi, non si occupa direttamente dei

comportamenti manifesti, ma si avvale di un questionario per scoprire le relazioni interpersonali tra i componenti di un gruppo. Il fine del questionario è quello di evidenziare la struttura psicosociale dei gruppi e di trascriverla in maniera oggettiva. Questo metodo mette in luce le attrazioni e le repulsioni che ci sono tra i vari componenti di un gruppo, attraverso quattro items che chiedono a questi ultimi di esprimere la propria opinione in termini di rifiuto, di scelta o di indifferenza nei confronti degli altri componenti. Il questionario consiste nel chiedere a tutti i membri di indicare in ordine decrescente di preferenza i compagni con i quali vorrebbero svolgere un'attività specifica (prima domanda) e al contrario con chi non vorrebbero assolutamente associarsi (seconda domanda). Nel questionario sono contenute anche due domande di carattere percettivo, in cui ai soggetti viene chiesto di esprimere da quali degli altri soggetti pensano di essere stati selezionati e da quali no (terza e quarta domanda) [Moreno 34].

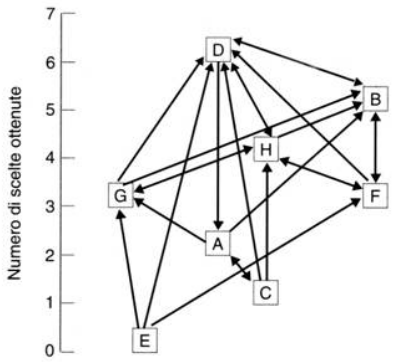


Figura 1.6 sociogramma

Nell'esempio della Figura 1.5, possiamo notare che gli attori sono rappresentati dalle persone, mentre le relazioni sono rappresentate da valori numerici (1, se le persone sono collegate, 0 se non lo sono). Il verso della relazione è orientato dalle colonne alle righe (esempio Bob like Ted ma non accade Ted like Bob, sono distinti). Inoltre, questo studioso utilizza la sociomatrice, visualizza un grafo diretto (digrafo) di una classe di ragazzi con la relazione di "like". In riferimento alla Figura 1.5 mostriamo la Tabella 1.1 che indica come i dati sono rappresentati. Se "Bob" ha una freccia verso "Ted" allora poniamo 1 se "Bob" non ha una freccia verso "Ted" allora poniamo 0. Notiamo che la diagonale non ha nessun valore.

	Bob	Carol	Ted	Alice
Bob	--	0	1	1
Carol	1	--	0	1
Ted	0	1	--	1
Alice	1	0	0	--

Tabella 1.1

L'analisi della rete sociale è una tecnica usata da analisti di "intelligence" per individuare dei patterns, anormali tra interazioni sociali. Tale metodo nasce nel 1930 dagli studi di struttura sociale nei campi della psicologia sociale e dell'antropologia sociale. L'analisi della rete sociale viene utilizzata per scoprire pattern relazionali, strutture comunitarie (o organizzazioni di altra forma) e reti informali (come corporazioni, gruppi filiali o reti di computer).

Granovetter dimostra che i soggetti inseriti in legami deboli, fatti, in altre parole, cioè di conoscenze amicali non troppo strette, hanno più possibilità di accesso ad informazioni e quindi di potenziali posizioni lavorative di proprio interesse, rispetto a coloro che investono socialmente soltanto nei legami forti, cioè i familiari, i parenti e gli amici intimi [Granovetter 83].

La versione di Internet di una rete sociale è una delle forme più evolute di comunicazione in rete. Tra le reti sociali preferiti dagli utenti troviamo *facebook* e *myspace* rispettivamente con 132 e 117 milioni di utenti, con il sorpasso del primo sul secondo nell'aprile del 2008 [1].

1.1.3 SN: matematico-fisico

Un importante esperimento riguardante la rete sociale è quello sulla teoria del mondo piccolo di [Migram 67]. L'esperimento esamina la lunghezza media del percorso per reti sociali tra residenti negli Stati Uniti. La ricerca ipotizza un mondo piccolo, costituito da una rete di collegamenti tra persone relativamente breve. È risaputo che qualsiasi coppia di persone nel mondo può essere connessa ad un'altra da una catena di conoscenti intermedi, tipicamente di lunghezza sei. L'esperimento è associato spesso con il popolare concetto dei *sei gradi di separazione*, vale a dire il numero di passaggi sociali (amici degli amici degli amici…), che separano, mediamente, qualsiasi essere umano da un altro [Watts 03].

In matematica e fisica una rete small-world o piccolo mondo è un tipo di grafico matematico nel quale la maggior parte dei nodi non ha molti nodi adiacenti, ma la maggior parte dei nodi può

[1] http://it.wikipedia.org/wiki/Rete_sociale, Febbraio, 2009.

raggiungere tutti gli altri con un piccolo numero di passi [Watts 99]. La ricerca ipotizza un mondo piccolo, costituito da una rete di collegamenti tra persone relativamente breve. Questa teoria generalizza ed esplora le caratteristiche d'insieme, che hanno reti connesse di elementi, indipendentemente dalle caratteristiche proprie degli elementi. Reti di router, compratori, attori e partner sessuali hanno almeno due caratteristiche simili: l'alto livello di aggregazione e il basso grado di separazione. La teoria illustra appunto come sia possibile conciliare questi due aspetti apparentemente contraddittori: il fatto che nonostante ogni elemento tenda ad avere relazioni prevalentemente con pochi altri (alta aggregazione) non impedisce di ottenere comunque una sua "vicinanza", tramite pochi intermediari, con qualsiasi altro elemento della rete (basso grado di separazione).

In statistica molti casi sono descritti tramite la distribuzione Gaussiana. Tuttavia, esistono dei casi in cui la distribuzione si discosta parecchio da quella che dovrebbe essere secondo la Gaussiana. In Internet, ad esempio, la maggior parte dei routers ne registrano un grado molto basso (per esempio "home" routers), ma ne esistono alcuni che hanno gradi estremamente alti (per esempio i routers che gestiscono i provider più famosi) [Mendelzon 96]. La distribuzione *power law* prova a formalizzare un modello che considera anche casi come questi in cui la distribuzione di Gauss non è sempre verificata.

Formalmente due variabili x e y sono connesse da una power law quando lo scatter plot (un esempio di plotter riferito a parole frequenti di wikipedia in Grafico 1.1) è lineare su una scala log-log:

$$y(x) = cx^{-\upsilon}$$

dove c e v sono costanti positive. La costante v è spesso chiamata esponente di power law.

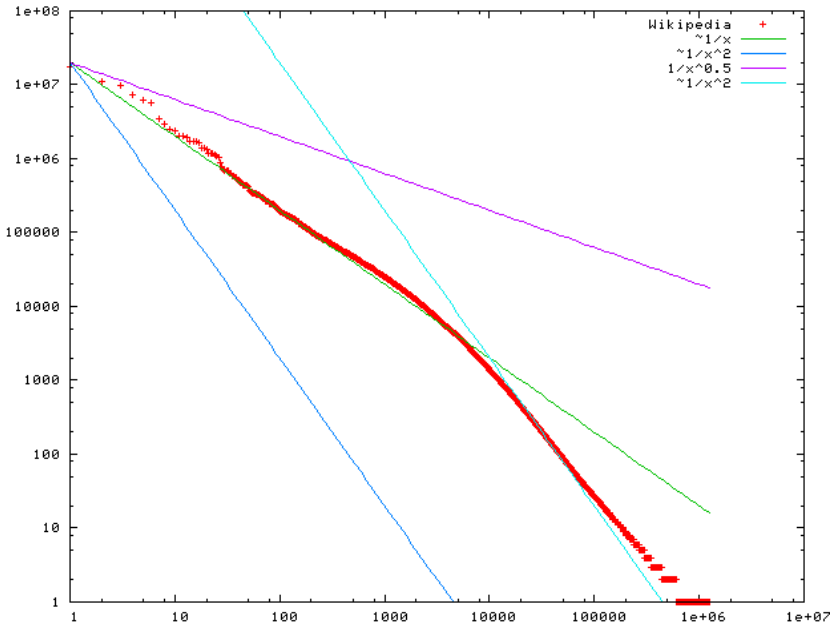


Grafico 1.1

Questi esempi mostrano che patterns possono essere estratti da una qualsiasi informazione strutturata a disposizione. Inoltre, eventuale struttura mancante può essere anche inferita da questi dati.

Nella prima metà del 1950 [Simon 55], è stata realizzata per la prima volta l'idea dei modelli *preferential* attachment. Tale idea è stata riscoperta recentemente per la capacità che questi modelli hanno di generare distribuzione secondo la power laws [Barabási 99]. Informalmente, il concetto che si vuole evidenziare è quello dei "i soldi vanno ai ricchi": nuovi nodi uniti al grafo, in passi successivi, si connettono prevalentemente a nodi già presenti. Questa idea di base è stata molto influente e ha formato la base di molti lavori futuri [Albert 00] [Albert 02] [Barabási 02].

I modelli di preferential attachment sono probabilmente i modelli più popolari attualmente, a causa delle loro capacità di creare una corrispondenza con i gradi del power law attraverso un insieme semplice di passi. Comunque, la comunità non si è dimostrata particolarmente interessata a sviluppare l'argomento, e a parte da documenti di Pennock [Pennock 02], piccoli sforzi sono stati fatti per trovare una ragione alle differenze tra power laws e grafi del mondo reale. Un insieme di modelli relativi ha mostrato aspettative notevoli recentemente: questi sono i modelli "edge-copying" [Kleinberg 99], dove un nodo (come il website) acquisisce archi copiando collegamenti da altri nodi (websites). Questo è simile a preferential attachment, perché le pagine con alti gradi saranno collegate a molte altre pagine e così hanno una più vasta chance di essere copiati.

Un'altra analisi può essere fatta in relazione alla topologia del grafo. Per esempio, Ravasz e Barabasi [Ravasz 03] analizzarono un grafo costruito da collegamenti tra attori apparsi nello stesso film e scoprirono che il grafo ha una topologia gerarchica propria. Grafi relativi ai film (studiando il

IMDb[2]) possono essere anche usati per eseguire classificazioni. Ad esempio Jensen e Nevill [Jensen 02] usano informazioni di tale tipo di grafo come mostrato in Figura 1.7 per predire se un film farà più di 2 milioni di dollari nella sua settimana d'apertura. In uno studio separato, inoltre, usano una struttura in cui evidenziano i film candidati e non candidati ai premi oscar per predire quali potrebbero avere più possibilità di successo [McGovern 03].

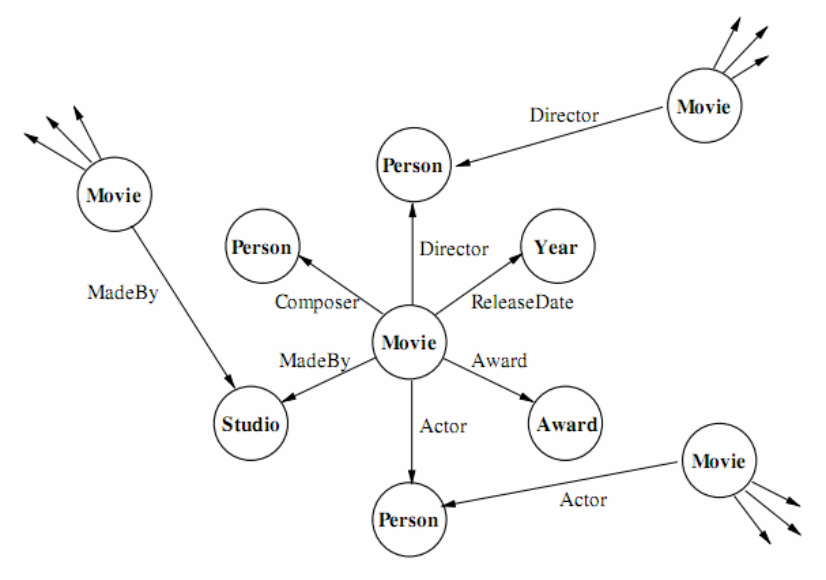


Figura 1.7 un esempio del grafo IMDb

1.1.4 SN: data mining

Il DM ha lo scopo di estrarre conoscenza implicita da databases di enormi dimensioni. Il GM, come sotto-area del DM, è caratterizzato da due aspetti principali: l'estrazione di modelli a partire dall'analisi di reti complesse e l'analisi temporale delle reti stesse.

Nel primo caso il DM è usato per identificare alcuni parametri critici, basati solo su strutture e proprietà non semantiche, che descrivono l'evoluzione microscopica di reti, come ad esempio della rete sociale. Nel secondo aspetto, invece, l'analisi si focalizza sullo studio e la formalizzazione dei cambiamenti che assume la composizione della rete con il passare del tempo.

Ogni problema di DM può essere affrontato tramite l'esecuzione di alcune fasi, definite da alcuni standard di riferimento, di cui il più popolare è chiamato CRISP-DM, mostrato in Figura 1.8.

[2] http://www.imdb.com/ - The Internet Movie Database

Il CRISP rappresenta l'attività di risoluzione di un problema di data mining tramite un processo iterativo composto dalle seguenti 6 fasi:

1. Business Understanding
2. Data Understanding
3. Data Preparation
4. Modeling
5. Evaluation
6. Deployment

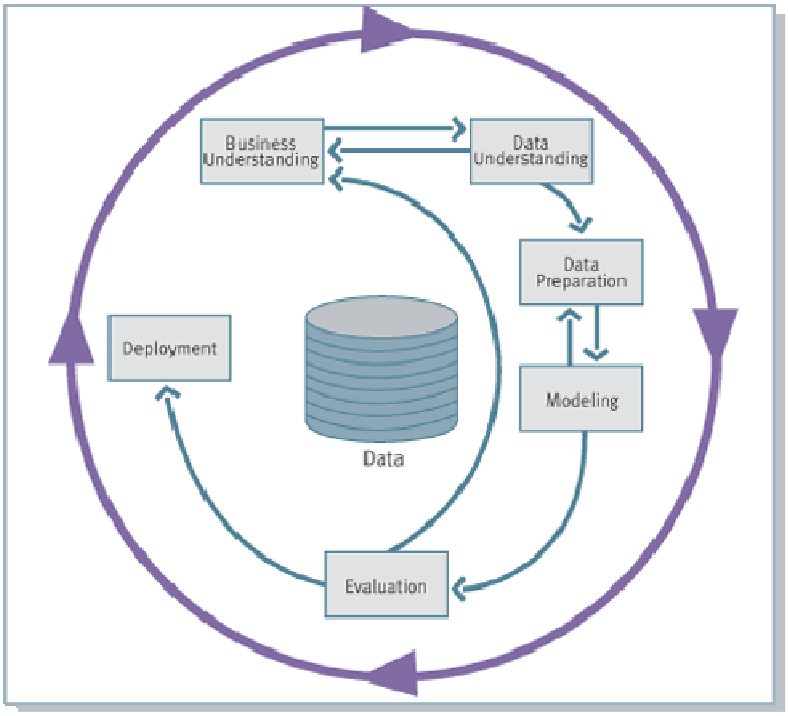


Figura 1.8 processo CRISP-DM

Appartenendo ad una sotto-area del DM, anche i problemi di GM possono essere risolti seguendo lo standard di riferimento appena presentato. Segue questa ottica anche la risoluzione del caso di studio "Social Network Analysis as Knowledge Discovery process: a case study on Digital Bibliography" [Coscia 09], che ha lo scopo estrarre informazioni da una rete di pubblicazioni contenute in una biblioteca digitale.

Il punto di partenza dell'analisi è un database contenente una serie di pubblicazioni digitali.

A partire dal database vengono svolte due distinte operazioni: la prima è l'estrazione degli autori delle varie pubblicazioni e la loro raccolta in una struttura dati, che tiene nota delle collaborazioni avvenute tra i vari autori per la realizzazione dei loro lavori.

La seconda operazione, che usa dati grezzi della bibliografia digitale, è la costruzione di una matrice authors-keywords, nella quale le righe rappresentano gli autori e le colonne rappresentano le chiavi ottenute con un processo di pulizia dei titoli e/o astrazioni del contenuto delle pubblicazioni.

Questa struttura dati è usata per ottenere una classificazione, che associa ogni autore al tipo di pubblicazioni da lui realizzate. Questo processo può essere implementato in due modi: automaticamente o con l'intervento di un esperto del dominio, che può creare un processo di classificazione più accurato.

Questi due strumenti sono usati in seguito nel processo di ricerca delle informazioni. Infatti, usando un linguaggio di query è possibile specificare vari tipi di limitazioni e filtri per estrarre da questi data warehouse una rappresentazione grafica della rete, che descrive le relazioni tra gli autori.

Una volta ottenuta la rete sociale è possibile eseguire due tipi differenti di analisi. A livello globale, in cui, attraverso la definizione di parametri statistici, che stabiliscono la forza delle relazioni a disposizione, è possibile ottenere una visione globale della rete (analisi globale).

A livello locale, invece, possiamo eseguire un vero e proprio processo di data mining: usando un algoritmo di GM è possibile estrarre patterns locali, che possono essere analizzati individualmente alla ricerca di qualche irregolarità nella rete e che quindi ne possa descrivere alcuni micro - comportamenti (analisi locale).

Lo studio di tipo locale, basata sull'esecuzione di algoritmi di GM, può, quindi, essere ricondotto all'esecuzione di un processo, che ricalca molto da vicino le tipiche fasi di un processo di DM mostrato in Figura 1.9 [Coscia 09].

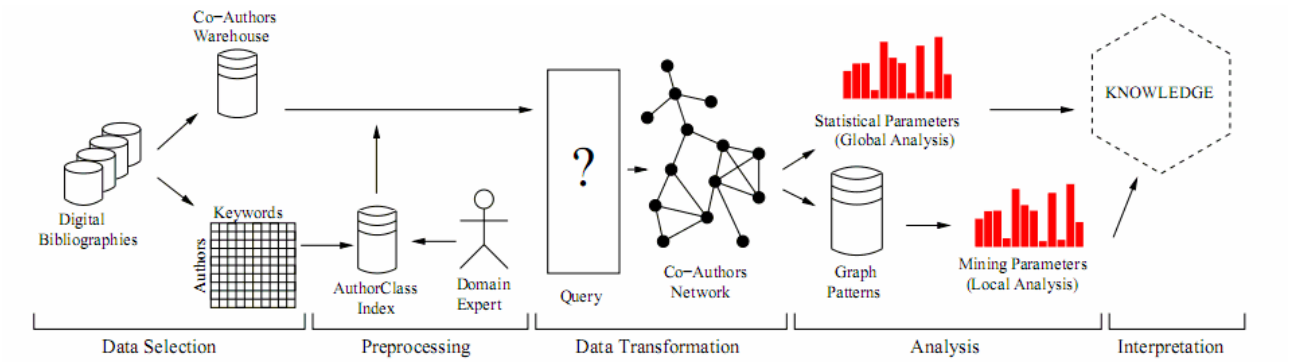


Figura 1.9 un esempio del processo del GM

Nel contesto del recupero di informazioni del web (information retrieval), gli algoritmi del PageRank [Page 98] e del HITS [Kleinberg 99] sono i più diffusi nell'approccio del LM. Il modello PageRank costruisce dei percorsi casuali, in cui, a volte, i link sono seguiti in modo sequenziale, mentre altre volte salta a nuove pagine web. L'algoritmo HITS assume un processo leggermente più complesso, modella il web come se fosse composto da due tipi di pagine: *hubs* e *authorities*. In particolare le hubs sono pagine web collegate a molte pagine authorities. Ad ogni pagina è assegnato un punteggio di hub e di authority attraverso un algoritmo iterativo che si basa sui punteggi delle pagine a lei vicine.

Nel dominio d'analisi delle reti sociali uno degli obiettivi è trovare un rank ordinato in una data rete sociale che ne determini la misura della loro importanza, nella misura della *centralità*. Le misure di centralità sono state l'oggetto di ricerca nella comunità della SNA per decenni. Queste misure caratterizzano alcuni aspetti della rete strutturata locale o globale attraverso, ad esempio, l'importanza degli individui in base alla loro connessione ad altri individui importanti [Lise 05].

1.2 Data mining

Il DM, visto come attività sviluppata, per far fronte all'esigenza di sfruttare il patrimonio informativo, contenuto nelle grandi raccolte di dati, costituisce la parte più importante di un processo (mostrato in Figura 1.10) più ampio, chiamato Knowledge discovery in databases (KDD): il KDD, descritto per la prima volta nel 1996 da Usama Fayyad, Piatetsky-Shapiro e Smyth, si può definire come:

Processo interattivo ed iterativo, costituito dall'insieme dei metodi e delle tecniche utilizzate allo scopo di identificazione di relazioni tra dati, che siano valide, nuove, potenzialmente utili e comprensibili [Fayyad 96].

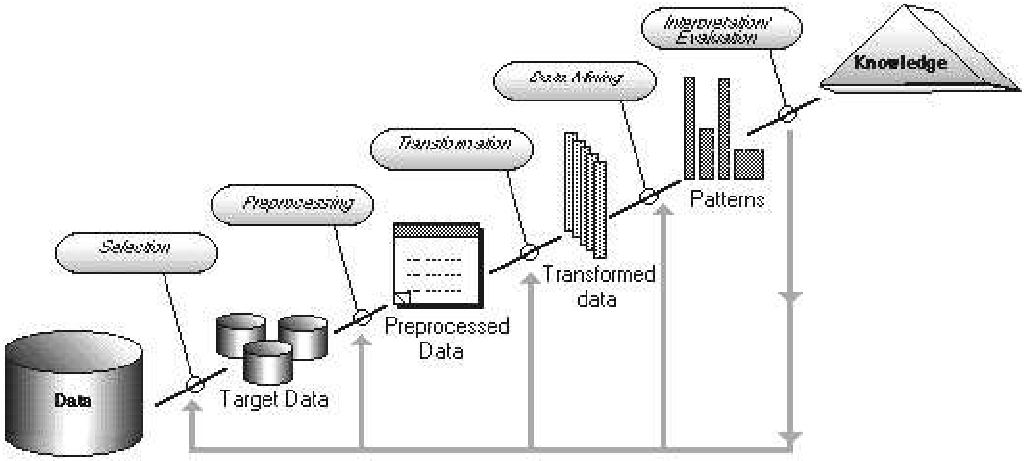


Figura 1.10 processo del KDD

Il processo KDD prevede come dati in input dati grezzi e fornisce come output informazioni utili ottenute attraverso le seguenti fasi:

- *Selezione*: i dati grezzi vengono segmentati e selezionati secondo alcuni criteri al fine di pervenire ad un sottoinsieme di dati, che rappresentano il nostro target data o dati obiettivo. È abbastanza chiaro come un database possa contenere diverse informazioni, che per il problema in fase di studio, possono risultare inutili. Per fare un esempio: se l'obiettivo è lo studio delle associazioni tra i prodotti comprati dai clienti di una catena di distribuzione al dettaglio, non ha senso conservare i dati relativi alla professione dei clienti; è invece assolutamente errato non considerare tale variabile, che potrebbe invece fornire utili informazioni relative nel caso in cui si volesse segmentare la clientela in base alle loro caratteristiche socio-demografiche;
- *Preparazione*: spesso non è conveniente né necessario analizzare l'intero contenuto di un insieme di dati ma può essere più adeguato prima selezionarne solo una parte e in seguito esplorare solo le informazioni scelte in modo da realizzare un'analisi su base campionaria. Fanno, inoltre, parte di questo stadio del KDD la fase di pulizia dei dati (data cleaning), che prevede l'eliminazione dei possibili errori e la decisione dei meccanismi di comportamento in caso di dati mancanti;
- *Trasformazione*: effettuata la fase precedente, i dati, per essere utilizzabili, devono essere trasformati. Si possono convertire alcuni tipi di dati in altri o definirne di nuovi, attraverso l'uso di operazioni matematiche e logiche sulle variabili. Inoltre, soprattutto quando i dati provengono da fonti diverse, è necessario effettuare una loro riconfigurazione al fine di garantirne la consistenza;
- *Data Mining*: ai dati trasformati vengono applicate una serie di tecniche in modo da poter ricavare delle informazioni che siano non banali o scontate ma utili e non predicibili a priori.

I tipi di dati che si hanno a disposizione e gli obiettivi che si vogliono raggiungere possono dare un'indicazione circa il tipo di metodo da scegliere per la ricerca di informazioni dai dati;

- *Interpretazione e valutazione*: il DM crea dei patterns, ovvero dei modelli, che possono costituire un valido supporto alle decisioni. Non basta però interpretare i risultati attraverso dei grafici che visualizzano l'output del DM, ma occorre valutare questi modelli e cioè capire in che misura questi possono essere utili. È, dunque possibile, alla luce di risultati non perfettamente soddisfacenti, rivedere una o più fasi dell'intero processo KDD;
- *Un fatto è certo*: l'intero KDD è un processo iterativo tra l'utente, il software utilizzato e gli obiettivi. È anche iterativo nel senso che la fase di DM può prevedere un'ulteriore trasformazione o un'ulteriore pulizia dei dati originali, ritornando di fatto alle fasi precedenti.

In un processo di DM si possono distinguere: la strategia, la tecnica e modelli.

La ***strategia*** definisce il tipo di modelli di analisi che vogliamo estrarre e si divide in:

- **Supervisionata**: quando l'obiettivo è quello di costruire un modello a partire da elementi di cui sappiamo già il risultato a cui deve portare l'analisi (training set). Durante la fase più importante di questa strategia, chiamata *fase di apprendimento,* le informazioni presenti nel training set vengono analizzate e generalizzate in un modello, che sarà applicato ai dati nuovi. Strategie di questo tipo rispondono a domande del tipo : *per quali motivi si verifica un certo fatto?*
- **Non supervisionata**: quando l'obiettivo è quello di definire un modello senza avvalersi di un apprendimento eseguito su casi di cui sappiamo a priori le relazioni tra le variabili. Questa strategia risponde a domande del tipo: *esiste qualcosa di interessante sui dati?*

La ***tecnica*** definisce la modalità in cui viene applicata la strategia per creare i modelli. Le tecniche da applicare dipendono dal tipo di strategia che vogliamo attuare e dal tipo di modello di DM che vogliamo estrapolare dai dati. Le più diffuse tecniche di DM sono le seguenti:

- classificazione
- regressione
- clustering
- scoperta di associazioni tra variabili
- scoperta di pattern sequenziali
- Predizione di valori

I ***modelli*** *di DM* sono rappresentazioni matematiche e schemi che evidenziano le relazioni tra le variabili che descrivono i dati. Tra i più diffusi modelli di DM citiamo:

- alberi di classificazione
- classificatori bayesiani
- reti neurali
- cluster
- regole associative
- pattern sequenziali
- pattern frequenti

In generale, i modelli di DM si possono dividere in due categorie: descrittivi o predittivi.

Un modello *descrittivo* si pone lo scopo di identificare le relazioni presenti tra i dati (correlazioni, trend, cluster, anomalie...). I modelli di tipo descrittivo hanno di solito una natura esplorativa e richiedono un post-processing dei dati per validare e spiegare i risultati ottenuti.

Un modello *predittivo* ha invece come obiettivo quello di predire il valore di un particolare attributo basandosi sul valore di altri attributi: quello da predire è chiamato target o dipendente, mentre quelli usati per fare la previsione sono chiamati indipendenti.

Il DM trova diverse applicazioni nell'ambito dell'economia e della finanza, nella scoperta di frodi bancarie, in applicazioni scientifiche, nella cura della salute e in sport e giochi. Uno degli ambiti in cui l'applicazione del DM si rivela particolarmente proficua è il Marketing. La strategia applicata in questo caso è quella della Market Basket Analysis (MBA), il cui scopo è quello di ricercare relazioni interessanti, chiamate *regole associative*, tra i prodotti delle vendite al dettaglio, al fine di sviluppare successivamente strategie di marketing incrociato.

Consideriamo un supermercato con una vasta tipologia di prodotti. Tipicamente le decisioni di business, che il management del supermercato deve prendere, includono quelle su cosa piazzare sul mercato, come progettare coupons, come disporre la merce sugli scaffali e quelle su come massimizzare il profitto. Analizzare le transazioni di dati riguardanti le vendite passate è un approccio usato comunemente per migliorare la qualità delle decisioni. Dati sulle vendite cumulativi durante un periodo di tempo (un giorno, una settimana, un mese, ecc.), sono disponibili nei grandi *mainframe* delle aziende. La memorizzazione dei dati è chiamata *basket* [Agrawal 93].

L'obiettivo è quello di costruire un modello di rappresentazione dei dati con regole del tipo *se-allora*, che correlano la presenza di certi valori nei dati. Ciò per stabilire quali prodotti hanno una maggiore probabilità di essere acquistati insieme ad altri. Per esempio, sul 90% di transazioni totali

di un supermercato si nota che *se si* è comprato il pane e il burro *allora* si è comprato anche il latte. La scoperta del dato che esistono certe tendenze di acquisto da parte dei clienti consente di organizzare la disposizione dei prodotti sugli scaffali, o nel catalogo, in modo da attirare l'attenzione sui prodotti acquistati insieme. Scopo dell'analisi è quello di scoprire le abitudini di acquisto dei clienti, studiando gli articoli che vengono acquistati contemporaneamente, nella stessa transazione.

In Figura 1.11 mostriamo l'algoritmo delle regole associative [Agrawal 93].

```
1)  L_1 = {large 1-itemsets};
2)  for ( k = 2; L_{k-1} ≠ ∅; k++ ) do begin
3)     C_k = apriori-gen(L_{k-1}); // New candidates
4)     forall transactions t ∈ D do begin
5)        C_t = subset(C_k, t); // Candidates contained in t
6)        forall candidates c ∈ C_t do
7)           c.count++;
8)     end
9)     L_k = {c ∈ C_k | c.count ≥ minsup}
10) end
11) Answer = ⋃_k L_k;
```

Figura 1.11 algoritmo Apriori

1.3 Graph mining

Molti dati interessanti oggi giorno, descrivono molto bene collegamenti di collezione di oggetti correlati. Questi potrebbero rappresentare reti omogenee, nelle quali c'è un unico tipo di oggetto e di collegamento, oppure potrebbero essercene più tipi, come reti eterogenee, nelle quali potrebbero esserci multipli tipi di oggetti e di collegamenti. Esempi di reti omogenee includono un singolo tipo, come si verifica nelle rete sociali, le connessioni tra persone di collegamenti di amicizia, oppure il WWW, un insieme di collegamenti di pagine web. Esempi di reti eterogenee sono incluse nel dominio medico come: descrizione di pazienti, malattie, terapie e contatti, o, nel campo del dominio bibliografo. descrizione di pubblicazione, autori e luogo del convegno come (DBLP). Il Link Mining tratta tecniche di DM ed esplicitamente considera questi collegamenti quando si costruiscono previsioni o descrizioni di modelli tra collegamenti di dati. Comunemente, il compito del LM include ranking di oggetti, rilevamento di gruppi, classificazione collettivo,

predizione di collegamenti e scoperta di sottografi. L'analisi di reti studia, invece, in modo dettagliato le aeree come l'analisi delle reti sociali, ipertesto mining e web analisi.

"Links", o più genericamente relazioni, questi collegamenti spesso mostrano patterns che possono indicare proprietà di dati come per esempio: l'importanza, il rank, o categorie di oggetti.

Gli algoritmi tradizionali di DM (regole associative, market basket analysis e cluster analysis), tentano comunemente di trovare patterns in un insieme di dati caratterizzato da una collezione di esempi indipendenti. Questo concetto è riconducibile al classico problema dell'inferenza statistica in cui l'obiettivo è quello di identificare modelli, dati dei campioni indipendenti e con distribuzione identica.

Il miglior modo di rappresentare un insieme di dati sono le reti o i grafi. Il dominio spesso consiste di una varietà di tipi di oggetti; gli oggetti possono essere collegati in vari modi. Perciò il grafo potrebbe avere differenti tipi di nodi e di archi.

LM è un'area di ricerca emergente, che incrocia i lavori di analisi dei link, hypertext mining, web mining e GM. Viene utilizzato il termine LM per inserire una speciale enfasi sui link [Lise 05].

1.3.1 Algoritmi

In generale, per scoprire sottostrutture frequenti si passa attraverso due fasi. Il primo passo è costituito dalla generazione di candidati sottostrutture frequenti, mentre la frequenza di ogni candidato è controllata nel secondo passo.

Gli algoritmi, richiedono un'operazione di *join* per unire due sottostrutture frequenti in una sottostruttura frequente più grande. Si distinguono differenti modi per generare sottostrutture più grandi: vertici e archi. Nell'ambito di sottostrutture frequenti viene usato l'algoritmo basato su Apriori. Questo algoritmo ha due *overhead* importanti:

1. unire due grafici frequenti di dimensione-k per generare un grafo candidato di dimensione-($k+1$);
2. controllare separatamente la frequenza di questi candidati.

Questi overheads costituiscono il collo di bottiglia dell'algoritmo. Per aumentare le performance sono utilizzati algoritmi non-Apriori come gSpan, MoFa, FFSM. Questi algoritmi sono ispirati da PrefixSpan [Pei 01], TreeMinerV [Zaki 02] e FREQT [Asai 2002] a una sequenza mining e alberi, rispettivamente. Tutti questi algoritmi adottano la metodologia della crescita del pattern [Han 00].

Proponiamo una serie di algoritmi sul GM: Apriori-based Graph Mining (AGM), Mining frequent subgraphs (FSG), Graph-Based Substructure Pattern Mining (gSpan).

Definizione grafo frequente: dato un dataset con grafi etichettati, $D = \{G1, G2, ..., Gn\}$, il supporto(g) (o frequenza(g)) è la percentuale (o numero) di grafi in D dove g è un sottografo. Un grafo è frequente se il suo supporto è più grande della soglia del minimo supporto (*min_supporto*).

1.3.1.1 AGM

Inizialmente l'algoritmo di mining di strutture frequenti è stato proposto da Inokuchi [Inokuchi 00], il quale condivide caratteristiche simili con Apriori item set mining [libro DGM].

Il framework generale del metodo Apriori è mostrato in Figura 1.12. S_k è l'insieme della sottostruttura di dimensione k. Ad ogni iterazione dell'algoritmo, scopre nuove sottostrutture frequenti con una dimensione in più. Queste nuove sottostrutture vengono generate dall'unione di due simili, ma leggermente differenti. Questa procedura genera candidati tramite l'istruzione riferita alla linea 4. I grafici frequenti sono identificati e utilizzati per generare candidati più grandi al prossimo passo.

***Algorithm* 5.1** Apriori(D, *min_support*, S_k)

```
Input: A graph dataset D and min_support.
Output: A frequent substructure set S_k.

1: S_{k+1} ← ∅;
2: for each frequent g_i ∈ S_k do
3:   for each frequent g_j ∈ S_k do
4:     for each size (k+1) graph g formed by the merge of
       g_i and g_j do
5:       if g is frequent in D and g ∉ S_{k+1} then
6:         insert g to S_{k+1};
7: if S_{k+1} ≠ ∅ then
8:   call Apriori(D, min_support, S_{k+1});
9: return;
```

Figura 1.12 algoritmo Apriori per i grafi

Per esempio, supponiamo che abbiamo due item set frequenti di dimensione 3: (abc) e (bcd), l'item set candidati frequente di dimensione 4 generato è (abcd). Perciò, il problema della generazione dei candidati nella sottostruttura frequenza diventa molto più complesso in caso di item set frequenti, di conseguenza sono stati introdotti molti modi per unire due sottostrutture mostriamo un esempio in Figura 1.13.

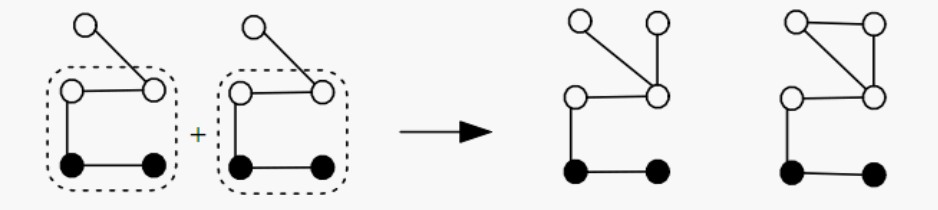


Figura 1.13

1.3.1.2 FSG

L'algoritmo FSG proposto da Kuramochi e Karypis [Kuramochi 01] adotta un metodo basato sull'incremento degli archi. La dimensione della struttura aumenta di un arco ad ogni iterazione. Nel FSG, due patterns di dimensione-*k* sono unite se e solo se condividono gli stessi sottografi con (*k-1*) archi. In questi algoritmi la dimensione del grafo è riferita al numero degli archi nel grafo. Nuovamente i candidati formano un grafo con l'aggiunta di due archi. In Figura 1.14 vengono mostrati i candidati formati dall'algoritmo FSG.

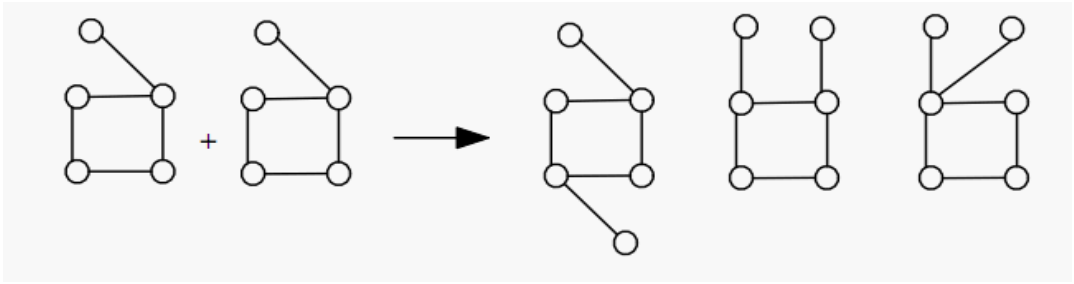


Figura 1.14

Qui in Figura 1.15 mostriamo l'algoritmo FSG.

Algorithm 1 fsg(D, σ) (Frequent Subgraph)

1: $F^1 \leftarrow$ detect all frequent 1-subgraphs in D
2: $F^2 \leftarrow$ detect all frequent 2-subgraphs in D
3: $k \leftarrow 3$
4: **while** $F^{k-1} \neq \emptyset$ **do**
5: $\quad C^k \leftarrow$ fsg-gen(F^{k-1})
6: $\quad$ **for each** candidate $g^k \in C^k$ **do**
7: $\quad\quad g^k$.count $\leftarrow 0$
8: $\quad\quad$ **for each** transaction $t \in D$ **do**
9: $\quad\quad\quad$ **if** candidate g^k is included in transaction t **then**
10: $\quad\quad\quad\quad g^k$.count $\leftarrow g^k$.count $+ 1$
11: $\quad F^k \leftarrow \{g^k \in C^k \mid g^k.\text{count} \geq \sigma|D|\}$
12: $\quad k \leftarrow k + 1$
13: **return** $F^1, F^2, \ldots, F^{k-2}$

Figura 1.15 algoritmo FSG

1.3.1.3 gSpan

L'algoritmo gSpan [Yan 02] è progettato per ridurre la generazione di grafici duplicati, garantendo ancora la scoperta di un insieme completo di grafici frequenti.

La ricerca depth-first è utilizzata nel gSpan per l'esplorazione del grafo. Inizialmente, inizia scegliendo un vertice a caso. I vertici del grafo sono marcati cosicché è possibile individuare i vertici visitati. L'insieme dei vertici visitati aumenta ripetutamente fino a quando è costruito un albero DFS. Un grafo potrebbe avere vari alberi DFS. Ciò dipende dal tipo di ricerca depth-first eseguita, cioè, in base all'ordine dei vertici visitati.

Gli archi evidenziati in Figura 1.16(b) 1.16(c) 1.16(d) mostrano 3 alberi DFS per lo stesso grafo di Figura 1.16(a). Quando costruiamo un albero DFS, la sequenza dei vertici visitati forma una linea ordinata. Utilizziamo un *subscripts* per memorizzare questo ordine, quando $i < j$ cioè significa che v_i è visitato prima di v_j. T è chiamato un *subscripting* DFS in G.

Dato T un albero DFS, chiamiamo un vertice iniziale in T, v_0 la radice e l'ultimo vertice visitato, v_n, i vertici rightmost. Il percorso diretto da v_0 a v_n è chiamato percorso rightmost. In Figura 1.16(b) 1.16(c) 1.16(d), vengono generati 3 differenti subscriptings sul albero corrispondente DFS. Il percorso rightmost è (v_0, v_1, v_3) in Figura 1.16(b) e 1.16(c), e (v_0, v_1, v_2, v_3) in Figura 1.16(d).

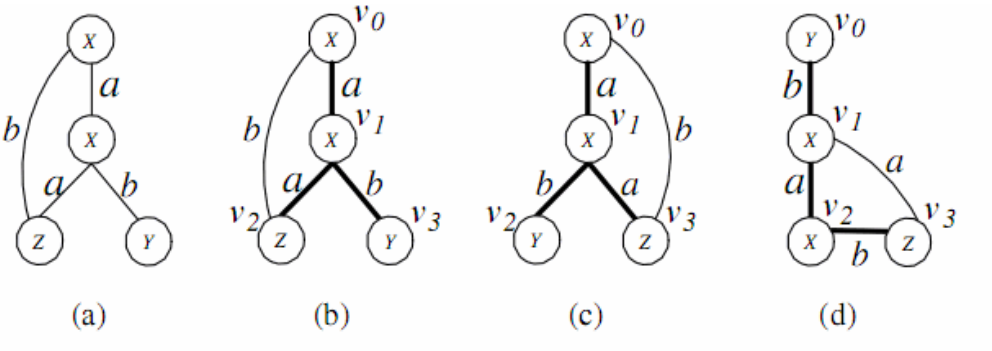


Figura 1.16

Dato un grafo G e un albero T di DFS in G, un nuovo arco e può essere aggiunto tra il vertice rightmost e altri vertici su percorsi rightmost; o può essere introdotto un nuovo vertice e connesso a vertici su rightmost. Entrambi i tipi di estensione prendono posto a un percorso rightmost, possono estendersi rightmost, denominato $G \Diamond_r e$ (per brevità qui T è omesso).

Trasformiamo ogni grafo subscripted a un sequenza di arco, chiamata *code* DFS, in modo che possiamo costruire un ordine tra tutte le sequenze generate. L'obiettivo è quello di selezionare il subscripting che genera la minore sequenza.

I dettagli di gSpan sono visualizzati nell'Algoritmo in Figura 1.17, gSpan è chiamato ricorsivamente per ampliare un pattern di un grafo fino a quando il supporto del grafo è minore del min_supporto o il suo code non è più minimo.

***Algorithm* 5.3** gSpan(s, D, *min_support*, S)

```
Input: A DFS code s, a graph dataset D, and min_support.
Output: A frequent substructure set S.

1: if s ≠ dfs(s), then
2:     return;
3: insert s into S;
4: set C to ∅;
5: scan D once, find all the edges e such that s can be
   rightmost extended to s◇r e; insert s◇r e into C and
   count its frequency;
6: sort C in DFS lexicographic order;
7: for each frequent s◇r e in C do
8:    Call gSpan(s◇r e, D, min_support, S);
9: return;
```

Figura 1.17 algoritmo gSpan

1.3.2 Complessità

Molti problemi decisionali presentano un'importanza significativa per un insieme di caratteristiche comuni. Per tali problemi, infatti, non sono stati individuati algoritmi polinomiali per la loro soluzione (né, d'altra parte, è stata trovata alcuna dimostrazione di una loro esponenzialità). Al tempo stesso, tutti questi problemi sono decidibili in tempo polinomiale da macchine di Turing non deterministiche.

Un algoritmo di questo tipo è del tutto inutile, salvo che per casi semplicissimi. Gli algoritmi, che utilizzano, al crescere delle dimensioni dell'input, una quantità di risorse "irragionevolmente alta", sono chiamati intrattabili, altrimenti sono trattabili.

Una funzione di complessità fattoriale o esponenziale porta a una rapida esplosione dei tempi di esecuzione. Ciò in funzione delle dimensioni dell'ingresso di n.

Come si vede nella Tabella 1.2, in cui si ipotizza di eseguire gli algoritmi su un calcolatore in grado di svolgere un milione di operazioni elementari al secondo. In riga, troviamo la funzione dell'algoritmo e sulle colonne la dimensione dell'input.

Complessità	$n=10$	$n=20$	$n=50$	$n=100$	$n=10^3$	$n=10^4$	$n=10^5$	$n=10^6$
n	$10\mu s$	$20\mu s$	$50\mu s$	$0{,}1ms$	$1ms$	$10ms$	$0{,}1s$	$1s$
$n\log_2 n$	$33{,}20\mu s$	$86{,}40\mu s$	$0{,}28ms$	$0{,}6ms$	$9{,}9ms$	$0{,}1s$	$1{,}6s$	$19{,}9s$
n^2	$0{,}1ms$	$0{,}4ms$	$2{,}5ms$	$10ms$	$1s$	$100s$	$2{,}7h$	$11{,}5g$
n^3	$1ms$	$8ms$	$125ms$	$1s$	$16{,}6m$	$11{,}5g$	$31{,}7a$	$\sim 300c$
2^n	$1ms$	$1s$	$35{,}7a$	$\sim 10^{14}c$	∞	∞	∞	∞
3^n	$59ms$	$\sim 1h$	$\sim 10^8 c$	∞	∞	∞	∞	∞

Tabella 1.2

Nella Tabella 1.3 riassumiamo invece le dimensioni massime dell'input elaborabile in un minuto in funzione della complessità dell'algoritmo.

Complessità	Input max
n	6×10^7
$n\log_2 n$	28×10^5
n^2	77×10^2
n^3	390
2^n	25

Tabella 1.3

La distinzione fondamentale è fra gli algoritmi che hanno complessità esponenziale (o più alta) e gli algoritmi di complessità polinomiale. Un algoritmo ha *complessità polinomiale* se la sua

complessità non cresce più rapidamente di un polinomio, ossia se è $O(n^k)$ per qualche k. Per esempio, $n\log(n)$ ha complessità polinomiale (perché cresce più lentamente del polinomio n^2, mentre 2^n cresce più rapidamente di qualunque polinomio). Questa considerazione è generale: gli algoritmi trattabili sono solo quelli di complessità polinomiale.

In generale, i problemi per i quali esiste un algoritmo trattabile sono anch'essi detti trattabili, altrimenti sono intrattabili. Quindi la classe dei problemi trattabili coincide con la classe dei problemi che ammettono l'esistenza di algoritmi polinomiali per la loro soluzione (su una Macchina di Turing). Tale classe viene chiamata P (Polinomiale).

Esiste una grande varietà di problemi di notevole interesse pratico, detti problemi NP-complete, per cui esistono forti motivi per pensare che siano tutti intrattabili, cioè non appartengano alla classe P, senza però poterlo asserire con certezza.

La classe dei problemi per cui esiste un algoritmo polinomiale nelle dimensioni dell'input (per una Macchina di Turing), utile per controllare che una soluzione data sia corretta, si chiama NP (Non deterministico Polinomiale).

Tutti i problemi della classe P sono ovviamente anche in NP. Esistono problemi in NP che non siano in P? A questa domanda non sappiamo ancora rispondere. Nessuno è mai riuscito a dimostrare per un problema NP-complete un algoritmo deterministico polinomiale. Non si sa se P=NP o se P≠NP.

I problemi NP-complete hanno un ruolo davvero speciale nella classe NP. Basterebbe trovare un algoritmo che risolva in tempo polinomiale un solo problema NP-complete per consentire di risolvere in tempo polinomiale qualunque altro problema della classe.

1.3.2.1 Suggerimenti per problemi NP-complete

Al momento attuale, è opinione diffusa che algoritmi per problemi NP richiedano un tempo super polinomiale, in base alla dimensione dell'input e si ignora se esiste un algoritmo qualsiasi che sia più veloce.

Nella pratica, di solito, i problemi NP-complete sono considerati intrattabili "per ignoranza". Quando, infatti, si scopre che un problema interessante è NP-complete, si può tranquillamente interrompere la ricerca di un algoritmo polinomiale, in quanto è assai improbabile riuscire a trovarlo, perché questo equivarrebbe a risolvere così la questione P=NP.

Questo, però, non significa che, un problema intrattabile sia assolutamente senza speranza di risoluzione algoritmica efficiente, tanto da rinunciare a provare a risolverlo. Si possono, infatti,

applicare tecniche algoritmiche ben note, che permettono di risolvere il problema in modo efficiente, non in generale, ma in casi particolari (o a costo di accettare soluzioni anche imprecise).

Esistono varie tecniche per risolvere numerosi problemi intrattabili d'importanza pratica. Per esempio: si possono "scegliere bene" quali soluzioni alternative considerare per prime. Ciò al fine di eliminare precocemente quelle soluzioni che, sicuramente, non sono accettabili. Questo semplice accorgimento può ridurre notevolmente il numero di passi dell'algoritmo, anche se il caso peggiore resta sempre intrattabile. Spesso, gli algoritmi sono basati su strategie empiriche dette "euristiche". In tali casi essi cercano di arrivare velocemente alla soluzione corretta di molti casi particolari. Anche la "parametrizzazione" aiuta notevolmente. Infatti, spesso, ci sono algoritmi veloci se i parametri di input sono fissi. Altre tecniche euristiche (quelle più sofisticate), rinunciano a produrre una soluzione esatta, accontentandosi di garantire un limite all'errore. Infine, esistono algoritmi esponenziali che, in pratica, si comportano sempre molto bene, in quanto l'esponenzialità corrisponde a casi patologici, che, di fatto, non si verificano mai. Un esempio, in tal senso, è il famoso algoritmo del simplesso. Tale algoritmo viene usato per la soluzione di problemi di programmazione lineare ed in pratica, è sempre polinomiale (anche se il caso peggiore è esponenziale). Si noti però che il problema della programmazione lineare è comunque nella classe P.

2 Definizione del problema

Prima di analizzare il problema, appare utile conoscere alcune proprietà del grafo: cos'è, com'è rappresentato, quali sono le sue potenzialità e quali sono i limiti.

In seguito descriveremo brevemente la struttura dati considerata, per poi introdurre la caratterizzazione del problema in modo dettagliato.

2.1 Dominio Applicativo: Analisi investigativa

Per *intelligence* si intende l'applicazione di una metodologia usata dall'esperto delle informazioni.

Tale metodologia è molto utilizzata per lo studio dei sistemi criminali (quali: terrorismo, psicopatologia politica). Il suo scopo è quello di osservare, capire, imparare, cioè vedere oltre l'apparenza, dove gli altri non vedono perché non possono o non vogliono. Per esempio, nel campo della criminologia, tale metodologia consente di gestire e analizzare le informazioni, attraverso cui comprendere precocemente i fenomeni di grande allarme e pericolosità sociale.

Quanto più l'attività d'intelligence sarà stata efficiente, finalizzata, cioè ad individuare, nuove attività e nuove tecniche operative criminali, tanto più congrue e tempestive risulteranno le attività di contrasto.

L'intelligence viene definita "informazione trattata" capace di fornire conoscenze aggiuntive o anche *knowledge*.

Il processo di intelligence inizia con l'*analisi*, ossia il ragionamento logico che, attraverso l'interpretazione dei fatti, perviene alla formulazione di *inferenze*.

L'attività di analisi può essere orientata verso obiettivi differenti. Si possono distinguere 3 tipi di analisi:

- Strategica
- Operativa
- Tattica

L'*analisi strategica* si interessa della fenomenologia criminale generale. Essa, individuando le linee evolutive e la vulnerabilità dei singoli fenomeni, è finalizzata a stabilire e predisporre adeguate misure di prevenzione e di contrasto. Tale tipo di analisi, al fine di sviluppare progetti di contrasto, si concentra su obiettivi a lungo termine controllando tendenze attuali ed emergenti,

mutamenti nell'ambiente criminale, minacce all'ordine ed alla sicurezza pubblica. L'attenzione di tale tipo di analisi è prevalentemente rivolta verso fenomeni sociali, economici, politici, ecc. non immediatamente collegati o collegabili con la realtà criminale.

L'*analisi operativa*, rispetto a quella strategica, presenta un raggio d'azione più circoscritto. Tale tipo di analisi, è orientata verso un obiettivo a breve termine o immediato ed è rivolta ad un particolare fenomeno ossia a fatti criminosi specifici. Essa è più aderente all'attività operativa e fornisce ipotesi che riguardano una particolare attività investigativa. Con tale tipo di analisi, studiando metodi, capacità, punti deboli, intenzioni, si possono sviluppare ipotesi riferite a individui o gruppi coinvolti nelle attività illecite oggetto di indagine.

L'*analisi tattica* è rivolta a neutralizzare peculiari attività criminali e spesso è incorporata nell'analisi operativa. La tattica è il ramo dell'arte militare che riguarda la condotta delle attività, terrestri, marittime, aeree, nel combattimento. Ispirandosi alla strategia, tale metodo di analisi viene impiegato in modo che si raggiunga l'obiettivo strategico.

Analizzeremo l'ultima tecnica, quella tattica, studiando i collegamenti tra entità (ad esempio persone, telefonini, organizzazioni) riguardanti la criminalità.

Nell'investigare possiamo suddividere le attività criminali in tre ambienti collegati fra loro, dal più generico al più specifico:

- Terrorismo (Al-Qaeda)
- Organizzazioni criminali (mafia, camorra)
- Piccoli reati

Questi ambienti sono rappresentati e studiati attraverso un SNA. Ciò consente di estrapolare fonte di informazione preziosa alle indagini dei giudici e di altri operatori.

Lo scopo principale è quello di facilitare il lavoro degli operatori del settore, semplificando le attività e i tempi di analisi.

Negli ultimi tempi l'analisi investigativa è stata influenzata dalle nuove tecnologie. In particolare, si è arricchito il dominio d'indagine a partire dalle informazioni provenienti dai mezzi telematici e dagli strumenti di telecontrollo. Questo ha provocato un aumento esponenziale della quantità di informazioni e di conseguenza della complessità delle elaborazioni.

Oggi giorno, per trattare e gestire in tempi brevi tutta questa mole di informazioni, c'è bisogno di sistemi informatici capaci di acquisire, immagazzinare e gestire tutti i dati che provengono da fonti eterogenee. Inoltre, per poter ottenere risposte in tempi utili è necessario che i software siano anche in grado di aiutare l'investigatore nel trovare ed evidenziare le informazioni rilevanti come supporto decisionale.

Sistemi informatici capaci di svolgere questi compiti dovrebbero essere altamente integrati in modo che l'efficacia delle elaborazioni vada di pari passo con la praticità di utilizzo, garantendo in ogni modo gli standard di qualità e sicurezza richiesti nel trattamento dell'informazione.

Discuteremo ora su quale informazione analizzare nella SN riferita alla criminalità.

2.2 Formalizzazione del problema

Dopo una panoramica introduttiva sulle caratteristiche di un SN e di come essa viene rappresentata da un grafo, discutiamo brevemente della rappresentazione dei dati memorizzati. In seguito, ne analizzeremo la struttura attraverso tecniche di GM.

Ritornando un attimo alla struttura dei dati, si considerano prima le *entità:* persone, cellulari, organizzazioni e dispositivi. Successivamente, saranno considerate altre entità che potrebbero essere introdotte: carte di credito, pos[3], e telepass[4].

Tutte queste entità sono connesse tra loro mediante relazioni tra persone (tramite telefono o altri strumenti), oppure sono connesse tra loro mediante relazioni di parentela tra persone (contatti tra persone e organizzazioni).

Il modello concettuale del database relazione è mostrato in Figura 2.1. Le due tabelle corrispondono ad *Entità* con attributi: Etichetta entità e Entità tipo; e *Relazione* con attributi: Src, Dest, Relazione tipo, Cardinalità e Etichetta relazione.

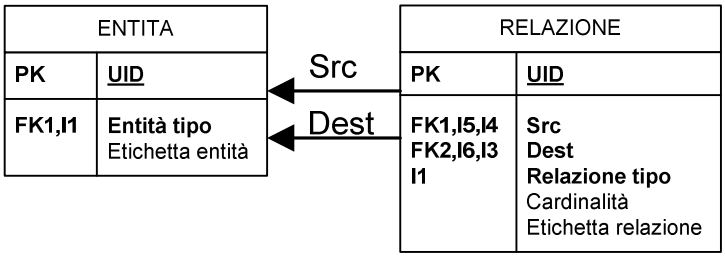


Figura 2.1 database

Le due tabelle sono collegate mediante due relazioni Src e Dest. Un esempio, di com'è realizzato il database è mostrato in Figura 2.2:

[3] Pos: dall'inglese Point Of Sale, letteralmente punto di vendita, è un'apparecchiatura automatica presso numerosi esercizi commerciali, mediante la quale è possibile effettuare, il pagamento dei beni acquistati o dei servizi ricevuti.
[4] Telepass: è un sistema di riscossione automatica del pedaggio autostradale.

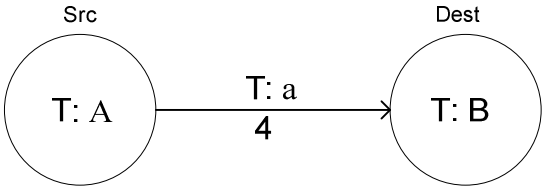


Figura 2.2 database

Il nodo Src ha un'entità di tipo *A* collegato, tramite la relazione di tipo *a* con una cardinalità pari a *4*, al nodo Dest con entità di tipo *B*.

Un'altra caratteristica del grafo è costituita dalla presenza di molti nodi isolati, non collegati a nessun altro nodo. Il grafo è molto denso. Esso presenta la maggior parte dei nodi collegati almeno da una relazione e una piccola minoranza di nodi isolati non collegati da nessuna relazione.

Il grafo considerato è multigrafo, quindi un nodo può avere più di una relazione (diverse), collegate ad un altro nodo.

In primo luogo, sostituiremo il termine SN riferendoci a un grafo, di grande dimensione. Tale dimensione complica notevolmente l'analisi, che presenta una complessità di calcolo elevata, come si vedrà in seguito NP-complete.

Mostriamo, un esempio (Figura 2.3) di un grafo tipico con più di 5.000 entità e altrettante relazioni, questo evidenzia la difficoltà nel gestire l'enorme mole di dati.

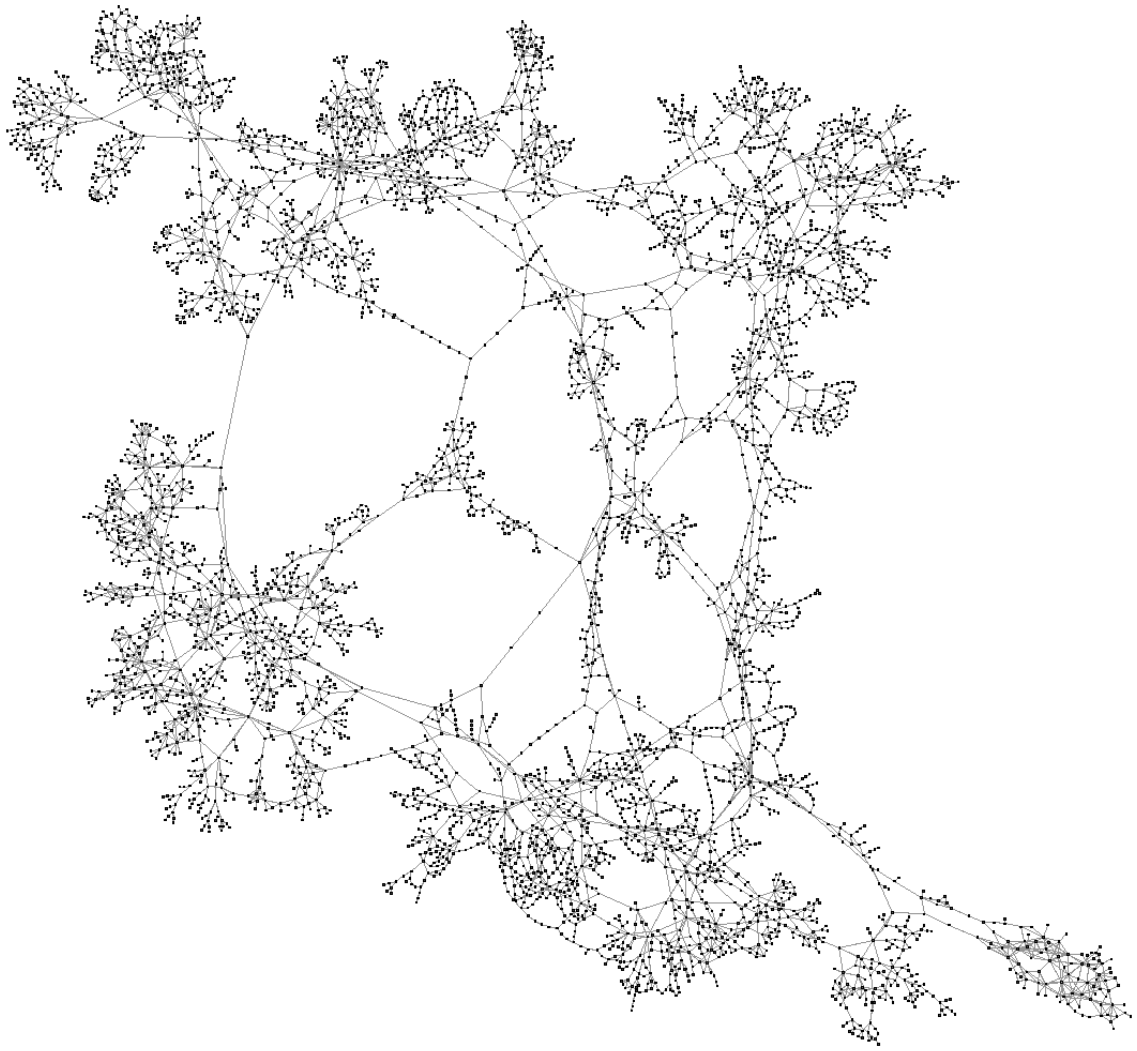


Figura 2.3 esempio di rete

Sono affrontati due problemi. Il primo, coinvolge la ricerca delle entità, con un ruolo di importanza cruciale all'interno del grafo, cioè cercare delle entità più significative. Il secondo problema è la presenza o meno di un pattern dato come input al grafo.

Tramite gli strumenti a disposizione del paragrafo precedente, sono formalizzati i problemi. Nel capitolo 3 sarà proposta una soluzione per entrambi.

2.2.1 Caratteristiche del grafo

Iniziamo ad analizzare i dati del grafo completo:

- **10.227** nodi.
- **79.975** relazioni.
- Denso.
- Presenta un gran numero di nodi esterni con una sola relazione al grafo. Significa che abbiamo dei nodi con un solo arco in uscita. Come mostra la Figura 2.4 il nodo isolato è di colore celeste.

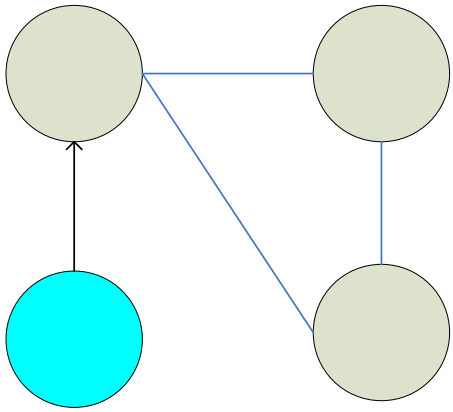


Figura 2.4

Notiamo che ci sono alcuni nodi con un grande numero di archi e un numero grandissimo di nodi con pochi archi come mostra la Grafico 2.1. Mostriamo, anche, le tabelle delle frequenze dell'entità (Tabella 2.1) e delle relazione (Tabella 2.2).

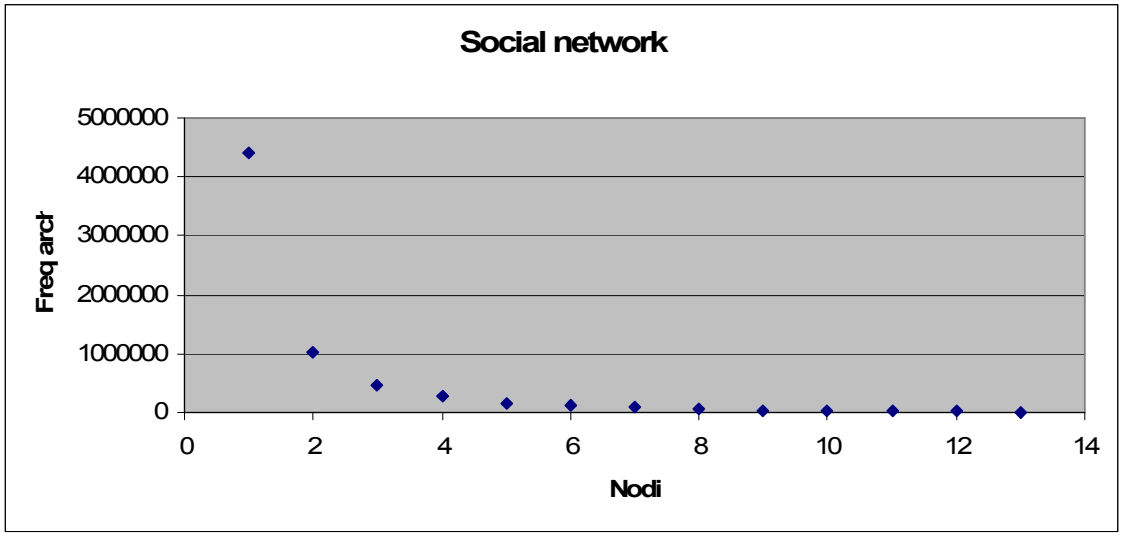


Grafico 2.1

Type	Num_Type	Freq_Type
Cellulari	3115	0.30
Persone	3010	0.29
Organizzazioni	2053	0.20
Dispositivi	1047	0.10
Luoghi	1002	0.10
Totale	**10227**	**1**

Tabella 2.1

Tipo	Frequenza	Frequenza%
Familiari	1311	0.0163926
Associazione_bus	2684	0.0335605
Personale	690	0.0086277
Proprietario	6619	0.0827634
Usato	6510	0.0814004
Ubicazione	1992	0.0249078
Traffico_telefonico	60169	0.7523476
Totale	**79975**	**1**

Tabella 2.2

2.2.2 Indice di centralità

Dopo un'analisi investigativa ho deciso di utilizzare un indice utilizzato per il web graph modificandolo secondo le esigenze del nostro problema. Ovviamente, per la mole di dati massiccia l'indice deve essere veloce e produrre risultati esatti.

La creazione di un indice nel grafo comporta una lista ordinata di nodi (entità), dal più importante al meno importante. L'importanza dei nodi si verifica per vari motivi (ad esempio, alcuni nodi possiedono più informazione utile e possono svolgere un ruolo cruciale all'interno della rete, tali nodi sono più importanti di altri).

Un modo per affrontare questo problema è considerare il PageRank di google mostrato in Figura 2.5. Poiché l'algoritmo di google assegna un indice di importanza alle pagine più significative, possiamo utilizzare lo stesso metodo, cioè considerare l'entità invece delle pagine.

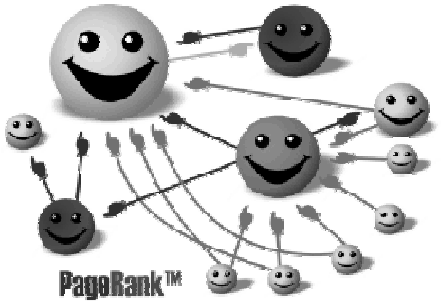


Figura 2.5 PageRank

Nell'esempio in Figura 2.5, si nota che il nodo più importante è quello più grande. Il criterio che utilizza google è quello di dare importanza al nodo che ha delle relazioni con altri nodi importanti. Questa operazione è eseguita ricorsivamente in modo da trovare tutte le pagine importanti.

È stato scelto questo criterio perché molto simile al nostro grafo, le pagine web rappresentano i nodi collegati tra di loro tramite relazioni.

Il grafo considera le seguenti misure:

- Gradi: i gradi di un nodo in uscita da un'entità.
- Cardinalità: peso o numero di volte che si ripete la relazione.
- Relazione tipo: Tipo della relazione (è residente, è usuraio di, è padre di, ...).
- Entità relazione: Tipo di Entità (persone, organizzazioni, dispositivi, ecc.).

Utilizzando le misure a disposizione, si cerca il nodo più importante, non soltanto attraverso il collegamento ai nodi, ma si considerano, anche la cardinalità, il tipo dell'entità e della relazione.

L'esempio in Figura 2.6 raffigura il grafo di un SN con misure sugli archi ed entità.

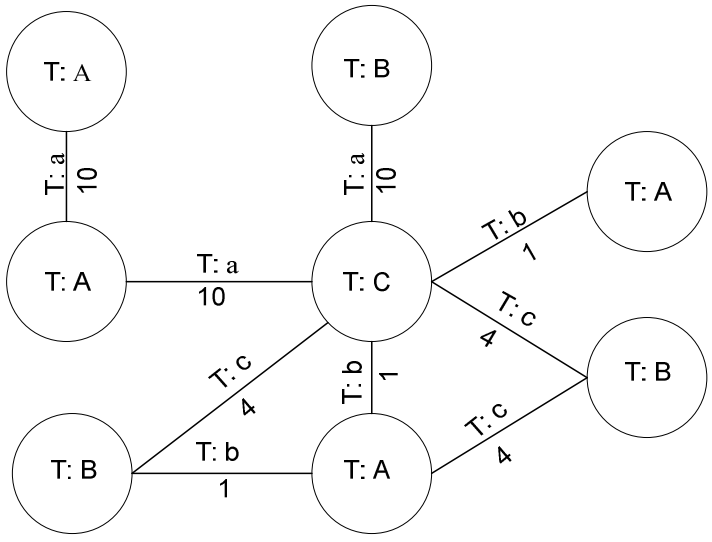


Figura 2.6 un esempio del grafo

Dalla Figura 2.6 si nota la presenza del nodo centrale (tipo *C*), collegato a tutti i nodi. In base al tipo del nodo è assegnato un valore, la stessa cosa occorre fare per le relazioni. Inoltre, le relazioni hanno un numero che indica la cardinalità, cioè la frequenza da un nodo ad un altro.

Infine, consideriamo la parametrizzazione dei valori dell'entità e delle relazioni. L'operazione è eseguita tramite un'assegnazione automatica dei valori dei tipi dell'entità e delle

relazioni, in base alla loro frequenza, oppure tramite un'assegnazione manuale impostando un valore più alto alle entità o relazioni, che sono più significative per l'utente.

2.2.2.1 Definizione dell'indice di centralità

In questa sezione utilizziamo un indice di *feedback* vedi [Abiteboul 03], cioè un nodo è più centrale se i suoi vicini sono centrali. Con il termine vicino si indicano i nodi figli. Il feedback è un metodo utile per l'analisi su *web graph*, il quale è definito come un insieme di pagine nel www connesse tramite *link web*. Tutti gli indici di centralità che utilizzano il feedback vengono risolti tramite un sistema lineare.

In questo paragrafo presentiamo il modello formale. Il "web" come un grafo *g*, dove le pagine web sono dei vertici e un collegamento di una pagina ad un'altra forma un arco diretto.

Un grafo *g* è *connesso* se, da una qualsiasi pagina possiamo arrivare ad un'altra pagina. Un grafo diretto *g* è detto *strettamente connesso* se per tutte le coppie di vertici i, j esiste un percorso diretto da i a j. Quando *g* non è connesso, ogni componente connesso potrebbe essere considerato separatamente.

Ora rappresentiamo il grafo come una matrice. Sia *g* un grafo diretto con *n* vertici. Fissando un ordine arbitrario dei vertici. Il grafo *g* può essere rappresentato come una matrice L[1..*n*, 1..*n*] tale che:

- L non è negativa, cioè $\forall i, \forall j, L[i, j] \geq 0$;
- $L[i, j] > 0$ se e solo se esiste un arco dal vertice i al vertice j.

Formalmente l'algoritmo dovrà eseguire il problema della creazione di *Indice di centralità*.
Problema: *Indice di centralità*.
Input: $G=(V, E, \mu, v)$.
Output: A vettore ordinato in base al criterio di importanza dei nodi $A=\{1\ldots|V|\}$.

2.2.3 Pattern grafo-sottografo

Dato in input un sottografo, verificare se tale sottografo è presente nel grafo globale. Nel nostro esempio il sottografo è realizzato con 3 entità (nodi) e 3 relazioni (arco), ogni entità e relazione ha un'etichetta e un tipo. Per l'*entità* il tipo equivale: dispositivi, organizzazioni, persone, numeri di telefono, luoghi; per la *relazione* il tipo equivale: parente, affiliato, amante, capo, utente, residente, contatto, ecc…

Anche in questo caso, il problema cruciale è l'elevata mole di dati da gestire. Quindi, occorre prendere delle precauzioni per la ricerca del pattern nel grafo completo.

Le misure considerate per l'analisi sono:

- Entità tipo: Tipo di Entità (persone, organizzazioni, dispositivi, ecc.).
- Relazione tipo: Tipo della relazione (è residente, è usurario di, è padre di, ...).
- Etichetta entità: Etichetta (testo corto) associata all'entità.
- Cardinalità: peso o numero di volte che si ripete la relazione.

Esempio di Figura 2.7 mostra un sottografo:

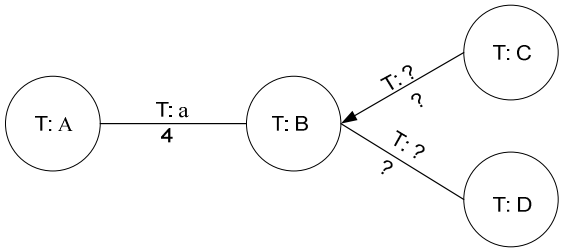


Figura 2.7 esempio sottografo

Dove T indica il tipo della relazione e dell'entità. Il carattere speciale *?* indica un qualsiasi tipo o un qualsiasi numero per la cardinalità.

2.2.3.1 Definizione del problema del pattern

Se consideriamo un sottografo con **10** entità possiamo calcolare le seguenti misure:

- *relazioni* $10(10 - 1) = \mathbf{90}$;
- $\deg_{-}(v) = \deg_{+}(v) = |E| = 10$;
- densità 90/(2*10*(10-1)) = 0,25;
- il sottografo è connesso sse ha 10 – 1 = 9 archi;
- Cayley n^{n-2}, cioè modi di costruire un albero etichettato 10^8 = 100.000.000.

Se vogliamo calcolare un isomorfismo tra due grafi con gli stessi nodi, ad esempio, con entrambi *n* nodi avremo *n*! possibilità di match, cioè match perfetto tra due grafi con gli stessi archi e nodi. Possiamo riscriverlo come:

$$n \times (n-1) \times (n-2) \times (n-3) \ldots \times 3 \times 2 \times 1$$

con *n* uguale a 9 e 10 avremo 9! = 362.880 e 10! = 3.628.800 possibilità di match.

Nel considerare un sottografo isomorfo, invece, bisogna indicare tutte le possibili soluzioni grafo-sottografo:

$$P(n,r) = \frac{n!}{(n-r)!}$$

Con *n* il numero dei nodi del grafo e *r* il numero dei nodi del sottografo.

La determinazione di una corrispondenza tra i nodi dei grafi, consistente con la struttura dei grafi stessi, è un problema dalla complessità esponenziale. Vale a dire con 10.000.000 di nodi, la ricerca di un sottografo con 10 nodi abbiamo:

$$P(10.000.000,\ 10) = \frac{10.000.000!}{(10.000.000-10)!} \approx \infty$$

L'isomorfismo grafo-sottografo è un problema NP-complete (anche se non è ancora dimostrato che l'isomorfismo sia un problema NP-complete). I tempi di matching degli algoritmi crescono esponenzialmente con la dimensione dei grafi. L'applicabilità è ristretta a problemi che coinvolgono grafi di piccole dimensioni.

L'isomorfismo grafo-sottografo è stato affrontato per primo da Ray e Kirsch (1957) i quali hanno proposto una tecnica di *backtracking* (breadth tracking). Successivamente fu affrontato da Sussengugh (1965) con il *partizionamento*. Poi è stato proposto l'*algoritmo* di Ullmann [Ullmann 76].

L'algoritmo di Ullmann è un algoritmo per la soluzione del problema dell'isomorfismo di sottografi. Il problema è NP-complete e l'algoritmo non fornisce una soluzione in tempo polinomiale. Tuttavia, esso utilizza tecniche di backtracking per diminuire il tempo effettivo di esecuzione. Tali tecniche, però, non hanno effetto sulla complessità asintotica, che rimane esponenziale. Infine, è proposto dal Prof. M. Vento, il quale propone un algoritmo molto efficiente [Foggia 01], che discuteremo nel prossimo capitolo.

Nel grafo di vaste dimensioni è molto difficile trovare la corrispondenza in tempo di calcolo ragionevole (quindi, come detto nel capitolo precedente (NP), bisogna ridurre la dimensione di ricerca e parametrizzare il pattern, in modo da abbassare, notevolmente, il numero di possibilità).

Il grafo è memorizzato all'interno di un database relazione abbastanza semplice (descritto nel paragrafo 2.2). Quindi è consigliabile seguire delle linee guida, ad esempio:

- prima, eseguire delle query per ridurre il numero di righe, cioè ridurre il numero dei nodi e delle relazioni del grafo.
- in seguito, occorre parametrizzare il pattern, cioè avere un pattern con delle etichette già definite in modo da poter selezionare dei nodi di partenza quando eseguiremo la ricerca.

Formalmente l'algoritmo dovrà eseguire il problema della ricerca di *Pattern grafo-sottografo*.

Problema: *Pattern grafo-sottografo*.

Input: $G=(V, E, \mu, v)$, grafo completo, $P=(V, E, \mu, v)$ sottografo.

Output: A vettore dei sottografi trovati:

- $A=[]$ se non esistono sottografi.
- $A=[1\ldots n]$ se esistono n sottografi.

3 Risoluzione

La soluzione dei problemi è eseguita tramite l'utilizzo di algoritmi presenti in letteratura, modificati secondo le caratteristiche dei problemi considerati.

Per la soluzione del problema dell'indice di centralità ho utilizzato On Line Page Importance Computation (OPIC). Trattasi di un algoritmo utilizzato per il calcolo del *PageRank*.

Per la soluzione del secondo problema, ricerca del sottografo, ho utilizzato l'algoritmo VF2.

3.1 On-Line Page Importance Computation

Come già detto, per la soluzione del primo problema utilizziamo l'algoritmo On-Line Page Importance Computation (OPIC) [Abiteboul 03]. Tale algoritmo assegna un valore d'mportanza alle pagine di internet. Poiché la struttura dati di internet è rappresentata da un grafo, possiamo adattare il PageRank al nostro algoritmo cambiando il criterio dell'importanza dalle pagine all'entità.

Per ogni entità, salviamo due valori: *cash e histosy.* Inizialmente, distribuiamo un po' di cash ad ogni nodo (per esempio, se ci sono n nodi, distribuiamo $1/n$ ad ogni nodo). Quando l'algoritmo è in esecuzione seleziona un nodo i. Il valore del cash di tale nodo è memorizzato nell'history del nodo i. Dopodichè, distribuiamo il cash del nodo visitato, in base ad un criterio d'importanza, tra tutti i nodi figli. Quindi, resetteremo il cash del nodo i a 0. Ciò avviene ogni volta che viene letto un nodo.

L'idea di base è quella di definire, in modo induttivo, il criterio d'importanza per i nodi e la generazione di un "punto fisso", per far terminare l'algoritmo. Se il grafo contiene n nodi, l'importanza è rappresentata come un vettore X con n che indica la dimensione. Consideriamo tre esempi, nei quali l'importanza è definita induttivamente dall'equazione $\bar{x}_{k+1} = L\bar{x}_k$:

- Un nodo è importante se punta ad un nodo importante. Allora L[i, j]=1 se esiste un arco tra i e j.
- Un "random walk" tutti i collegamenti in uscita hanno la stessa probabilità di essere scelti. L'importanza del nodo rappresenta la probabilità di un "random walk" sul web (grafo), allora L[i, j]=1/d[i] se e solo se esiste un arco tra i e j.
- L'ultimo metodo, assegna l'importanza in base al valore del nodo e della relazione del figlio. Moltiplicando il nodo e la relazione, il valore generato è dato da L[i,j]=val[i,j]/sum[i]

dove val[*i*,*j*] è la Cardinalità[5] dal nodo *i* al nodo *j*, mentre sum[*i*] è la somma della Cardinalità totale dei nodi adiacenti di *i*.

In tutti i casi, questo implica la soluzione induttivamente dell'equazione del tipo di $\bar{x}_{k+1} = L\bar{x}_k$, dove *L* è una matrice non negativa.

Per ogni matrice non negativa *L*, potrebbero verificarsi diversi problemi:

- Potrebbero prospettarsi diverse soluzioni. Questo accade quando lo spazio del vettore corrisponde al massimo eigenvalue[6] ha dimensione più grande di 1.
- Persino se esiste un'unica soluzione, l'iterazione potrebbe non convergere, se il grafo non ha alcuna delle proprietà desiderate.

Denotiamo C_t e H_t i valori dei vettori *C* e *H* alla fine del passo *t* dell'algoritmo. Il vettore C_0 denota il valore del vettore *C* inizializzato (1/*n*), quindi:

- $C_t[i]$ è il cash del nodo *i* al passo *t*.
- $H_t[i]$ è la storia del nodo *i* al passo *t*, cioè la somma dei cash precedenti del nodo.

Il totale della somma del cash è una costante. Mentre per ogni nodo *i*, *H*[*i*] tende ad infinito. Per ogni nodo letto *j* avremo:

$$H_t[j] + C_t[j] = C_0[j] + \sum_{(i\ ancester\ of\ j)} \left(\frac{L[i,j]}{out[i]} H_t[i]\right)$$

Teorema: Il limite di $H_t[j] = sum(H_t[j])$ è l'importanza del nodo (entità) *j*.

[5] N.B. in questo esperimento la cardinalitá tra un nodo e un altro ha un valore di 1

[6] Eigenvalue: corrisponde ad un valore scalare di un vettore denominato eigenvector. Un eigenvector è una trasformazione lineare di un vettore con valori diversi da 0, i quali quando subiscono una trasformazione, potrebbero cambiare la loro lunghezza, ma non la loro direzione.

3.1.1 Implementazione

Prima di eseguire l'algoritmo, sono svolte delle operazioni per utilizzare i dati in modo coerente. Proponiamo una serie di passi seguiti per tale processo:

1. Creazione del valore:
   - I nodi con più di un arco in uscita saranno accorparti in un unico arco con un solo valore.
   - È considerato solo il campo "Cardinalità".
2. L'algoritmo OPIC è modificato secondo i criteri di selezione del nodo e l'importanza del nodo.
3. La selezione del nodo è eseguita con il metodo Greedy.
4. Al nodo figlio è assegnato il valore in base alla percentuale dell'arco e del nodo che collega il padre al figlio.
5. Per dare importanza al nodo con più archi ho eseguito il quadrato del valore del cash ad ogni nodo.
6. L'algoritmo terminerà quando il limite massimo di iterazioni è raggiunto. Quindi quando t è maggiore di *Max_iter* l'algoritmo termina.

Consideriamo due vettori $C[1\ldots n]$ (cash) e $H[1\ldots n]$ (history) dove n indica il numero dei nodi. L'inizializzazione di C non incide sul risultato. L'history di un nodo è semplicemente un numero. Inoltre, è inserita una variabile G per ottimizzare l'esecuzione di $|H|=\sum_i H[i]$, cosicché ad ogni passo $G=|H|$. L'algoritmo è il seguente:

```
for each i let C[i] := val[i,j]*val[i,j];
%% il nodo j indica il figlio di i. Sono sommati tutti i
%% figli di i
for each i let tot := tot+C[i];
for each i let H[i] := 0;

G:=0;
do forever
%% metodo dei minimi quadrati e in base
%% ad un massimo di iterazione
     begin
     %%scegliere il nodo i in base al metodo Greedy;
     %% ogni nodo è selezionato infinite volte
     H[i] += C[i];
          %% accesso al disco per pagina
          for each child j of i,
          do C[j] += C[i](val_pon[i,j]/sum[i])%;
          %% Distribuzione del cash
          %% in base a L
     G += C[i];
     C[i] := 0;
     end
```

Ad ogni passo, una stima dell'importanza del nodo k al passo t è $\frac{H_t[k]}{G}$. Una stima più precisa è $\frac{H_t[k]+C_t[k]}{G+tot}$ dove *tot* indica il totale del cash.

Il vantaggio principale dell'algoritmo è quello di non memorizzare la matrice del grafo ma solo i vettori, necessari alla sua rappresentazione. Questo comporta una serie di benefici:

- Sono richieste meno risorse di memoria rispetto ad algoritmi standard.
- È facile da implementare.

Un esempio di come l'algoritmo funziona. In Figura 3.1 è rappresentato il grafo.

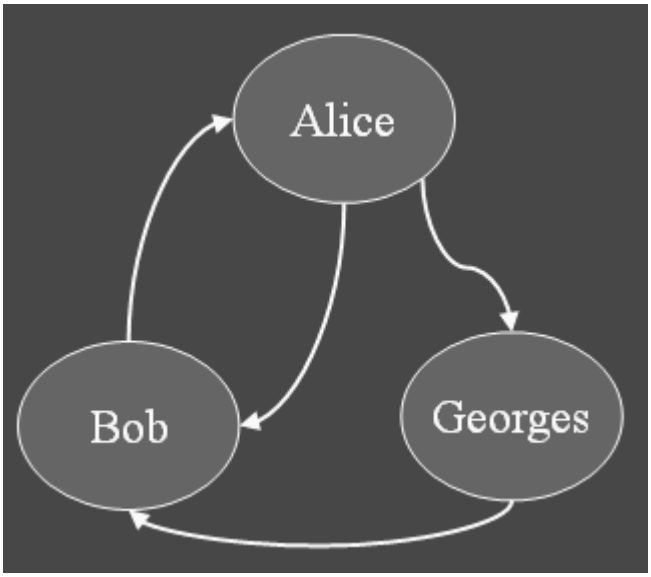


Figura 3.1 esempio dell'esecuzione OPIC

Esecuzione dell'algoritmo con il totale cash pari a 1.

[**Alice,Bob,Georges**] (**cash**)	[**Alice, Bob, Georges**] (**history**)
[0.33, 0.33, 0.33] (t=0)	[0, 0, 0]
Read <Alice>	Read <Alice>
[0, 0.50, 0.50]	[0.33, 0, 0]
Read Bob	Read Bob
[0.5, 0, 0.5]	[0.33, 0.50, 0]
Read Georges	Read Georges
[0.5, 0.5, 0]	[0.33, 0.50, 0.50]
Read Bob	Read Bob
[1, 0, 0]	[0.33, 1.0, 0.50]
Read Alice	Read Alice
[0, 0.5, 0.5]	**[1.33, 1.0, 0.50]**

Il nodo più centrale è Bob, seguono Alice e Georges. Se calcoliamo l'importanza avremo: 0.36 (Alice), 0.38 (Bob), 0.26 (Georges). L'importanza è stata calcolata con la seguente formula $\frac{H_t[k] + C_t[k]}{G + tot}$.

3.2 Pattern

Il problema è dato da un grafo e un sottografo (in input), occorre verificare se esiste una o più corrispondenze sottografo-grafo. Questa operazione è denominata pattern.

I suggerimenti, per trovare rapidamente una soluzione, sono: mappare solo nodi con le stesse etichette e con un numero compatibile di archi; iniziare dai nodi non usuali, cioè quello contenente, per esempio, un numero molto alto di nodi vicini; ridurre la complessità spaziale e computazionale ad una funzione polinomiale, preservando la generalità dell'algoritmo, per applicarla a grafi di grandi dimensioni.

I metodi proposti sono: *tecnica esaustiva*, *backtracking* e *programmazione dinamica*. Il primo viene abbandonato per il tempo di esecuzione irrealistica dell'algoritmo. Il secondo metodo (dove il pioniere è Jeff Ullmann, professore di Computer Science di Stanford), è stato migliorato da un altro algoritmo VF e in seguito VF2, che lo supera per la riduzione dell'uso della memoria. Questo algoritmo è proposto da M. Vento, professore di Computer Science dell'Università degli Studi di Napoli "Federico II" e da altri collaboratori. La programmazione dinamica è molto utile in teoria, ma molto difficile da costruire computazionalmente.

In seguito, illustreremo l'algoritmo VF, come funziona e come è implementato, con l'aggiunta di ulteriori consigli per diminuire lo spazio di ricerca. Poi continueremo con l'algoritmo VF2 adattandolo per la soluzione del problema.

3.2.1 Algoritmo grafo-sottografo

```
PROCEDURE Match(s)
INPUT: an intermediate state s; the initial state s0 has M(s0)=∅
OUTPUT: the mappings between the two graphs

IF M(s) covers all the nodes of G2 THEN
      OUTPUT M(s)
ELSE
Compute the set P(s) of the pairs candidate for inclusion in M(s)
      FOREACH (n, m)∈P(s)
            IF F(s, n, m) THEN
            Compute the state s' obtained by adding (n, m) to M(s)
                  CALL Match(s')
            END IF
      END FOREACH
      Restore data structures
END IF
END PROCEDURE
```

M(*s*) è il mapping parziale dello stato *s*, cioè l'insieme corrente delle coppie di nodi dei due grafi, che sono corrispondenti. P(*s*) è l'insieme delle coppie candidate ad estendere lo stato *s*. La costruzione di P(*s*) determina l'ordine in un cui sono considerati i possibili mapping. F(*s*, *n*, *m*) è il predicato di fattibilità (feasibility predicate), che indica se l'aggiunta della coppia (*n*, *m*) allo stato *s* produce un mapping inconsistente.

La scelta delle coppie dei candidati è basata sui *terminal sets*, cioè insiemi di nodi direttamente adiacenti a quelli già inseriti nel mapping parziale. Le coppie sono scelte in modo da assicurare che l'inconsistenza è riconosciuta il più presto possibile. Le coppie candidate sono generate in accordo ad un criterio di ordinamento, in modo da evitare che i diversi cammini di ricerca diano uno stesso mapping parziale.

Un insieme di condizioni per assicurare che il mapping trovato è un isomorfismo grafo-sottografo sono:

- Condizioni supplementari di *look-ahead*, per tagliare rami nell'albero di ricerca rimuovendo stati che non conducono ad una soluzione consistente.
- Per gli *attributed relational graphs*, le condizioni assicurano che i nodi e i rami che si trovano nel mapping hanno attributi compatibili.

Alcune condizioni controllano se i successori o i predecessori nei nodi che devono essere aggiunti siano già nel mapping parziale, o nei propri terminal sets.

Esempio: R_termin(s, n, m) $\Leftrightarrow$

$$(\text{Card}(\text{Succ}_1(n) \cap T^{in}_1(s)) = \text{Card}(\text{Succ}^2(m) \cap T^{in}_2(s))) \wedge$$
$$\wedge(\text{Card}(\text{Pred}_1(n) \cap T^{in}_1(s)) = \text{Card}(\text{Pred}^2(m) \cap T^{in}_2(s)))$$

cioè se il numero di predecessori (successori) di *n* che sono in $T^{in}_1(s)$ è uguale al numero di predecessori (successori) di *m* che sono in $T^{in}_2(s)$.

La struttura dati richiede una rappresentazione per ogni stato che permette lo svolgimento efficiente delle seguenti operazioni:

- Controllare se un nodo è nel mapping corrente e cercare il nodo corrispondente nell'altro grafo.
- Controllare se un nodo è nei terminal sets.
- Ricalcare i terminal sets dopo l'aggiunta di nuove coppie di nodi.

La nuova versione VF2, necessita di un tempo di calcolo poco maggiore, ma con un risparmio dello spazio di memoria:

- La rappresentazione dello stato è riformulata in modo da condividere la memoria tra i differenti stati.
- Questo richiede che dopo l'esplorazione di uno stato, le strutture dati condivise devono essere riportate alla condizione precedente l'esplorazione.

  In questo modo i requisiti di memoria possono essere ridotti a O(*n*).

Questo è realizzato nella sostituzione dei terminal sets, considerando la rappresentazione di uno stato come l'insieme dei nodi che sono nel mapping parziale o direttamente adiacenti ai nodi del mapping. Questi insiemi hanno le seguenti proprietà:

- Terminal sets vengono facilmente derivati dagli insiemi IN/OUT per differenza.
- Gli insiemi IN/OUT, così come l'insieme del mapping corrente, sono monotoni, nel senso che se un nodo *n* è in uno di questi insiemi allo stato *s*, esso rimarrà nell'insieme per tutti gli stati derivati da *s*.

La monotonia permette ad uno stato *s* di condividere le strutture dati usate per questi insiemi con tutti i suoi stati discendenti. Quindi, avremo due vettori core_1[n] e core_2[n], che codificano il mapping corrente. Quattro vettori in_1[n], out[n], in_2[n], out[n], che codificano gli insiemi IN/OUT. L'elemento *n* è 0, se il nodo n non è nell'insieme, altrimenti rappresenta il livello nel DFS (Depth-first state) in cui il nodo è stato inserito nell'insieme.

Quando l'algoritmo effettua il backtrack da uno stato *s* è facile ricondurre gli array ai valori che avevano precedentemente l'esplorazione dello stato *s*.

Quindi, gli array possono essere condivisi da tutti gli stati, questo implica la riduzione della memoria a O(*n*).

3.2.2 Vantaggi VF2

Alcuni vantaggi dell'algoritmo sono:

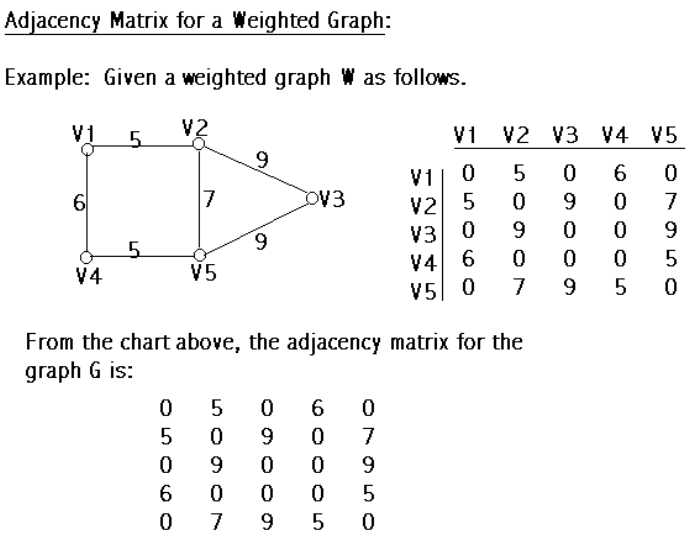
- Applicabile ad ogni tipo di grafo.
- Adatto per i grafi con attributi.
- Può calcolare sia l'isomorfismo tra grafi sia l'isomorfismo grafo-sottografo.
- Rappresenta nello spazio degli stati con una strategia di ricerca Depth-first.
- 5 regole di fattibilità per la riduzione dello spazio degli stati.
- VF2 è una versione ottimizzata di VF avendo una complessità lineare di memoria.

4 Esperimenti

In questo capitolo ho eseguito una serie di esperimenti, per testare gli algoritmi descritti nei capitoli precedenti. Per la implementazione degli algoritmi ho utilizzato due linguaggi di programmazione. Il *C#* per l'indice di centralità e il *C++* per la ricerca del pattern.

La nostra struttura di dati è di tipo multigrafo (cioè tra due nodi ci possono essere più relazioni), quindi saranno eseguite delle operazioni di pulizia dei dati. Prima di caricare i dati in memoria accorpiamo tutte le relazioni in un'unica relazione come descritto nel capitolo precedente. Inoltre, i nodi con relazioni con se stessi sono modificati: per ogni relazione è creato un nuovo nodo fittizio con la relazione in modo da avere sempre un grafo semplice.

La struttura utilizzata è una matrice di adiacenza mostrata in Figura 4.1.



	V1	V2	V3	V4	V5
V1	0	5	0	6	0
V2	5	0	9	0	7
V3	0	9	0	0	9
V4	6	0	0	0	5
V5	0	7	9	5	0

Figura 4.1

4.1 Indice di centralità

L'indice è testato con **377** nodi e **500** relazioni, il numero d'iterazioni è di **100000**. Il Grafico 4.1 evidenzia che i primi nodi hanno valori alti. Questo significa che hanno un elevato grado di nodi vicini. In seguito i valori diminuiscono e si avvicinano verso lo 0, sono i nodi con una relazione o con nessuna relazione

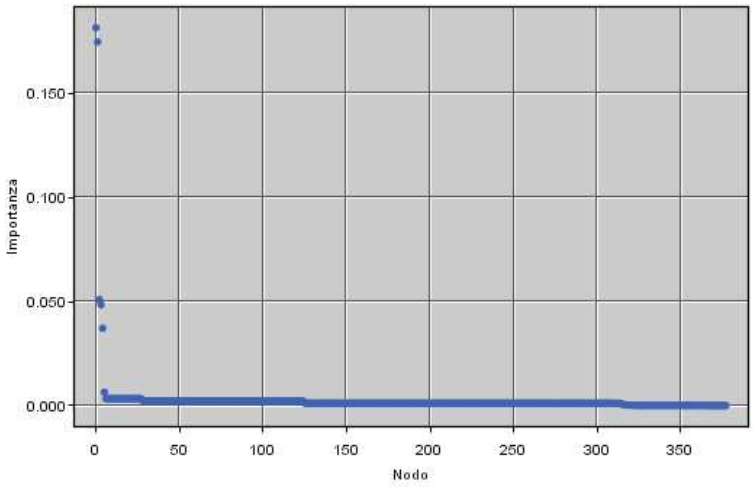


Grafico 4.1

Prendendo i primi 10 valori e ridisegnando il grafico (Grafico 4.2) viene visualzitta un'iperbole.

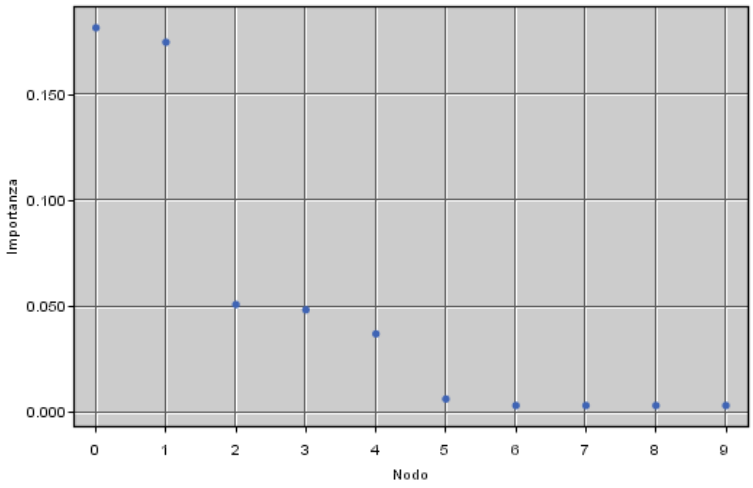


Grafico 4.2

Nella Tabella 4.1 è mostrato l'elenco delle entità importanti. Consideriamo le prime 8 entità poiché le successive entità hanno un valore molto basso. La prima colonna indica il numero

dell'entità, la seconda colonna indica il valore d'importanza dell'entità. Come notiamo i primi nodi hanno valori relativamente alti poiché come già ricordato sono quelli con più relazioni.

Num. Nodo	Valore
0	0.181490634839681
1	0.174680994494935
2	0.0509044888998443
3	0.0484735411021342
4	0.0371189910876978
5	0.00640039827624631
6	0.00332460911168258
7	0.00332460911168258

Tabella 4.1

Ora visualizziamo un campione di 53 entità su 377 entità per visualizzare i nodi più importanti dato che la visualizzazione di 377 è molto problematica. Ho realizzato un task in modo da creare un file con nodi e relazioni. I nodi sono etichettati da 1 a 53. Mostriamo in Figura 4.2 la visualizzazione del grafo tramite il programma open source *Pajek*. I primi 3 nodi (etichettati con 1,2,3) hanno valori alti e come possiamo vedere sono in una posizione centrale ad ogni componente.

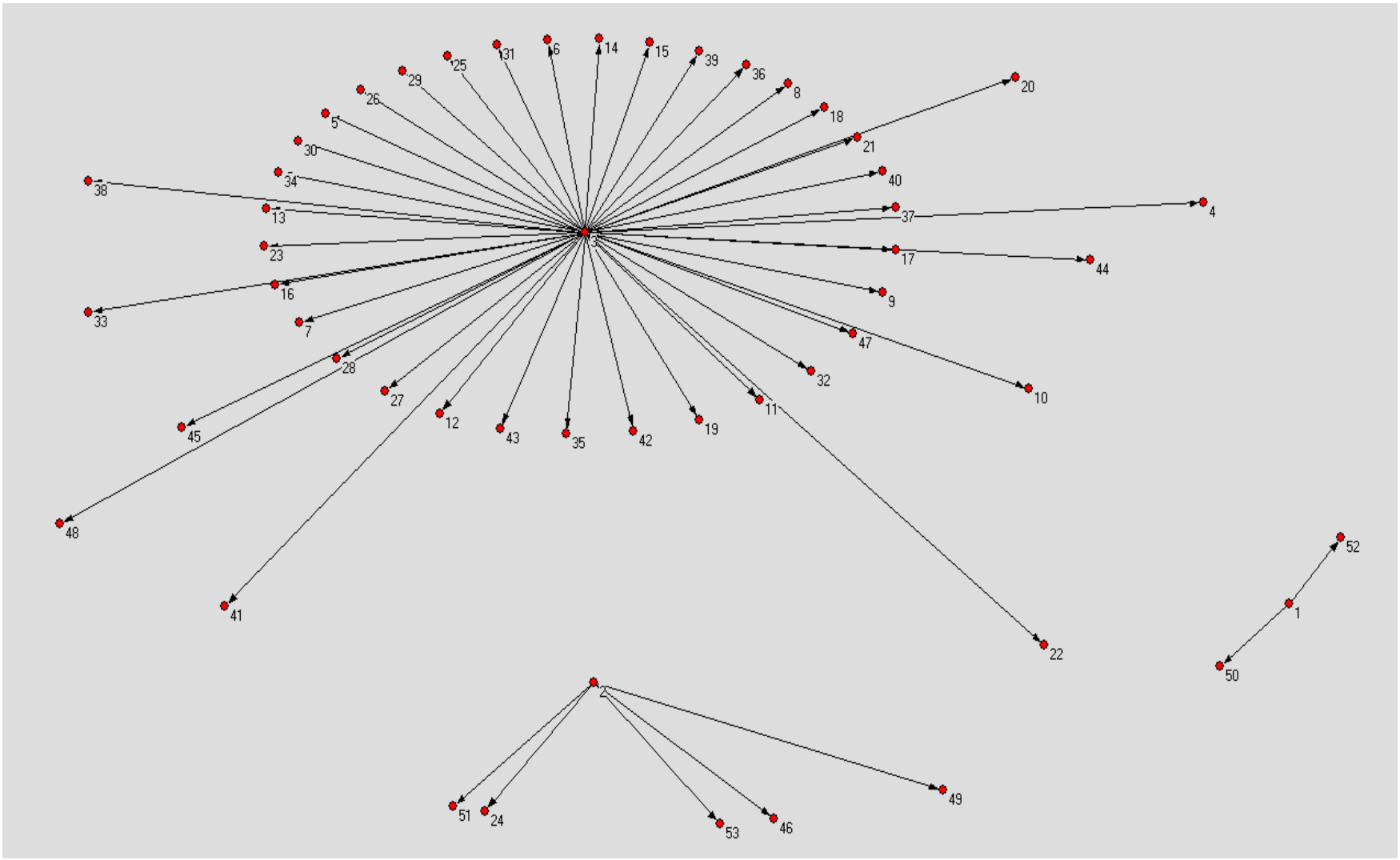

3
14
15
6
31
25
29
26
5
30
34
38
13
23
16
33
7
28
45
48
41
27
12
43
35
42
19
11
32
47
10
22
9
17
44
37
4
40
21
20
18
8
39
36
2
51
24
53
46
49
52
1
50

Figura 4.2

Ora ridisegniamo il Grafico 4.3 sul ascisse rappresentiamo i nodi mentre sul ordinate rappresentiamo il valore del nodo. In seguito tracciamo l'intercetta dei punti. Come era scontato appare ancora una parabola. Infine calcoliamo la funzione della parabola per evidenziare se l'indice segue una power law:

$$y = 0.138x^{-1.0142}$$

Prelevando l'esponente di tale equazione, cambiando il segno e aumentiamolo di una unità in modo da ottenere 2.0142 che indica il valore della power law.

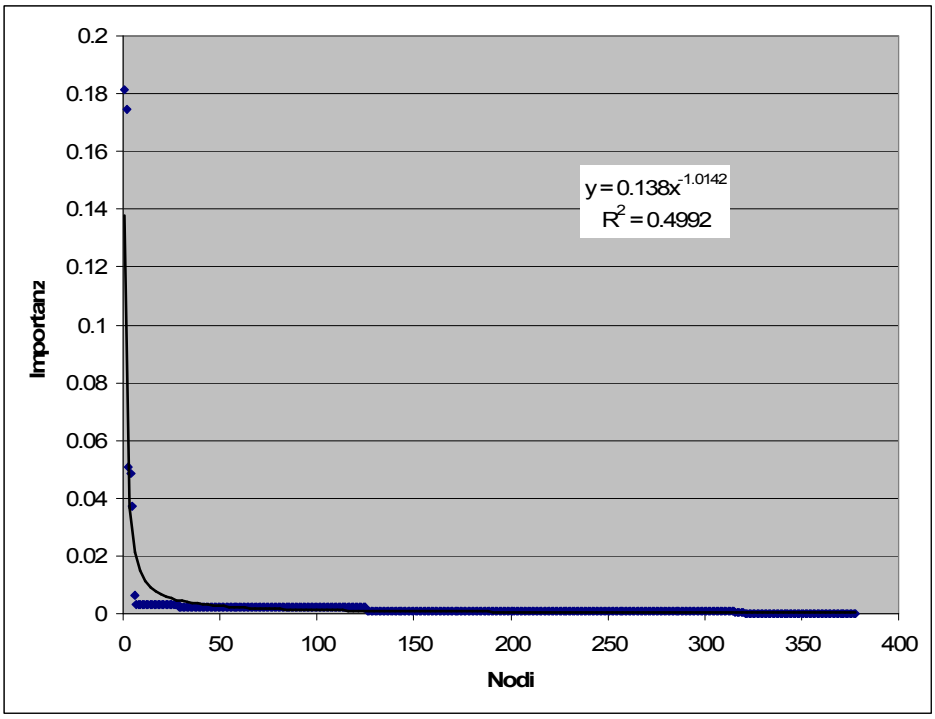


Grafico 4.3

Infine impostiamo gli assi cartesiani in modo logaritmo del Grafico 4.3 per ottenere l'ultimo grafico (Grafico 4.4).

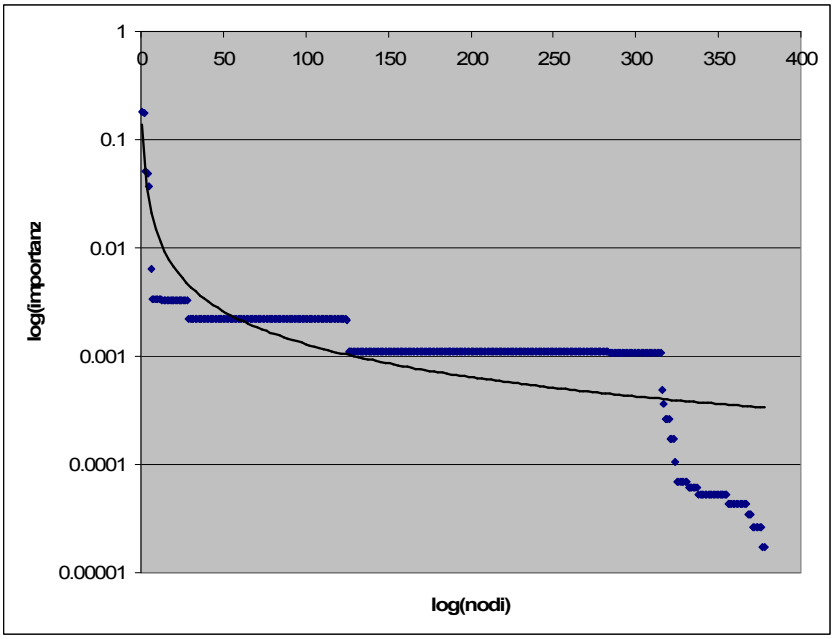


Grafico 4.4

4.2 Pattern

La libreria utilizzata in questo algoritmo ha il compito di prende in input i dati da due file. Il primo è il grafo completo. Il secondo è il sottografo da cercare. L'algoritmo può avere tre possibili soluzioni:

1. nessun sottografo trovato;
2. solo un sottografo trovato;
3. più di un sottografo trovato.

L'esperimento è stato svolto su un grafo denso con **10.227** nodi e **79.975** relazioni. L'algoritmo è testato con 5 configurazioni di sottografi. Nella ricerca del sottografo non è stato preso in considerazione il tipo dell'entità poiché poco rilevante per le analisi svolte.

Le configurazioni riguardano un sottografo presente nel grafo in modo da stabilire quanto tempo di calcolo impiega l'algoritmo.

Le etichette dell'entità hanno un valore simbolico crescente da 0 a 4. mentre le relazioni sono visualizzate nella Tabella 4.2.

Macroclasse	Classe	Name
1	1001	È parente di
	1002	is_grandparent
	1003	È zio di
	1004	È parente di
	1005	È sopato con
	1006	is_godparent
2	2007	È affiliato
	2008	È composto da
	2009	È pattern di
	2010	È capo di
	2011	is_dealer
3	3012	is_lover
	3013	È amico di
	3014	È nemico di
	4015	È proprietario di
5	5017	È utente
6	6018	È residente in
	6019	È domiciliato in
	6020	Ha un ufficio in
	6021	is_visitor
	6022	is_located
7	7023	Contatto
	7024	Chiama
	7025	È connesso da
8	8025	È la stessa entità

Tabella 4.2

Il primo sottografo denominato S1 (Figura 4.3) ha tutti gli archi pesati sia con la cardinalità e sia con il tipo della relazione rappresentato da un numero per semplicità.

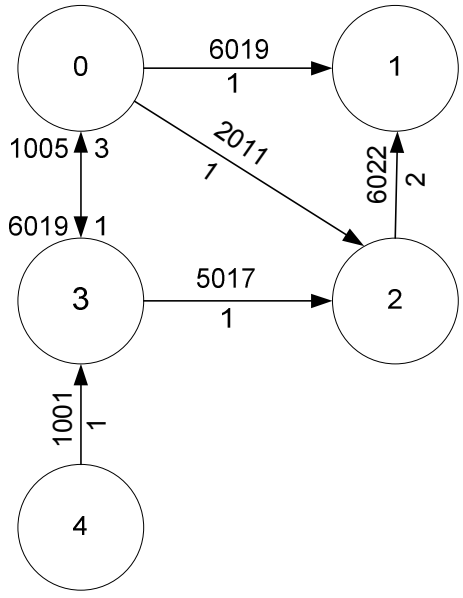


Figura 4.3

Il secondo sottografo denominato S2 (Figura 4.4) ha tutti gli archi pesati solo con l'attributo del tipo di relazione.

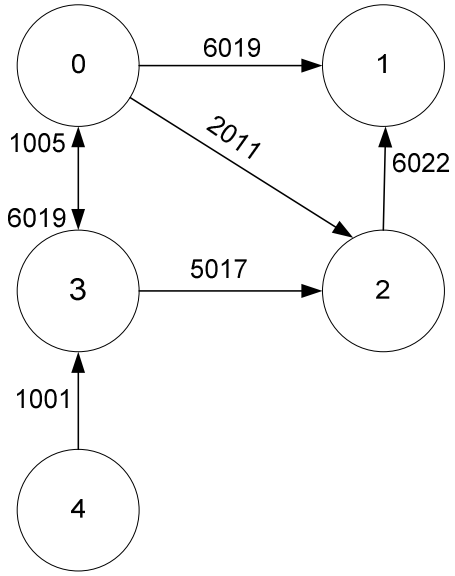


Figura 4.4

Il terzo sottografo denominato S3 (Figura 4.5) ha tutti gli archi pesati solo con l'attributo della cardinalità.

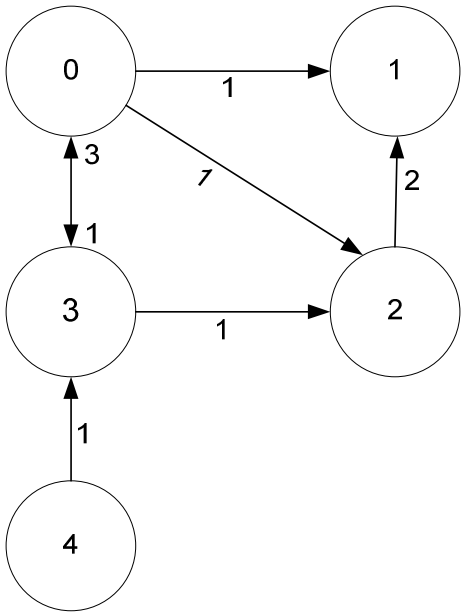


Figura 4.5

Il quarto sottografo denominato S4 (Figura 4.6) non ha nessuno attributo sugli archi.

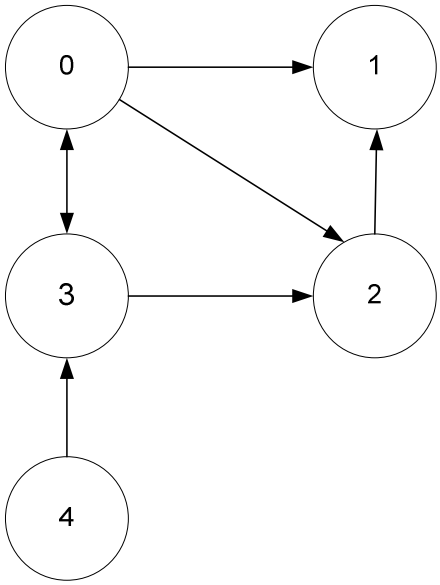


Figura 4.6

Il quinto sottografo denominato S5 (Figura 4.7) ha quattro nodi e non ha nessuno attributo sugli archi.

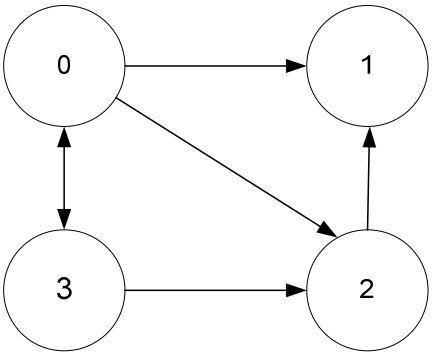


Figura 4.7

Dopo l'esecuzione dell'algoritmo con varie configurazioni, riporto qui in basso una Tabella 4.3 riassuntiva di tempi di calcolo e numeri dei risultati.

	Tempo	**Numero Risultati**
S1	~1 sec	1
S2	~1 sec	1
S3	30 sec	49.151
S4	3 ore 45 min	22.534.131
S5	35 min	968.947

Tabella 4.3

La ricerca del pattern è molto rapida nelle prime tre configurazioni. Poiché quando consideriamo un sottografo con attributi sono scartati i nodi che non portano a nessun match col grafo, quindi il tempo di calcolo è relativamente basso. Viceversa le due configurazioni finali, senza attributi, impiegano molto tempo nella ricerca del sottografo poiché bisogna considerare tutte le possibilità di match.

Le configurazioni con gli attributi portano ad una ricerca più specifica. Infatti nelle prime due configurazioni **S1** e **S2** i sottografi trovati sono entrambi **1**. Mentre, per le configurazioni senza attributi il numero dei risultati cresce notevolmente fino a **22.534.131**.

5 Conclusioni e sviluppi futuri

5.1 Conclusioni

Il problema trattato non è stato di semplice soluzione, sia per la complessità della struttura dei dati iniziali che per la loro dimensione.

È stato fatto uno studio della situazione iniziale, in particolare sui dati e sul dominio al fine di stabilire l'effettiva attuabilità di un eventuale processo di analisi. Oltre alla fattibilità in senso proprio si è anche cercato di quantificare lo sforzo necessario al raggiungimento di risultati soddisfacenti.

Il progetto ha richiesto inoltre solide conoscenze nell'ambito dello sviluppo software, capacità che in alcuni momenti del progetto sono state cruciali per risolvere particolari problemi non risolvibili in altra maniera. L'intero ciclo del lavoro è durato 10 mesi. In particolare durante i primi 4 si è investigato sul tipo di algoritmo da scegliere per la successiva implementazione. Gli altri 6 mesi sono stati spesi per l'implementazione del progetto, sviluppo il test del programma, e per la realizzazione della documentazione di tesi.

L'obiettivo della tesi era diviso in due parti: la prima nel ricercare un indice di centralità su una rete (grafo) di migliaia di nodi; la seconda parte trovare un pattern (sottografo) in una rete (grafo). Il problema è stato affrontato tramite la modifica di due algoritmi: il primo OPIC e il secondo VF2.

L'indice di centralità (OPIC) mostra come già detto una power law molto elevata pari a 2.0142. Questo è scontato perché ci sono pochi nodi con un valore altissimo e moltissimi con un valore basso. L'utilità di quest'indice è molto utile per trovare i primi 10 nodi in una rete di migliaia di nodi. La Figura 4.2 evidenzia appunto quanto detto, i primi 3 nodi (1, 2, 3) hanno un numero elevato di nodi principalmente il nodo 1 ha un numero di vicini pari a 43.

L'algoritmo per la ricerca (VF2) del pattern evidenzia che l'attributo principale, quando iniziamo la ricerca del sottografo, è l'attributo relazione, infatti, il tempo di calcolo è identico se inseriamo la cardinalità. Sono rilevanti S1 e S2, perché il tempo di calcolo è sotto ad un secondo di calcolo, in particolare si nota la differenza di tempo tra **S2** e **S3** il primo con solo attributo tipo relazione e il secondo con solo attributo cardinalità, quindi l'attributo tipo relazione è molto specifico mentre la cardinalità più generale.

5.2 Sviluppi futuri

Come ricordato in precedenza, il processo risultante dallo studio svolto è in gran parte riutilizzabile, e può essere adattato a tantissimi tipi d'analisi differenti, soprattutto nel campo del terrorismo. In realtà è possibile trovare in una rete di cellule terroristiche gli attori chiave della cellula in modo da poter annullare il suo potere e lo scambio d'informazione. Quindi, tale indice potrebbe essere modificabile attraverso l'utilizzo di nuovi parametri, come ricordato nel paragrafo 2.2, è possibile utilizzare altri tipi d'entità e di relazioni in modo conoscere altri tipi di misure nel modello e renderlo più vicino alla realtà.

Per grafici senza attributi è ancora un problema ostico risolvere l'isomorfismo del grafo quindi è necessario ricercare altre strategie per la ricerca di un pattern in tempi abbastanza brevi in modo da ottenere sempre risultati e informazione in tempo reale.

Bibliografia

[Abiteboul 03] Serge Abiteboul, Mihai Preda, Grégory Cobena, *Adaptive On-Line Page Importance Computation*, World Wide Web Conference , 2003.

[Agrawal 93] Rakesh Agrawal, Tomasz Imieliński, Arun Swami, *Mining association rules between sets of items in large databases*, In Proceedings of ACM SIGMOD International Conference on Management of Data, 1993.

[Albano 06] Antonio Albano, *Basi di dati di supporto alle decisioni*, Dicembre, 2006.

[Albert 00] R. Albert, A.-L. Barabási, *Topology of complex networks: Local events and universality*, Physical Review Letters, Vol. 85, N. 24, 2000.

[Albert 02] R. Albert, A.-L. Barabási, *Statistical mechanics of complex networks. Reviews of Modern Physics*, Vol. 74, N. 1, pp. 47–97, 2002.

[Asai 2002] T. Asai, K. Abe, S. Kawasoe, H. Arimura, H. Satamoto, S. Arikawa, *Efficient sub-structure discovery from large semi-structured data*, In Proceedings of 2002 SIAM International Conference Data Mining, Arlington, 2002.

[Barabási 99] A.-L. Barabási, R. Albert, *Emergence of scaling in random networks*, Science, Vol. 286, pp. 509–512, 1999.

[Barabási 02] A. L. Barabási, H. Jeong, Z. Neda, E. Ravasz, A. Schubert, T. Vicsek, *Evolution of the social network of scientific collaborations*, Physica A, Vol. 311, pp. 590–614, 2002.

[Boissevain 73] Boissevain J., J Clyde Mitchell, *Network analysis: Studies in Human Interaction*, The Hague-Mouton, 1973.

[Borgatti 03] S.P. Borgatti, P.C. Foster, *Journal of Management*, Vol. 29, N. 6, 2003, pp. 991-1013.

[Coscia 09] Michele Coscia, Fosca Giannotti, Ruggero G. Pensa, *Social Network Analysis as Knowledge Discovery process: a case study on Digital Bibliography*,

[Diane 06] Diane J. Cook, Lawrence B. Holder, *Mining Graph Data*, Hardcover, 2006.

[Fayyad 96] Fayyad U., Piatesky-Shapiro G., Smyth P, *From Data Mining to Knowledge Discovery: an Overview*, Advances in Knowledge Discovery and Data Mining, pp. 1-34, AAAI Press, 1996.

[Foggia 01] P. Foggia, C. Sansone, M. Vento, *An Improved Algorithm for Matching Large Graphs*, accepted for the 3rd IAPR-TC15 Workshop on Graph-based Representations, Ischia, 2001.

[Granovetter 83] Mark Granovetter, *The Strength of Weak Ties: A Network Theory Revisited*, Sociological Theory, Vol. 1, pp. 201-233, 1983.

[Han 00] J. Han, J. Pei, and Y. Yin, *Mining frequent patterns without candidate generation*, In Proceedings of 2000 ACM-SIGMOD International Conference on Management of Data, pp. 1–12, 2000.

[Inokuchi 00] Inokuchi, Akihiro and Washio, Takashi and Motoda, Hirosh, *An Apriori-Based Algorithm for Mining Frequent Substructures from Graph Data*, PKDD 2000: Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery, pp 13-23, Springer-Verla, 2000.

[Jensen 02] D. Jensen, J. Neville, *Data mining in social networks*, In Workshop on Dynamic Social Network Modeling and Analysis, 2002.

[Kleinberg 99] J. Kleinberg, S. R. Kumar, P. Raghavan, S. Rajagopalan, A. Tomkins, *The web as a graph: Measurements, models and methods*, In Proceedings of the International Conference on Combinatorics and Computing, pp. 1–17, 1999.

[Kleinberg 99] J. Kleinberg, *Authoritative sources in a hyperlinked environment*, Journal of the ACM, Vol. 46, N. 5, 1999, pp. 604-632.

[Kuramochi 01] Kuramochi, Michihiro and Karypis, George*, Frequent Subgraph Discovery*, ICDM 2001: Proceedings of the 2001 IEEE International Conference on Data Mining, pp 313-320, IEEE Computer Society, 2001.

[Lise 05] Lise Getoor, Christopher P. Diehl: *Link mining: a survey*, SIGKDD Explorations, Vol. 7, N. 2, 2005, pp. 3-12.

[Mendelzon 96]. A. Mendelzon, G. Michaila, e T. Milo, *Querying the world wide web*, In Proceedings of the International Conference on Parallel and Distributed Information Systems, pp. 80–91, 1996.

[Milgram 67] Milgram, S., *The small world problem*, Psychology Today 2, 1967, pp. 60-67.

[Moreno 34] Moreno, J. L., *Who Shall Survive?*, Washington: Nervous & Mental Disease Publishing Co., 1934.

[Newman 03] M. E. J. Newman, *The structure and function of complex networks*, SIAM Review, Vol. 45, 2003, pp. 167-256.

[Page 98] L. Page, S. Brin, R Motwani, T. Winograd, *The PageRank citation ranking: Bringing order to web*, Technical report, Stanford University, 1998.

[Pei 01] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, M.-C. Hsu, *PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth*, In Proceedings of 2001 International Conference on Data Engineering, pp. 215–224, Heidelberg, 2001.

[Pennock 02] D. M. Pennock, G. W. Flake, S. Lawrence, E. J. Glover, C. Lee Giles, *Winners don't take all: Characterizing the competition for links on the Web*, Proceedings of the National Academy of Sciences, Vol. 99, N. 8, pp. 5207–5211, 2002.

[Ravasz 03] E. Ravasz, A.-L. Barabasi, *Hierarchical organization in complex networks*, Physical Review E, Vol. 67, 2003.

[Simon 55] H. Simon, *On a class of skew distribution functions*, Biometrika, Vol. 42, pp 425–440, 1955.

[Ullmann 76] J. R. ULLMANN, *An Algorithm for Subgraph Isomorphism*, Journal of the ACM, 1976.

[Zaki 02] M. J. Zaki, *Efficiently mining frequent trees in a forest*, In Proceedings of 2002 ACM SIGKDD International Conference on Knowledge Discovery in Databases, pp. 71–80, 2002.

[Watts 99] Watts, D. J., *Small Worlds*, Princeton University Press, 1999.

[Watts 03] Watts, D. J., *Six Degrees: The Science of a Connected Age*, Hardcover, 2003.

[Yan 02] Yan, Xifeng and Han, Jiawei, *gSpan: Graph-Based Substructure Pattern Mining*, ICDM 2002: Proceedings of the 2002 IEEE International Conference on Data Mining, IEEE Computer Society, 2002.

Ringraziamenti

E cosi dopo sei lunghi anni sono arrivato a scrivere la pagina che conclude la mia tesi e con essa anche la mia carriera da studente.
Ho pensato molto a cosa scrivere in questo momento e per dire tutto servirebbero troppe parole che aggiungerebbero altre pagine ad una tesi già di per sé *troppo* lunga…
Tuttavia ci sono delle persone a cui voglio veramente dire grazie: primi tra tutti il Prof. il Prof. Albano e il Prof. Petreschi, Michele Coscia e il mio tutore didattico Ernesto Lastres che mi hanno fornito un aiuto costante nello sviluppo del progetto, guidandoci nei momenti più complessi e dandomi la possibilità di imparare e crescere professionalmente.
Come poi non ringraziare tutte le persone che in questi anni da studente mi hanno accompagnato. Un grazie grande così va anche a mia zie Lena, che mi ha aiutato ancora una volta nella redazione della tesi, e mia zia Nina che mi ha dato motivazione nel scrivermi all'università.
Un grazie a Pasquale, Claudio, Rosario, Carlo, Teresa e a tutti i compagnucci e compagnuccie di ESN Pisa che in modi diversi e in momenti diversi mi hanno reso la vita studentesca e personale piena e divertente e mi hanno aiutato a crescere vi voglio bene!
Grazie inoltre a tutte le persone che per me in questi anni sono stati un riferimento molto importante: Alessia, Ilenia, i compagni di erasmus, ma anche Elisa, Luca, Massimo l'ideatore delle " le bombe". Infine Luca, Antonio, Luigi, Idilio.
L'ultimo ringraziamento è per i colleghe e amici dell'OAMI che in questi mesi mi hanno supportato e aiutato nella tesi: Andrea, Elena, Roman, Angel, Jose, Miguel, Fran, e il mio jefe Fernando
Ultimo ringraziamento in ordine di scrittura, ma primo in ordine di importanza, va alla mia famiglia che in questi lunghi anni mi è sempre stata vicina fidandosi di me in modo incondizionato e permettendomi giorno per giorno di diventare la persona che sono: è a loro che dedico questo lavoro, questo giorno e tutte i miei (speriamo tanti) successi futuri.