Alma Mater Studiorum · Università di Bologna

FACOLTÀ DI SCIENZE MATEMATICHE, FISICHE E NATURALI
Corso di Laurea in Scienze e Tecnologie Informatiche

PROGETTAZIONE ED IMPLEMENTAZIONE DI STRUMENTI PER LA VALUTAZIONE DI RETI COMPLESSE CON PROPRIETÀ SCALE-FREE

Relazione Finale in Reti di Calcolatori

Relatore:
GABRIELE D'ANGELO

Presentata da:
GIULIO CIRNIGLIARO

Sessione II
Anno Accademico 2010-2011

Indice

Elenco delle figure

Elenco delle tabelle

Introduzione

Il cervello è una rete di cellule nervose connesse da assoni e le cellule stesse sono reti di molecole connesse da reazioni biochimiche. Anche le società sono reti di persone collegate da rapporti di amicizia, parentela e legami professionali. Su più larga scala, catene alimentari ed ecosistemi possono essere rappresentati come reti di specie viventi. E le reti pervadono la tecnologia: Internet, reti elettriche e sistemi di trasporto non sono che pochi degli esempi possibili. Anche il linguaggio che si sta usando in questo momento per veicolare questi ragionamenti a chi legge è una rete, fatta di parole connesse da relazioni sintattiche. [1]

A dispetto dell'importanza e della pervasività delle reti, gli scienziati hanno sempre avuto poca comprensione delle loro strutture e proprietà. In che modo le interazioni di alcuni nodi non funzionanti in una complessa rete genetica possono generare il cancro? Come può avvenire così rapidamente la diffusione in taluni sistemi sociali e di comunicazioni, portando ad epidemie di malattie e a virus informatici? Come possono alcune reti continuare a funzionare anche dopo che la maggioranza dei loro nodi ha, invece, smesso di farlo? [1]

Modelli elementari, come quello introdotto nel 1959 dai matematici Paul Erdös e Alfréd Rényi, hanno influenzato gran parte del nostro modo di concepire i sistemi interconnessi. Essi assumono che i sistemi complessi siano collegati fra loro casualmente, ipotesi adottata da sociologia, biologia ed informatica. Tali modelli hanno una capacità predittiva notevole poiché spiegano, ad esempio, perché ognuno di noi è separato da chiunque altro da sole

sei strette di mano, fenomeno osservato già nel 1929, ma che ottenne risonanza nelle scienze dure solo dopo che Duncan Watts e Stephen Strogatz ebbero esteso la sua portata al di là della sociologia. Tuttavia, l'innegabile successo dell'ipotesi di reti casuali aprì una questione fondamentale: le reti reali sono realmente casuali? Cioè, sistemi come la cellula od una società potrebbero funzionare senza soluzione di continuità se i loro nodi, molecole o persone, fossero connessi fra loro casualmente? [2] Se la topologia di queste reti realmente devia da quella di un grafo casuale, abbiamo bisogno di sviluppare strumenti e misure per cogliere in termini quantitativi i loro principi organizzativi di base. Negli ultimi anni abbiamo assistito ad enormi avanzamenti in questa direzione, spinti da diversi sviluppi paralleli. Innanzitutto, l'informatizzazione dell'acquisizione dei dati in tutti i settori ha portato alla nascita di grandi database sulla topologia di diverse reti reali. In secondo luogo, la sempre crescente potenza di calcolo ci ha consentito di investigare reti contenenti milioni di nodi, esplorando problematiche che non era stato possibile raggiungere prima. In terzo luogo, il lento, ma evidente crollo dei confini tra le discipline ha offerto ai ricercatori l'accesso a diversi database, consentendo loro di svelare generiche proprietà di reti complesse. [3] Così, nel 1998, Albert-László Barabási ed Eric Bonabeu, assieme a Hawoong Jeong e Réka Albert si lanciarono in un progetto di mappatura del World Wide Web, aspettandosi di trovare una rete casuale. Ecco perché: le persone seguono solamente i propri interessi quando decidono di quali siti mettere link nei propri documenti e, data la diversità degli interessi di tutti e lo straordinario numero di pagine tra cui si può scegliere, lo schema delle connessioni risultante dovrebbe apparire abbastanza casuale. Le misurazioni, tuttavia, non rispettarono le aspettative. Il software sviluppato per questo progetto saltava da una pagina web all'altra e raccoglieva tutti i link possibili. Benché questo robot virtuale raggiungesse solo una piccola frazione dell'intero Web, la mappa generata rivelò qualcosa di abbastanza sorprendente: una piccola frazione di pagine estremamente connesse mantengono sostanzialmente unito l'intero World Wide Web. Più dell'80% delle pagine sulla mappa avevano

meno di 4 collegamenti ed una ristretta minoranza, meno dello 0,01% di tutti i nodi, ne aveva più di 1000. [1] Una rete con tale configurazione viene definita *scale-free* (ad invarianza di scala) poiché, se si considera la relazione tra il numero di nodi ed il numero delle loro connessioni si osserva che il suo grafico è di tipo esponenziale negativo, e quindi invariante per cambiamenti di scala [WIKI], e la legge matematica seguita è detta *power law* (letteralmente "legge di potenza"). I nodi con elevato numero di connessioni vengono definiti *hub* (letteralmente "mozzo" o "fulcro"); la loro presenza, ovviamente, risulta incompatibile con la teoria delle reti casuali.
Negli ultimi anni, i ricercatori hanno rilevato la struttura *scale-free* in un'incredibile varietà di sistemi, anche all'interno delle reti sociali. Una collaborazione fra scienziati dell'università di Boston e dell'università di Stoccolma, ad esempio, ha mostrato che una rete di relazioni sessuali in un campione di persone in Svezia seguiva una *power law*: benché la maggior parte degli individui avesse solo pochi partner sessuali durante la propria vita, una piccola parte (gli *hub*) ne avevano centinaia. [1] Altri esempi di reti *scale-free* possono essere le pubblicazioni scientifiche (nodi) e le citazioni (archi) che le connettono, gli attori di Hollywood (nodi) e le partecipazioni nei film (archi). Molte altre sono le reti che si è scoperto soddisfare tale proprietà; esse sono riportate nella tabella 1 (tratta da [1]).
Nel momento in cui si riesce a comprovare la topologia *scale-free* di alcune reti di importanza cruciale (ad es. Internet) , sorge spontanea una riflessione: esattamente quanto sono affidabili queste tipologie di reti? La buona notizia è che i sistemi complessi risultano essere sorprendentemente elastici nei confronti di rotture accidentali. Infatti, benché centinaia di router in Internet soffrano quotidianamente malfunzionamenti, la rete raramente subisce grandi fratture. Un grado similare di robustezza caratterizza i sistemi viventi: le persone raramente si accorgono delle conseguenze di migliaia di errori nelle loro cellule, che vanno da mutazioni a proteine non ripiegate correttamente. Qual è l'origine di tale robustezza? L'intuizione ci dice che il guasto di un numero sostanziale di nodi produca come risultato un'inevitabile fram-

RETI	NODI	COLLEGAMENTI
Metabolismo cellulare	Molecole coinvolte nel bruciare nutrienti per produrre energia	Partecipazione alla stessa reazione biochimica
Hollywood	Attori	Partecipazione allo stesso film
Internet	Router	Linee ottiche ed altre connessioni fisiche
Rete proteica di regolazione	Proteine che aiutano a regolare le attività di una cellula	Interazioni fra proteine
Collaborazioni di ricerca	Scienziati	Essere co-autore di pubblicazioni
Relazioni sessuali	Persone	Rapporti sessuali
World Wide Web	Pagine web	URL*

**Uniform Resource Locator*

Tabella 1: Esempi di reti *scale-free*

mentazione della rete. Questo è sicuramente vero per le reti casuali: se una rilevante frazione dei nodi viene rimossa, questi sistemi si spezzano in piccole isole non comunicanti. Le simulazioni di reti *scale-free*, tuttavia, mostrano un risultato differente: possono guastarsi fino all'80% dei router in Internet scelti in modo casuale, ma i rimanenti formano ancora un raggruppamento compatto, in cui c'è ancora un percorso fra una qualsiasi coppia di nodi. È ugualmente difficile disgregare la rete di interazione proteica di una cellula: le misurazioni indicano che anche dopo aver introdotto una grande quantità di mutazioni casuali, le proteine non affette continuano a lavorare in sinergia. [1] La presenza di nodi-chiave quali gli *hub*, però, porta ad un rovescio della medaglia: in una serie di simulazioni si è osservato che la rimozione di una minima parte di *hub* da Internet frammenta il sistema i piccolissimi

gruppi di router isolati senza speranza. Similmente, esperimenti sul lievito hanno mostrato che la rimozione delle proteine maggiormente connesse ha una possibilità di uccidere l'organismo molto maggiore di quella che si avrebbe rimuovendo altri nodi. [1] Affidarsi agli *hub* può essere vantaggioso o no, a seconda del sistema. Certamente, la resistenza a guasti casuali è una buona notizia sia per Internet che per la cellula. Per quest'ultima, inoltre, ciò comporta anche la possibilità di studiare nuove strategie per selezionare gli obiettivi dei medicinali, portando potenzialmente a cure che uccidano solamente batteri o cellule dannose, colpendo selettivamente i loro *hub*, lasciando intatto il tessuto sano. Tuttavia, la capacità di un piccolo gruppo di hacker ben informati di interrompere l'intera infrastruttura delle comunicazioni colpendo gli *hub* è ragione di maggior preoccupazione. [1]
Lo studio delle reti scale-free ha condotto, inoltre, ad un'analisi di diversi fenomeni che le interessano dall'interno: come si diffondono facezie, mode e malattie nelle reti sociali? Come si propagano i guasti lungo le linee elettriche sul larga scala? In che modo si trasmettono i virus nella rete Internet? [4] A queste domande hanno provato a dare risposta due scienziati, Alessandro Vespignani e Romualdo Pastor-Satorras, che hanno studiato analiticamente e numericamente un modello dinamico su larga scala sulla diffusione di epidemie in reti complesse [4]. Per fare ciò è stato utilizzato il modello standard *susceptible-infected-susceptible* (SIS), letteralmente "recettivo-infetto-recettivo". Ogni nodo della rete rappresenta un individuo ed ogni collegamento è una connessione lungo cui l'infezione può diffondersi ad altri individui. Il modello SIS si basa su una caratterizzazione grezza degli individui nella popolazione. Più precisamente, essi possono trovarsi solamente in due stati discreti, ossia *susceptible*, "in salute", ed *infected*, "infetto". Questi stati trascurano completamente i dettagli del meccanismo di infezione all'interno di ogni individuo. Ad ogni *time step*, ogni nodo in salute viene infettato con probabilità ν se è connesso ad uno o più nodi infetti. Allo stesso tempo, i nodi infetti vengono curati e tornano nuovamente in salute con probabilità δ, definendo un tasso effettivo di diffusione $\lambda = \nu/\delta$ (senza perdita

di generalità si pone $\delta = 1$). Gli individui passano stocasticamente attraverso il ciclo *susceptible-infected-susceptible*, da cui il nome del modello. Tale modello non prende in considerazione la possibilità di rimozione di individui a causa di morte od immunizzazione. La topologia della rete che chiarisce le interazioni fra gli individui è di primaria importanza nel determinare molte delle caratteristiche del modello. Nelle topologie classiche, il risultato più significativo è la generale previsione di una soglia epidemica λ_c diversa da zero. Se il valore λ è al di sopra della soglia, ossia $\lambda \geq \lambda_c$, l'infezione si diffonde e rimane persistente nel tempo. Se, invece, $\lambda < \lambda_c$, l'infezione si estingue con velocità esponenziale. [8] Da questi studi è emerso che, mentre per una classica rete esponenziale la soglia epidemica è una costante positiva, per una larga classe di reti *scale-free* essa tende a zero. In altre parole, le reti *scale-free* sono prone alla diffusione ed alla persistenza delle infezioni, a prescindere dal tasso di diffusione degli agenti epidemici stessi. Ciò implica che i virus informatici possono diffondersi in lungo ed in largo su Internet con la sola infezione di una piccola frazione dell'enorme rete. Fortunatamente, ciò è bilanciato dalla prevalenza esponenzialmente bassa e dal fatto che ciò è vero solamente per un intervallo di tassi di diffusione molto piccoli ($\lambda \ll 1$), che tende a zero. [4]

Capitolo 1

Modelli teorici

1.1 Cenni sul modello Erdös-Rényi

Per oltre un secolo, la modellazione di sistemi e processi fisici e non fisici è stata eseguita con l'implicita assunzione che gli schemi di interazione fra gli individui del sistema o processo sottostante potessero essere incorporati all'interno di una struttura regolare, addirittura universale. Come già precedentemente rilevato, nei tardi anni '50 due matematici, Paul Erdös e Alfréd Rényi, mossero un enorme passo in avanti nella teoria matematica classica dei grafi. Descrissero, infatti, la topologia di una rete complessa mediante il concetto di *grafo casuale*. [4] Per capire il concetto a livello intuitivo, è sufficiente immaginare di avere una grande quantità ($N \gg 1$) di bottoni sparpagliati sul pavimento; si congiungono, poi, con la stessa probabilità p, tutte le coppie di bottoni mediante un filo. Il risultato è un esempio fisico di un grafo casuale del modello Erdös-Rényi (ER) con N nodi e circa $pN(N-1)/2$ archi. [4]
Da un punto di vista formale, esistono due diverse formulazioni di tale modello: si considera un grafo $\Gamma_{n,N}$ che ha n vertici etichettati $P_1, P_2, ..., P_n$ e N archi. Si suppone che questi N archi siano scelti casualmente fra gli $\binom{n}{2}$ archi possibili, così che tutte le $\binom{\binom{n}{2}}{N} = C_{n,N}$ scelte possibili siano equiprobabili. Così, se $G_{n,N}$ denota uno qualsiasi dei $C_{n,N}$ grafi formati dagli n

punti etichettati ed aventi N archi, la probabilità che il grafo casuale $\Gamma_{n,N}$ sia identico a $G_{n,N}$ è $1/C_{n,N}$. Un'altra formulazione equivalente è la seguente: si suppone che siano dati n vertici etichettati $P_1, P_2, ..., P_n$. Si scelga un arco a caso fra gli $\binom{n}{2}$ archi possibili, cosicché tutti tali archi siano equiprobabili. Successivamente si scelga un altro arco fra i rimanenti $\binom{n}{2} - 1$ e si continui questo processo cosicché, se già fissati k archi, uno qualsiasi degli $\binom{n}{2} - k$ archi rimanenti abbia ugual probabilità di essere scelto come successivo. [5]

1.2 Dalle reti casuali al modello Barabási-Albert

Benché l'intuizione chiaramente suggerisca che molte reti complesse presenti nella vita reale non siano né completamente regolari, né completamente casuali, il modello ER è stato l'unico approccio sensato e rigoroso che ha dominato i ragionamenti degli scienziati sulle reti complesse per quasi mezzo secolo, essenzialmente a causa dell'assenza di elevata potenza computazionale ed informazioni topologiche dettagliate riguardo reti reali su larga scala. [4] L'aumento della potenza dei calcolatori che ha avuto luogo negli ultimi anni, però, ha portato alla possibilità di acquisire grandi moli di dati riguardo la topologia di reti reali di vaste dimensioni. Un enorme passo verso la comprensione delle caratteristiche generiche dello sviluppo delle reti è stata la recente scoperta di un sorprendente grado di auto-regolamentazione che caratterizza le proprietà delle reti complesse su larga scala. Esplorando alcuni grandi database che descrivono la topologia di grandi reti che spaziano dal World Wide Web (WWW) agli schemi di citazioni nelle pubblicazioni scientifiche, recentemente Albert-László Barabási e Réka Albert hanno dimostrato che, indipendentemente dalla natura del sistema e dall'identità dei suoi costituenti, la probabilità $P(k)$ che un vertice nella rete sia connesso a k altri vertici si comporta come una *power law*, seguendo $P(k) \sim k^{-\gamma}$. [6] Questo risultato indica che le reti di grandi dimensioni si auto-organizzano in uno stato privo di scala (da cui il termine *scale-free*), caratteristica non prevista dai modelli di reti casuali esistenti. [7] Una caratteristica comune a tutti questi modelli

(tra cui l'ER) è, infatti, che prevedono che la distribuzione della connettività dei nodi $P(k)$ abbia un andamento esponenziale [6], con un picco ad un valore medio [4]. Per comprendere l'origine di questa discrepanza, Barabási ed Albert hanno rilevato che vi sono due aspetti delle reti reali che non sono incorporati all'interno di tali modelli. [6] In primo luogo, essi assumono che si parta con un numero fisso di vertici (N) che vengono connessi o ricablati casualmente, senza modificarne la quantità. Al contrario, molte reti presenti nel mondo reale sono aperte e si formano mediante la continua aggiunta di nuovi nodi al sistema, ossia N cresce durante il periodo di vita della rete. Per esempio, la rete di attori cresce con l'aggiunta di nuovi attori al sistema, il WWW cresce esponenzialmente nel corso del tempo con l'aggiunta di nuove pagine web, e la letteratura scientifica cresce con la pubblicazione di nuovi *paper*. Conseguentemente, una peculiarità comune a questi sistemi è che la rete si espande continuamente con l'aggiunta di nuovi nodi, che sono connessi a quelli già presenti nel sistema. [7]

In secondo luogo, i modelli di reti casuali assumono che la probabilità che due nodi siano connessi fra loro sia casuale ed uniforme. Tuttavia, molte reti reali mostrano connettività preferenziale. Per esempio, è più probabile che ad un nuovo attore sia assegnato un ruolo di supporto ad attori più conosciuti e affermati. La probabilità che un nuovo attore, quindi, reciti con uno più affermato è molto maggiore rispetto a quella di recitare assieme ad altri attori poco noti. In modo analogo, è più probabile che una pagina web appena creata contenga collegamenti a documenti ben conosciuti con connettività già alta, ed è più probabile che un in un nuovo manoscritto sia citato un *paper* molto conosciuto e già frequentemente citato, rispetto ad un altro meno conosciuto e, quindi, meno citato. Questi esempi indicano che la probabilità con cui un nuovo nodo si connette a quelli già esistenti non è uniforme; c'è una probabilità maggiore che sia connesso ad un nodo che ha già una consistente quantità di connessioni. [7]

1.3 Il modello Barabási-Albert (BA)

Alla luce di queste considerazioni, Barabási ed Albert hanno sviluppato un modello per la generazione di reti con proprietà *scale-free*. Per incorporare il carattere di crescita della rete, si inizia con un piccolo numero (m_0) di nodi e ad ogni time step si aggiunge un nuovo nodo con m ($\leq m_0$) archi che lo connettono ad m diversi nodi già presenti nel sistema. Per includere, invece, il meccanismo di annessione preferenziale, si assume che la probabilità Π che un nuovo nodo sia connesso al nodo i dipenda dal grado k_i di quel nodo (si definisce grado di un nodo in un grafo il numero di archi in esso incidenti), cosicché $\Pi(k_i) = k_i / \sum_j k_j$. [7] Come già enunciato, la probabilità $P(k)$ che un vertice nella rete sia connesso a k altri vertici si comporta come una *power law* ed è, inoltre, indipendente rispetto al tempo (e, conseguentemente, rispetto alla dimensione del sistema) poiché, a dispetto della continua crescita, il sistema organizza sempre se stesso in uno stato *scale-free* stazionario. [7] Riassumendo, l'algoritmo BA può essere schematizzato come segue:

1. **crescita**: si inizia con un piccolo numero (m_0) di nodi e ad ogni *time step* un nuovo nodo viene introdotto nel sistema e connesso ad $m(\leq m_0)$ nodi già esistenti;

2. **annessione preferenziale**: la probabilità Π_i che un nuovo nodo sia connesso al nodo i (uno degli m nodi già esistenti) dipende dal grado k_i del nodo i, cosicché risulti $\Pi(k_i) = k_i / \sum_j k_j$.

1.3.1 Algoritmo BA deterministico

Oltre all'algoritmo sopra descritto, lo stesso Barabási in [9] ne descrive uno per la generazione di grafi *scale-free* in maniera deterministica. La rete viene costruita in modo iterativo, con iterazioni che ripetono e riutilizzano gli elementi generati nei passi precedenti, nel modo seguente:

passo 0: si inizia con un nodo singolo, che viene designato come radice del grafo;

passo 1: si aggiungono due ulteriori nodi, entrambi connessi alla radice;

passo 2: si aggiungono due unità di tre nodi, ognuna delle quali perfettamente uguale alla rete creata nell'iterazione precedente (passo 1), connettendo ognuno dei nodi inferiori di queste due unità alla radice (vedi figura 1.1). Ossia, la radice guadagnerà quattro nuovi collegamenti;

passo 3: si aggiungono due unità con nove nodi ciascuna, identiche alle unità generate nell'iterazione precedente e si connettono tutti gli otto nodi inferiori delle due nuove unità alla radice.

Queste regole possono facilmente essere generalizzate. Infatti, al passo n verrà eseguita la seguente operazione:

passo n: si aggiungono due unità di 3^{n-1} nodi ciascuna, identiche alla rete creata nell'iterazione precedente (passo $n-1$), e si connette ognuno dei 2^n nodi inferiori di queste unità alla radice della rete.

Concentrandosi maggiormente sulla distribuzione del grado dei nodi, si può determinare con esattezza il valore di γ di tale algoritmo. Al passo i il grado dell'*hub* più connesso, la radice, è $2^{i+1}-2$. Nella successiva iterazione due copie di questo *hub* appaiono nelle due nuove unità aggiunte. Iterando ulteriormente, all'n-esimo passo saranno presenti nella rete 3^{n-i} copie di questo *hub*. Tuttavia, le due copie appena create non aumenteranno il loro grado con ulteriori iterazioni. Perciò, dopo n iterazioni vi saranno $(2/3)3^{n-i}$ nodi con grado $2^{i+1}-2$. Da ciò otteniamo che l'andamento della distribuzione del grado, determinata dagli *hub*, segue $P(k) \backsim k^{-\frac{\ln 3}{\ln 2}}$. Così l'esponente del grado è $\gamma = \frac{\ln 3}{\ln 2}$. L'algoritmo proposto genera, quindi, una rete con $\gamma = \frac{\ln 3}{\ln 2}$ fisso; tuttavia, si può modificare facilmente il modello per cambiare l'esponente di scala variando il numero di collegamenti connessi alla radice ad ogni passo. [9]

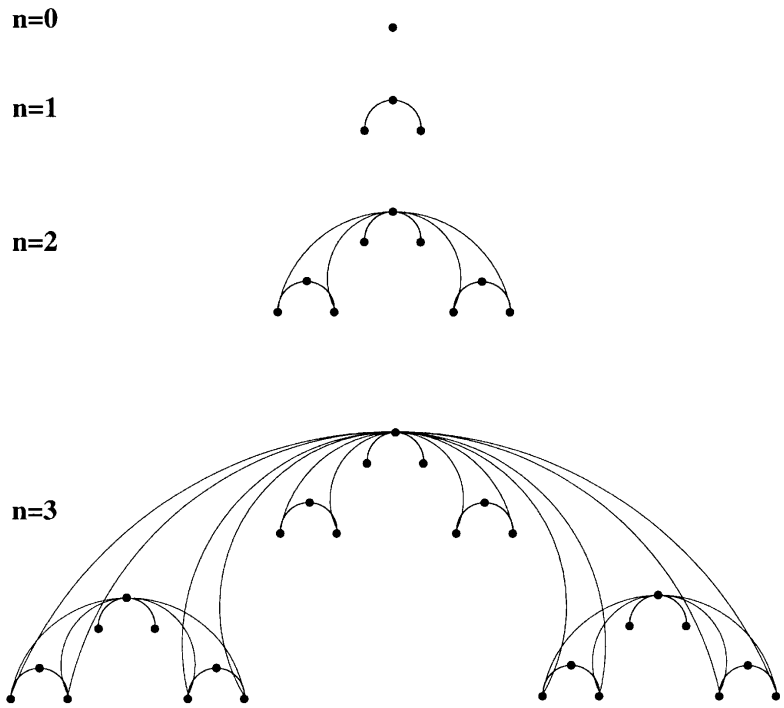

Figura 1.1: Costruzione della rete *scale-free* deterministica. Vengono mostrati i primi quattro passi del procedimento iterativo (tratto da [9])

1.4 Uncorrelated Configuration Model (UCM)

Un modello che si discosta dalla formulazione presentata da Barabási ed Albert, ma che risulta interessante, è quello per la generazione di reti *scale-free* casuali senza correlazioni, presentato in [10]. È stato verificato, infatti, che le reti reali sono caratterizzate, al di là della distribuzione del grado dei nodi, dalla presenza di correlazioni fra i gradi stessi; si può verificare, infatti, che nodi altamente connessi siano preferenzialmente connessi ad altri nodi altamente connessi e viceversa, oppure che nodi altamente connessi siano più probabilmente connessi ad altri con poche connessioni. Per reti non correlate, i gradi nei punti terminali di ogni arco sono completamente indipendenti. Ciononostante, tali reti sono ugualmente importanti da un punto di vista pratico. [10]

Il modello classico di generazione di reti casuali con distribuzione del grado dei nodi arbitraria presenta una problematica nel caso di reti con distribuzione *scale-free*, ossia l'elevata presenza di autoanelli ed archi multipli, non desiderati nell'ambito della simulazione, sia perché la maggior parte delle reti reali non presenta tali strutture, sia per evitare ambiguità nella definizione della rete e di tutte le sue dinamiche. [10]
L'algoritmo UCM, viceversa, genera una rete *scale-free* casuale senza correlazione, nel modo seguente:

1. si assegna ad ogni nodo i, in un insieme di N nodi inizialmente non connessi, un numero k_i di *stub* (parti terminali di arco emergenti dal nodo), dove k_i è tratto dalla distribuzione probabilistica $P(k) \backsim k^{-\gamma}$ e soggetto ai vincoli $m \leq k_i \leq N^{1/2}$ e $\sum_i k_i$ pari, con m grado minimo. Come ottimizzazione numerica di questo algoritmo, si pone m=2, in modo da generare reti connesse con probabilità 1;

2. si costruisce la rete scegliendo gli *stub* casualmente e connettendoli formando archi, rispettando i gradi preassegnati ed evitando autoanelli ed archi multipli. [10]

Capitolo 2

Implementazione dell'algoritmo

In questo capitolo viene presa in esame l'implementazione dell'algoritmo BA, con particolare attenzione agli aspetti computazionali. Esso è basato su due cardini fondamentali: *crescita* ed *annessione preferenziale*, entrambi elementi caratteristici delle reti reali; esse, infatti, si arricchiscono continuamente di nuovi nodi (crescita), che si connettono agli altri nodi del sistema non sempre in maniera casuale ma, anzi, spesso a nodi già ricchi di collegamenti (annessione preferenziale). Ne sono esempi, come visto, la rete Internet (sia dal punto di vista delle pagine web che da quello dei collegamenti fisici), la rete degli attori di Hollywood, le pubblicazioni scientifiche, le interazioni proteiche ecc. Anche l'implementazione dell'algoritmo effettuata presenta entrambi gli elementi del modello BA, crescita ed annessione preferenziale. Si è scelto di partire da un grafo non orientato di dimensione arbitraria N (minimo due nodi) completamente connesso. Ad ogni *time step* si aggiunge al grafo un nodo k privo di connessioni; si sceglie, quindi, seguendo alla lettera l'algoritmo BA, uno dei nodi del grafo già esistenti con probabilità proporzionale al suo grado (il funzionamento verrà mostrato in seguito nel dettaglio), generando un arco fra tale nodo ed il nodo k. Il procedimento viene iterato per il numero desiderato v di archi da generare a partire da k; tale valore viene stabilito prima dell'avvio della simulazione, con unico vincolo $v \leq N$. L'algoritmo prosegue fino all'annessione del numero desiderato

di nodi. Come si può facilmente notare, entrambi i cardini del BA vengono rispettati: come indicato in letteratura, non si parte da un grafo vuoto, bensì da un numero arbitrariamente piccolo di nodi; la scelta di un grafo iniziale completamente connesso è dettata dalla necessità di rendere uguale il "peso" di ogni nodo del primo nucleo, in modo che risultino tutti eligibili all'annessione in maniera equiprobabile. La rete, quindi, cresce in maniera discreta con l'annessione seriale di nodi (primo cardine), mantenendo sempre la correlazione fra grado dei nodi ed annessione dei nuovi a quelli preesistenti (secondo cardine). Più nel dettaglio, per poter implementare questa seconda meccanica, si è utilizzato il seguente algoritmo:

1. si genera il grafo di partenza con N nodi completamente connesso; pertanto, ogni nodo avrà esattamente $N-1$ connessioni, quindi grado $N-1$. Si costruisce virtualmente un segmento orientato di lunghezza $N(N-1)$, poiché costituito da N segmenti di modulo pari al grado dei nodi;

2. si stabilisce, quindi, un ordinamento lineare dei nodi stessi (ad esempio basandosi sul loro ID progressivo) e li si colloca, ordinatamente, lungo il segmento orientato, assegnando ad ognuno un numero intero rappresentante l'estremo destro della propria posizione. Il modulo del segmento relativo ad un nodo k è trivialmente ricavabile effettuando la differenza fra l'estremo destro di k e l'estremo destro del nodo $k-1$, antecedente a k nell'ordinamento stabilito;

3. ad ogni iterazione i, si aggiunge al grafo un nuovo nodo non connesso;

4. si estrae un numero casuale compreso fra zero e l'estremo destro del nodo $i-1$ e, mediante ricerca dicotomica, si rintraccia a quale segmento esso faccia riferimento, ottenendo conseguentemente il candidato r all'annessione con il nuovo nodo i. Si verifica, inoltre, che non sia già presente un arco fra r ed i, nel caso in cui si stia generando più di un arco da i; la totale assenza di autoanelli è garantita, invece, dal

meccanismo stesso di costruzione del grafo, in quanto ogni nodo aggiunto può essere collegato solo ai precedenti già esistenti. Al termine di tale verifica, se ottenuto esito positivo, si aggiorna il grado del nodo r (espandendo di una unità il relativo segmento) e si adegua tutta la struttura dati a tale aggiornamento, traslando di una unità tutti i nodi compresi fra $r+1$ e $i-1$; in caso contrario si procede ad una nuova estrazione e si ripete il passo;

5. si itera il procedimento descritto al punto precedente fino alla creazione dei v archi uscenti dal nodo i, stabiliti in partenza;

6. si aggiorna la struttura dati inserendo il nodo i con la seguente relazione: $d_i = d_{i-1} + v$, con d_i estremo destro del nuovo nodo i e d_{i-1} estremo destro del nodo $i-1$;

7. si torna al punto 3 fino al riempimento del grafo con la quantità desiderata di nodi.

La figura 2.1 riassume l'algoritmo appena enunciato.
È stato verificato sperimentalmente (con risultati mostrati successivamente) che tale algoritmo porta alla generazione di un grafo *scale-free* di dimensione arbitraria.

2.1 Analisi computazionale

Dal punto di vista computazionale, l'algoritmo sopra presentato presenta come principale criticità l'aggiornamento della struttura dati del segmento orientato del grado dei nodi. Se, infatti, si possono stimare con costo $O(1)$ sia l'aggiunta del nodo al grafo, sia l'estrazione del numero casuale, altrettanto non può essere fatto con l'aggiornamento della struttura. Ad ogni passo, infatti, se nel grafo sono presenti N nodi, nel caso pessimo il nuovo nodo $N+1$ verrà annesso al nodo con ID 0 (prima iterazione) ed ai nodi immediatamente seguenti (iterazioni successive). Considerando, quindi, ad ogni nuovo nodo la generazione di v archi, si avrà $C_{BA} \backsim v \cdot O(N)$.

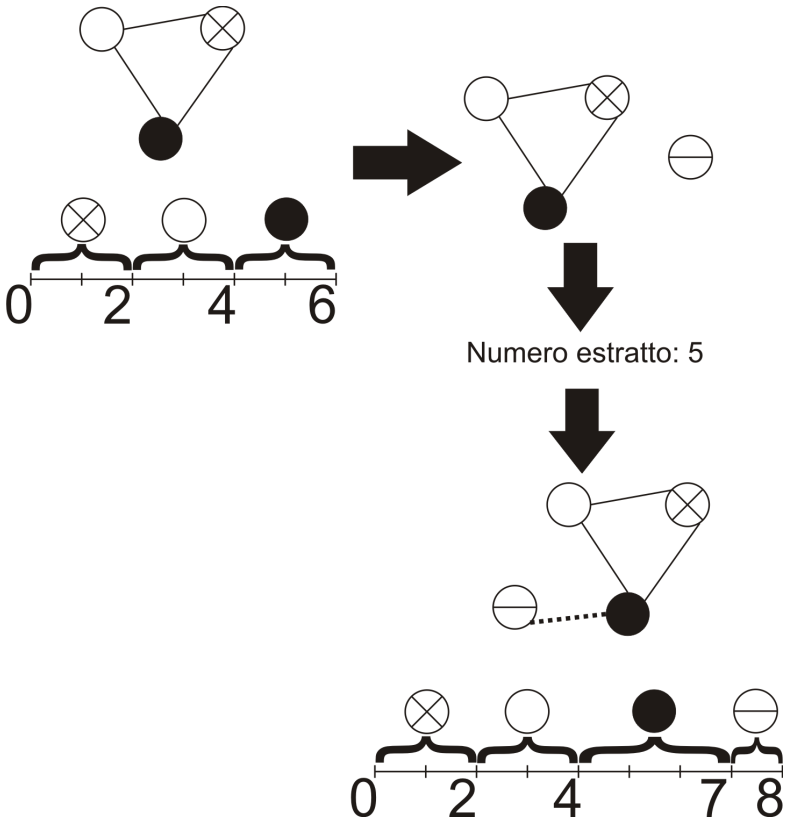

Figura 2.1: Primi passi dell'algoritmo di costruzione della rete *scale-free*, con nucleo iniziale di tre nodi. Dopo aver aggiunto al grafo il nuovo nodo, si estrae un numero casuale nell'intervallo $[0, 6]$; successivamente, si procede all'annessione del nuovo nodo ed all'aggiornamento della struttura dati

2.2 Linguaggio e librerie

Per l'implementazione dell'algoritmo è stato scelto di utilizzare il linguaggio standard ANSI C, principalmente per motivi prestazionali, in ambiente Linux, compilato con GCC[1] versione 4.4.5. Esso è stato integrato con l'utilizzo di librerie di supporto alla realizzazione delle strutture dati necessarie: dopo numerose ricerche, per la realizzazione della struttura dati astratta grafo è stata scelta la libreria igraph [IGRAPH], rilasciata sotto licenza GNU-

[1]GNU Compiler Collection, precedentemente noto con il nome di GNU C Compiler.

GPL[2]. Essa incorpora tutti gli elementi fondamentali richiesti: creazione di un grafo vuoto e relativa inizializzazione, aggiunta di nodi ad un grafo dato e creazione di un arco fra due vertici. In essa sono forniti, inoltre, alcuni metodi utili per generare rapidamente grafi con particolari caratteristiche topologiche, ad esempio circolari o completamente connessi, funzionali alla generazione del nucleo iniziale da cui avviare l'algoritmo. Il segmento orientato descritto nella precedente sottosezione, invece, è stato realizzato con la seguente struttura dati:

```
struct nodo
{
    unsigned int grado;
    unsigned int posizione;
};
```

come si può notare, ogni nodo è identificato da una coppia di elementi: il *grado*, che rappresenta il modulo del segmento ad esso relativo e la *posizione* del nodo all'interno del segmento orientato. Il grafo, quindi, dal punto di vista dei nodi, viene visto come un'array monodimensionale di *struct nodo*, cui si può accedere in maniera diretta, per maggiore rapidità d'esecuzione. La posizione dell'ultimo nodo dell'array, quindi, rappresenta l'estremo destro del segmento orientato e, più nello specifico, l'estremo superiore dell'intervallo di ricerca da utilizzare all'atto dell'annessione di un nuovo nodo al grafo. Per terminare la dissertazione riguardo l'implementazione dell'annessione preferenziale, occorre prendere in esame la metodologia di estrazione del valore da andare a ricercare dicotomicamente nel segmento orientato; a tale scopo è stata utilizzata la libreria Gnu Scientific Library [GSL]. Una volta inizializzato un *random number generator* con un seme prelevato mediante una semplice chiamata alla funzione *rand* standard, ad ogni iterazione si estrae un numero casuale con distribuzione uniforme nell'intervallo $[0, p_N]$, con p_N posizione dell'ultimo nodo connesso al grafo sul segmento orientato,

[2]GNU General Public License.

e da esso si ricava il secondo estremo dell'arco proveniente dal nodo appena inserito.

2.3 Strutture dati e dettagli implementativi

Più nel dettaglio, sono stati utilizzati alcuni costrutti e strutture dati forniti con le librerie di cui sopra, integrati da ADT[3] implementati con elementi base del linguaggio C.

2.3.1 Grafo

La struttura grafo è stata implementata, con l'ausilio di *igraph*, mediante il tipo `igraph_t`, per la cui inizializzazione viene utilizzato il metodo `igraph_full`, dal quale può essere determinata la dimensione del primo nucleo; con tale metodo, come deducibile dalla sua *signature*, si costruisce un grafo di dimensione arbitraria completamente connesso. Le operazioni necessarie all'esecuzione dell'algoritmo sono anch'esse integrate in *igraph*: inserimento di un nuovo nodo (`igraph_add_vertices`), inserimento di un arco (`igraph_add_edge`) e conteggio dei vicini di un dato nodo, per determinarne il grado. Quest'ultima operazione non viene effettuata in maniera diretta, ma richiede alcuni passi intermedi: occorre, in primo luogo, istanziare un *selettore* di nodi e, mediante il metodo `igraph_vs_adj`, esso selezionerà gli adiacenti al nodo fornito come argomento; successivamente, si invoca sul selettore il metodo `igraph_vs_size`, per determinare la quantità di vicini selezionati e salvare tale valore in una variabile. Sebbene questa operazione possa risultare onerosa, occorre precisare che viene effettuata solamente sul nucleo iniziale; nei passi successivi è sufficiente modificare le strutture dati ausiliarie.

[3] Abstract Data Type.

2.3.2 Gestione numeri casuali

Per la generazione dei numeri interi casuali per l'annessione dei nuovi nodi, effettuata utilizzando la GSL, è necessaria innanzitutto l'istanziazione di un *random number generator* (`gsl_rng`), cui occorre assegnare all'atto dell'allocazione, mediante il metodo `gsl_rng_alloc`, un algoritmo di generazione fra quelli presenti all'interno della libreria. Opzionalmente può essere fornito al generatore un *seed* differente da quello di default con il metodo `gls_rng_set`; nel caso specifico, è stata utilizzata la funzione standard *rand*. Per l'estrazione dei numeri casuali, invece, è stata scelta la distribuzione uniforme, implementata per numeri interi dal metodo `gsl_rng_uniform_int`.

2.3.3 Segmento orientato dei nodi

Come già accennato in precedenza, per la gestione del segmento orientato da cui scegliere i nodi dopo le estrazioni, è stata implementata una `struct` che contiene tutto ciò che è necessario per abbreviare i tempi di esecuzione, anche a dispetto del consumo di memoria: in essa, infatti, sono contenuti due dati, `grado` e `posizione`, parzialmente ridondanti, poiché dalla posizione si può ricavare il grado, sottraendo tale valore a quello del nodo precedente; tuttavia, l'esplicitazione di entrambi i dati diminuisce, seppur di poco, l'utilizzo di cicli di CPU. Per la gestione dell'intero grafo viene allocato dinamicamente (ossia mediante `malloc`) un *array* di `struct nodo`. Ogni nodo viene identificato con il proprio ID progressivo (assegnato ordinatamente da *igraph* all'aggiunta di ogni nodo), che trova corrispondenza nell'indice dell'array di *struct*. L'aggiornamento consiste nell'incremento del valore *grado* del nodo annesso e del valore *posizione* di tutti i successivi all'interno del vettore. L'annessione di un nuovo nodo implica una ricerca dicotomica all'interno dei valori di *posizione* dei nodi esistenti, già ordinati per costruzione, per rintracciare il candidato all'annessione.

2.3.4 Calcolo dei tempi di esecuzione

Per l'analisi prestazionale è stato necessario introdurre delle primitive di registrazione dei *time stamp*, di cui riportare i valori. Come punto di inizio della misurazione è stata scelta la `igraph_full`, ossia la prima istruzione di manipolazione vera e propria del grafo; come punto di fine, ovviamente, è stato scelto il termine del ciclo di aggiunta dei nodi al grafo. I risultati sono riportati in secondi.

2.3.5 Parallelismo

Data l'intrinseca serialità dell'algoritmo, è risultato adatto ad un approccio parallelo solamente l'aggiornamento della struttura dati del segmento orientato: a tale scopo è stata utilizzata la libreria standard OpenMP [OMP], pienamente supportata dalla versione di GCC indicata in precedenza. Per provare ad accelerare l'aggiornamento del segmento al termine di ogni aggiunta di arco è stata, quindi, inserita una direttiva `#pragma omp parallel` all'interno del ciclo. I risultati prestazionali verranno mostrati nel dettaglio nei capitoli successivi.

Capitolo 3

Visualizzazione di grafi

Un aspetto interessante è la possibilità di rappresentare graficamente i grafi risultanti dalle simulazioni. Formalmente non esiste uno standard per la rappresentazione visuale dei grafi, ma attualmente lo standard *de facto* è il linguaggio DOT (di cui si possono trovare riferimenti in [DOT]), con cui risulta abbastanza agevole la rappresentazione di reti statiche, data anche la sua semplicità di scrittura. All'interno della libreria *igraph* vi è il metodo `igraph_write_graph_dot` che consente, a partire da un `igraph_t`, di scrivere il corrispondente file DOT, di cui è utilizzata solo la sintassi di base, ossia creazione di nodi ed archi (orientati o no).

3.1 Dot ed altri strumenti command-line

A fronte del linguaggio vi sono una serie di strumenti di base, eseguibili da linea di comando, per l'interpretazione e l'effettiva rappresentazione sotto forma di immagine, da adattare a seconda della tipologia di grafo in oggetto: ne sono esempi *dot*, tool omonimo adatto alla rappresentazione di grafi orientati, *neato*, che si adatta meglio per grafi di piccole dimensioni (circa 100 nodi) e di cui si conoscono poche caratteristiche topologiche, *circo*, funzionale per rappresentare grafi ciclici multipli. Essi, però, non sono gli unici:

molti altri si trovano in [GVIZ]. Un esempio di file renderizzato con *dot* è presente in figura 3.1.

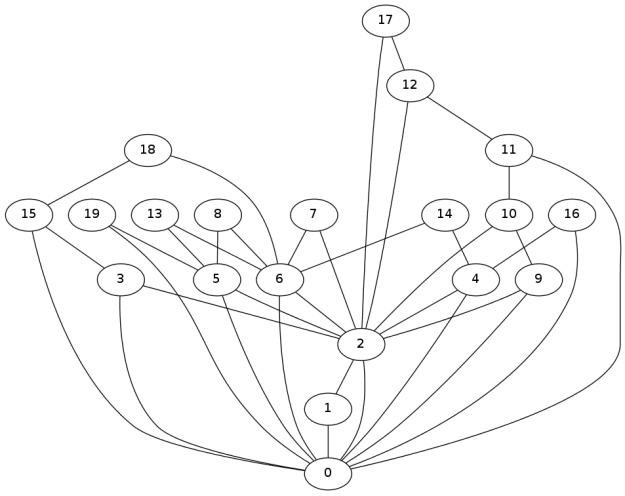

Figura 3.1: Esempio grafo renderizzato con *dot*

3.2 Gephi

Uno strumento dotato di interfaccia grafica per la visualizzazione e la modifica di grafi ottenuti a partire da un file DOT è Gephi [GEPHI], rilasciato sotto licenza GNU-GPL per più piattaforme (Windows, Linux, Macintosh). Una volta caricato il file DOT, è possibile modificare il *layout* del grafo da interfaccia, semplicemente effettuando operazioni di trascinamento col mouse. È possibile, inoltre, modificare l'aspetto grafico del grafo stesso modificando, ad esempio, la colorazione dei nodi, in maniera statica (assegnando un colore a ciascun nodo) o dinamica (ad esempio stabilendo un gradiente di colorazione dei nodi a seconda del loro grado). Fra le caratteristiche presenti, è di rilievo la compatibilità con il formato non-standard ed XML-based GEXF [GEXF], altra alternativa di rappresentazione dei grafi: a differenza di *dot*, la sua caratteristica più interessante è, anche in relazione alle caratteristiche dell'algoritmo BA, la possibilità di creare grafi dinamici, quindi mutevoli nel

tempo. Gephi, una volta caricato un file GEXF contenente un grafo dinamico, mostra una *timeline* mediante la quale è possibile scorrere la "vita" del grafo ed osservare la sua evoluzione (figura 3.2). Molte altre sono le funzioni che offre questo software, ma in questo contesto risultano non rilevanti; per una panoramica completa sulle possibilità di Gephi si rimanda a [GEPHI].

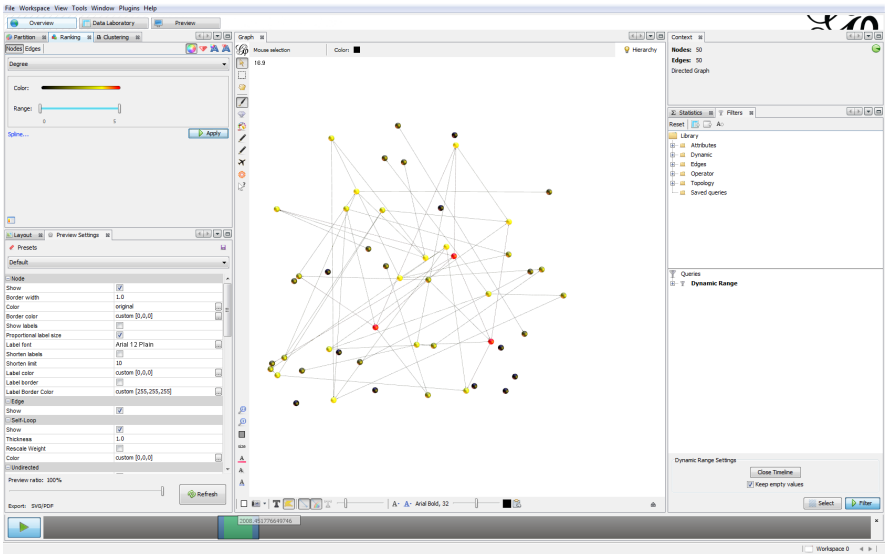

Figura 3.2: Interfaccia grafica di Gephi. In evidenza, nella parte bassa dello schermo, la *timeline* di scorrimento dei grafi dinamici

3.3 Ubigraph

Un altro strumento visivamente più "accattivante" e che offre notevoli possibilità sul fronte dell'interazione con altri software mediante API[1] dedicate, è Ubigraph [UBI]. Esso presenta una struttura client-server: il server, lanciato su di una qualsiasi macchina, rimane in ascolto su di una porta predefinita e da un qualsiasi software (anche da remoto, previo port-forwarding) è possibile invocare istruzioni che provochino il rendering tridimensionale di nodi ed archi sulla finestra del server; ovviamente, la coerenza e la consisten-

[1] Application Programming Interface.

za delle chiamate, sia dal punto di vista sintattico che per una visualizzazione corretta del grafo, è compito del programmatore. Come anche per gli strumenti citati in precedenza, è possibile personalizzare alcune caratteristiche del grafo, come forma e colore dei nodi, apposizione di etichette a nodi ed archi ed altre caratteristiche grafiche applicabili ad oggetti tridimensionali, come ad esempio il livello di dettaglio del rendering; seppur in versione alpha, per un utilizzo ordinario risulta stabile e abbastanza veloce, anche su macchine con ridotte capacità hardware. Per quanto riguarda i linguaggi di programmazione, l'interazione con il server è disponibile per Python, Ruby, Perl, Java, C, C++ ed altri; il motore di rendering utilizza OpenGL, mentre per la realizzazione delle chiamate al server sono state utilizzate le XML-RPC [XMLRPC]. Il software è disponibile per Macintosh e per alcune distribuzioni Linux (sia a 32 che a 64 bit); il server viene rilasciato come eseguibile già compilato, utilizzabile gratuitamente previa accettazione di un contratto di licenza, mentre il client viene distribuito sotto forma di sorgenti, con licenza Apache 2.0 [APACHE].

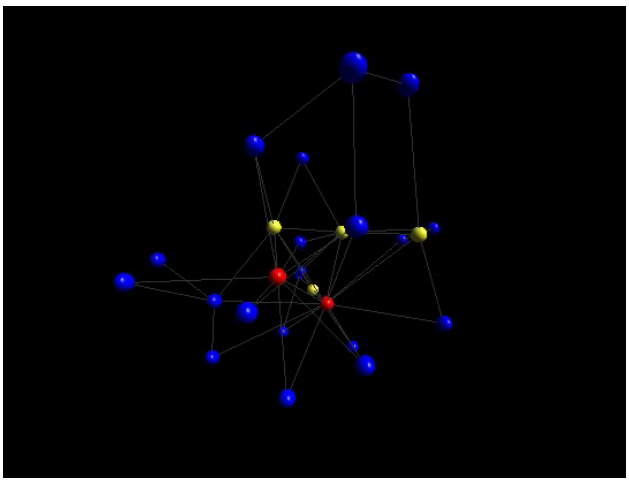

Figura 3.3: Esempio di grafo con Ubigraph. Le caratteristiche come colore o forma dei nodi possono essere modificate in qualsiasi momento, sia per nodi singoli che per tutto il grafo, mediante chiamate a metodi a disposizione

Capitolo 4

Valutazione sperimentale dell'implementazione

In questo capitolo vengono riportati i risultati dei test sull'algoritmo implementato, sia da un punto di vista qualitativo, ossia aderenza delle reti simulate alla topologia *scale-free*, sia quantitativo, effettuando misurazioni relative alle performance dell'algoritmo con differenti parametri iniziali. Per quest'ultimo tipo di analisi è stata presa in esame anche la variante parallelizzata dell'algoritmo.
I test sono stati effettuati su di una macchina del laboratorio del gruppo di ricerca sulla simulazione parallela e distribuita [PADS], presso il dipartimento di Scienze dell'Informazione a Bologna. Il calcolatore utilizzato è equipaggiato con due processori Intel™Xeon™da 2,80GHz e tecnologia Hyper-Threading™, 1GB RAM e sistema operativo Ubuntu 10.04.

4.1 Modalità di misurazione

Prima di presentare i risultati, è necessario introdurre brevemente le modalità con cui sono stati effettuati i test, cosicché i grafici successivi risultino di agevole lettura. Per i test prestazionali, si è deciso di effettuare esecuzioni successive dell'algoritmo, generando quindi istanze differenti, a partire da

1000 nodi, raddoppiando di volta in volta fino ad un massimo di 64000; per i test di valutazione topologica, la dimensione del grafo adottata è di 15000 nodi, quantità ritenuta sufficiente per poter mostrare le peculiarità del modello *scale-free*. Si è deciso, inoltre, di generare sempre 2 archi ad ogni annessione di nodo (come peraltro indicato in letteratura) facendo, però, variare la dimensione del nucleo iniziale di partenza secondo tre valori differenti: 3, 5 e 7 nodi; in tutti i casi, tali grafi sono stati generati *fully-connected*, in modo da rendere tutti i nodi eligibili per l'annessione del primo nuovo nodo in maniera equiprobabile.

4.2 Verifica topologica del grafo

Per verificare le principali caratteristiche topologiche di un grafo esistono numerosi metodi, presentati in [11]. Nel caso specifico, lo strumento che più risulta funzionale alla verifica della correttezza della topologia dei grafi generati dalla simulazione è il grafico relativo alla *degree distribution* dei nodi, ossia la distribuzione del grado[1] dei nodi. Intuitivamente, la principale peculiarità delle reti *scale-free* dal punto di vista della quantità di archi provenienti da un dato nodo è la presenza di una ristrettissima minoranza di nodi con grado elevatissimo (i cosiddetti *hub*), a fronte della quasi totalità di nodi con pochissime connessioni, peraltro principalmente verso gli *hub*. In effetti, in [3] viene mostrato come la *degree distribution*, rappresentata in scala logaritmica, assuma l'andamento di un'approssimazione per punti di una retta monotona decrescente, che si "apre" in prossimità dell'asse delle ascisse. Ciò indica, come appena enunciato, che vi sono una grande quantità di nodi con grado molto basso ed una ristrettissima minoranza di nodi con grado elevatissimo, gli *hub*.
Nelle figure 4.1, 4.2 e 4.3 sono rappresentati i grafici della distribuzione del grado dei nodi nel caso delle simulazioni effettuate sull'algoritmo implementa-

[1] Si ricorda che, in un grafo non orientato, il grado di un nodo è definito come il numero di archi in esso incidenti.

to: anche in questo caso si può osservare l'andamento tipico della *scale-free*, anche variando la dimensione del nucleo iniziale. Nella figura 4.4 è rappresentata, invece, la sovrapposizione dei 3 grafici precedenti, che mostra la sostanziale uguaglianza dei risultati ottenuti, anche con nuclei differenti, perlomeno per grafi delle dimensioni qui sperimentate. Non è possibile, perciò, affermare che in generale la quantità di nodi presente nel nucleo iniziale non influenzi in alcun modo lo svolgimento dell'algoritmo, in quanto sarebbero necessari ulteriori esperimenti con sostanziali variazioni sia nella dimensione del grafo iniziale, sia in quella complessiva del grafo risultante al termine della simulazione.

4.3 Valutazione prestazionale

Per quanto riguarda le misurazioni dei tempi di esecuzione dell'algoritmo, sia per la versione seriale che per quella parallela sono state effettuate cinque misurazioni consecutive per ogni dimensione del grafo risultante (da 1000 a 64000 nodi, raddoppiando ad ogni test) ed anche per ogni dimensione di grafo iniziale stabilita (3, 5, 7 nodi). Nella tabella 4.1 sono riportati media e varianza di tali tempi (espressi in secondi): come si può notare, salvo alcuni casi, la "distanza" dei valori misurati dalla media (per l'appunto la varianza) è sempre abbastanza ristretta mostrando, a dispetto della casualità delle istanze, poco scarto nei tempi d'esecuzione su grafi di piccole e medie dimensioni; tuttavia, specialmente nella variante parallelizzata, è da evidenziare una varianza in aumento all'aumentare della dimensione del nucleo iniziale per grafi di grandi dimensioni.
Nelle figure 4.5 e 4.6 vengono mostrate graficamente le medie riportate in tabella. È facile osservare che l'aumento di tempo necessario al completamento del grafo non è lineare all'aumentare dei nodi, ma di ordine superiore. Il rallentamento, peraltro, si evidenzia all'atto dell'aggiunta dei nodi a grafo già parzialmente formato, circa dai 20000 nodi in poi: se, infatti, fino a tale valore il rallentamento può approssimarsi alla linearità, oltre si ha un'esplo-

sione. Ciò è dovuto alla notevole quantità di aggiornamenti da effettuare sulle strutture dati allocate in memoria.
La variazione della dimensione del nucleo iniziale, come si può osservare, non porta variazioni medie sostanziali. Per poter osservare una differenza graficamente apprezzabile, occorre aumentare notevolmente la dimensione del grafo: in tal caso viene mostrato un miglioramento, comunque minimo, ottenuto con un grafo iniziale di dimensioni ridotte. Anche nel caso della variante parallela, si può osservare come la dimensione dei nuclei iniziali influisca in minima parte e solamente per grafi di grandi dimensioni invertendo, però, la tendenza rispetto all'algoritmo seriale: si ottengono, infatti, tempi di esecuzione minori con grafi iniziali di dimensione maggiore.

4.3.1 Seriale e parallelo: confronto

Come evidenziano i grafici di confronto in figura 4.7, l'*overhead* introdotto dalla gestione dei *thread* è superiore al vantaggio che la parallelizzazione può comportare; tuttavia, occorre ricordare che sarebbe necessario effettuare test su grafi di dimensioni ancora maggiori prima di poter scartare l'alternativa parallela in senso assoluto. L'aspetto interessante emerso dalle simulazioni compiute (e mostrato dai grafici) è la diminuzione dello scarto fra algoritmo seriale e parallelo all'aumentare della dimensione del nucleo iniziale osservabile, come in tutti i casi precedenti, solo per grafi molto grandi: nel caso di 64000 nodi, infatti, la distanza fra l'algoritmo seriale e quello parallelo con nucleo iniziale di 3 nodi è maggiore del corrispondente grafo con nucleo iniziale di 7. L'aggiornamento della struttura dati, evidentemente, risulta più oneroso, vista la dimensione maggiore del nucleo iniziale e, quindi, la maggior probabilità che gli *hub* appartengano ad esso. Se ciò accade, infatti, si ottiene un maggior costo computazionale nell'aggiornamento della struttura nel caso di numero elevato di nodi, rendendo conveniente la parallelizzazione. Si ricorda, altresì, che lo stesso algoritmo Barabási-Albert, per come è formulato, non lascia molte possibilità di implementazioni parallele; basti pensare al modello stesso da cui deriva: la *crescita* coinvolge l'annessione di un solo

nodo alla volta all'interno del sistema e l'*annessione preferenziale* è intrinsecamente seriale, poiché da ogni nuovo nodo viene generato un arco alla volta, indipendentemente da quanti ve ne debbano essere al termine del processo. Perciò, l'unica vera possibilità di miglioramento risiede principalmente nell'ottimizzazione prestazionale a livello del codice stesso (strutture dati, scelta del compilatore ecc).

		Seriale		Parallelo	
	Nodi	E_t	σ_t^2	E_t	σ_t^2
3	1000	$0,495$	$0,003 \cdot 10^{-3}$	$0,498$	$0,003 \cdot 10^{-3}$
	2000	$2,024$	$0,111 \cdot 10^{-3}$	$2,431$	$1,914 \cdot 10^{-2}$
	4000	$9,127$	$5,998 \cdot 10^{-3}$	$10,601$	$4,122 \cdot 10^{-2}$
	8000	$42,051$	$4,497 \cdot 10^{-2}$	$46,710$	$8,527 \cdot 10^{-1}$
	16000	$195,649$	$1,100$	$210,461$	$3,173 \cdot 10^{-1}$
	32000	$1059,202$	$25,487$	$1145,654$	$5,787 \cdot 10^{-1}$
	64000	$5865,090$	$694,105$	$6343,525$	$165,069$
5	1000	$0,497$	$0,005 \cdot 10^{-3}$	$0,500$	$0,004 \cdot 10^{-3}$
	2000	$2,025$	$0,035 \cdot 10^{-3}$	$2,383$	$6,041 \cdot 10^{-3}$
	4000	$9,050$	$4,944 \cdot 10^{-3}$	$10,530$	$2,861 \cdot 10^{-2}$
	8000	$41,493$	$1,362 \cdot 10^{-2}$	$46,678$	$3,659 \cdot 10^{-2}$
	16000	$195,844$	$9,123 \cdot 10^{-1}$	$218,334$	$11,186$
	32000	$1067,548$	$22,284$	$1142,862$	$28,711$
	64000	$5890,515$	$408,382$	$6332,448$	$467,252$
7	1000	$0,503$	$0,006 \cdot 10^{-3}$	$0,503$	$0,008 \cdot 10^{-3}$
	2000	$2,024$	$0,230 \cdot 10^{-3}$	$2,393$	$1,388 \cdot 10^{-2}$
	4000	$9,052$	$0,374 \cdot 10^{-3}$	$10,686$	$1,014 \cdot 10^{-1}$
	8000	$41,439$	$2,023 \cdot 10^{-2}$	$46,949$	$2,218 \cdot 10^{-2}$
	16000	$197,401$	$6,959 \cdot 10^{-1}$	$213,097$	$1,336$
	32000	$1072,211$	$83,358$	$1138,998$	$29,905$
	64000	$5869,134$	$1080,428$	$6306,220$	$512,161$

Tabella 4.1: Media E_t e varianza σ_t^2 dei tempi di esecuzione (in secondi) dell'algoritmo seriale e della variante parallela, con grafo iniziale di dimensione 3, 5, 7 nodi

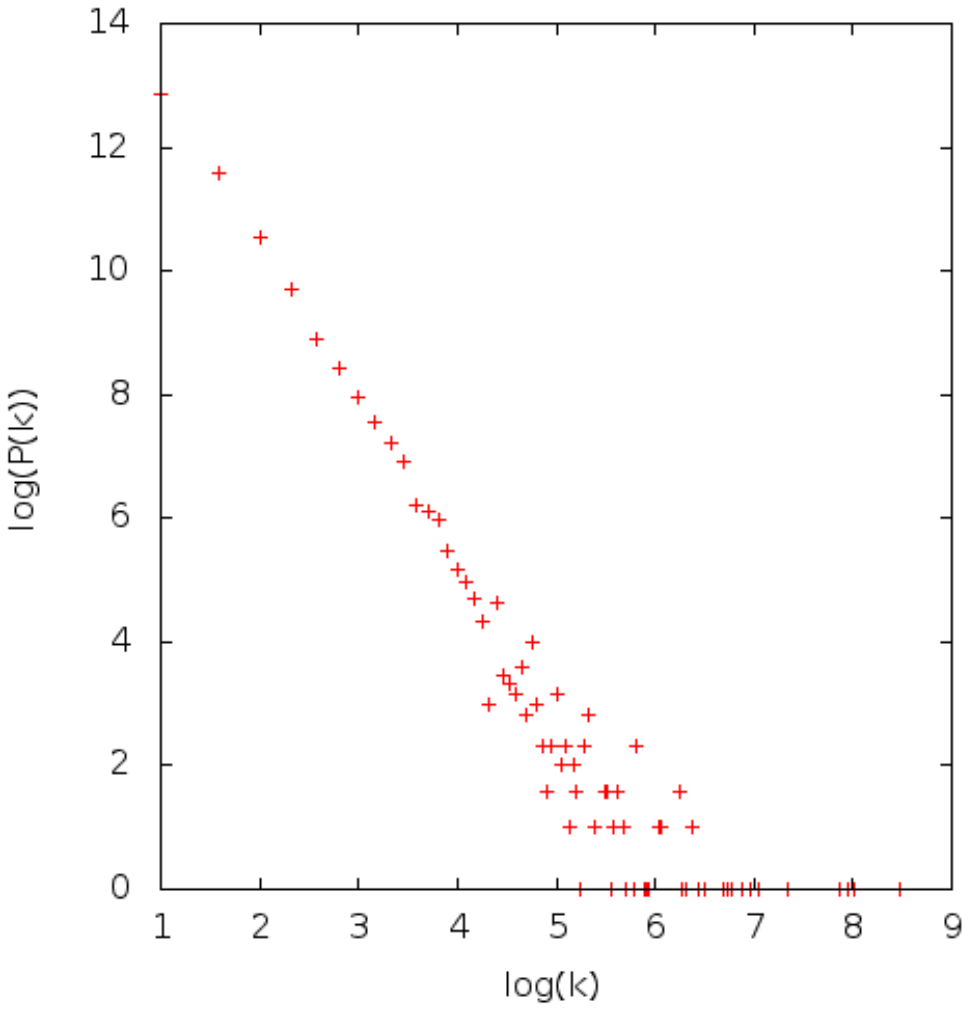

Figura 4.1: Grafico della distribuzione del grado dei nodi nel caso di nucleo di dimensione 3

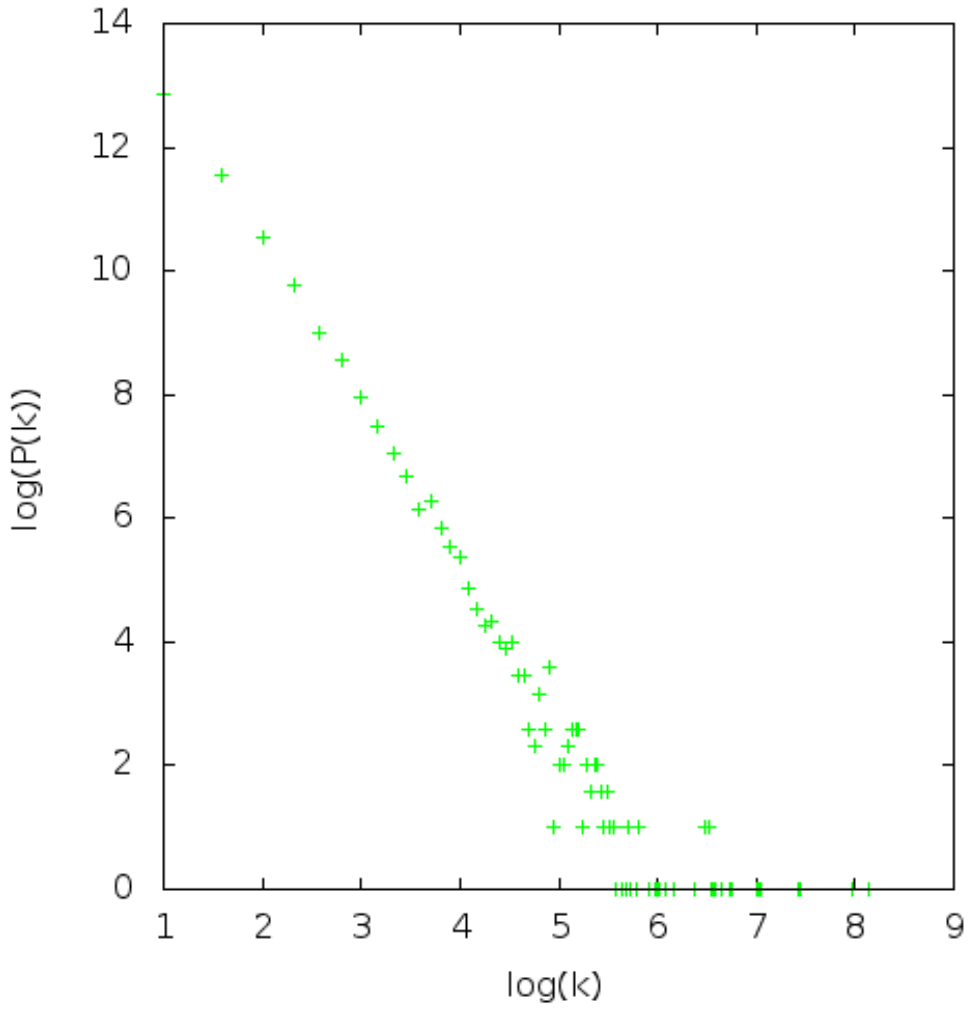

Figura 4.2: Grafico della distribuzione del grado dei nodi nel caso di nucleo di dimensione 5

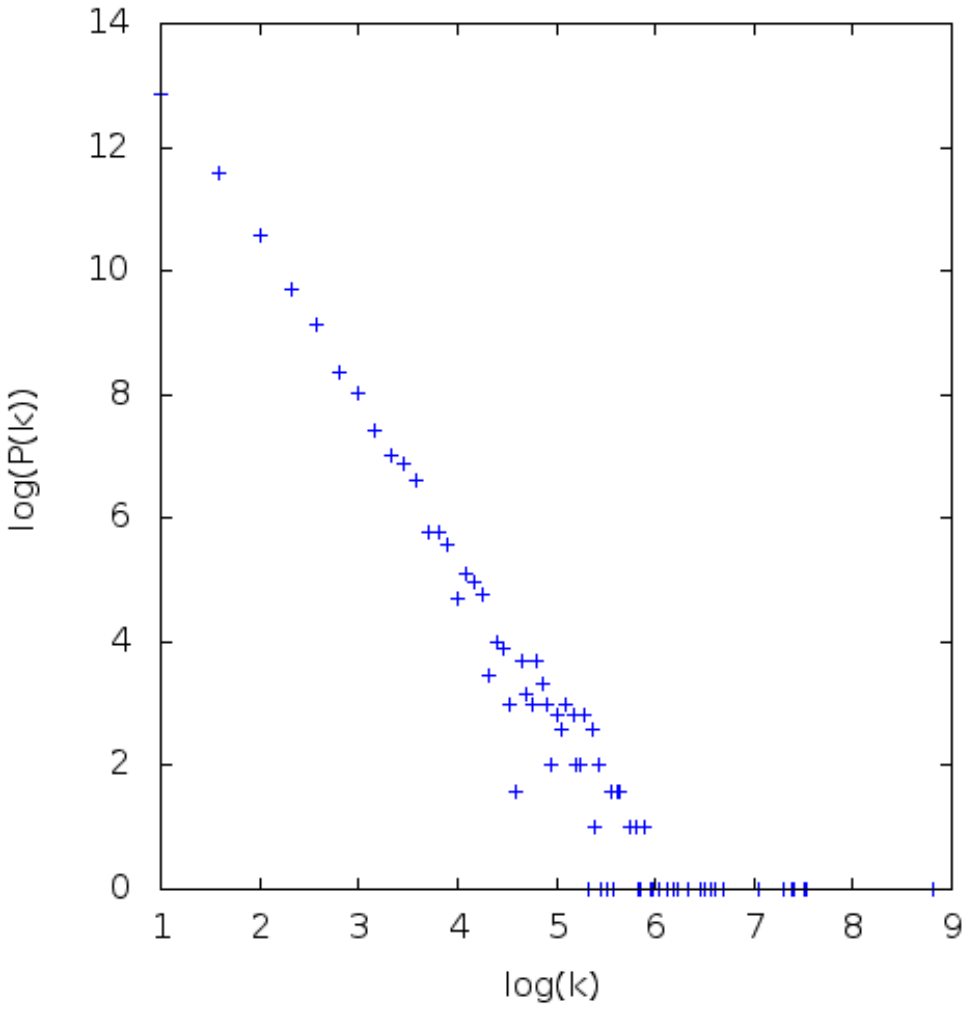

Figura 4.3: Grafico della distribuzione del grado dei nodi nel caso di nucleo di dimensione 7

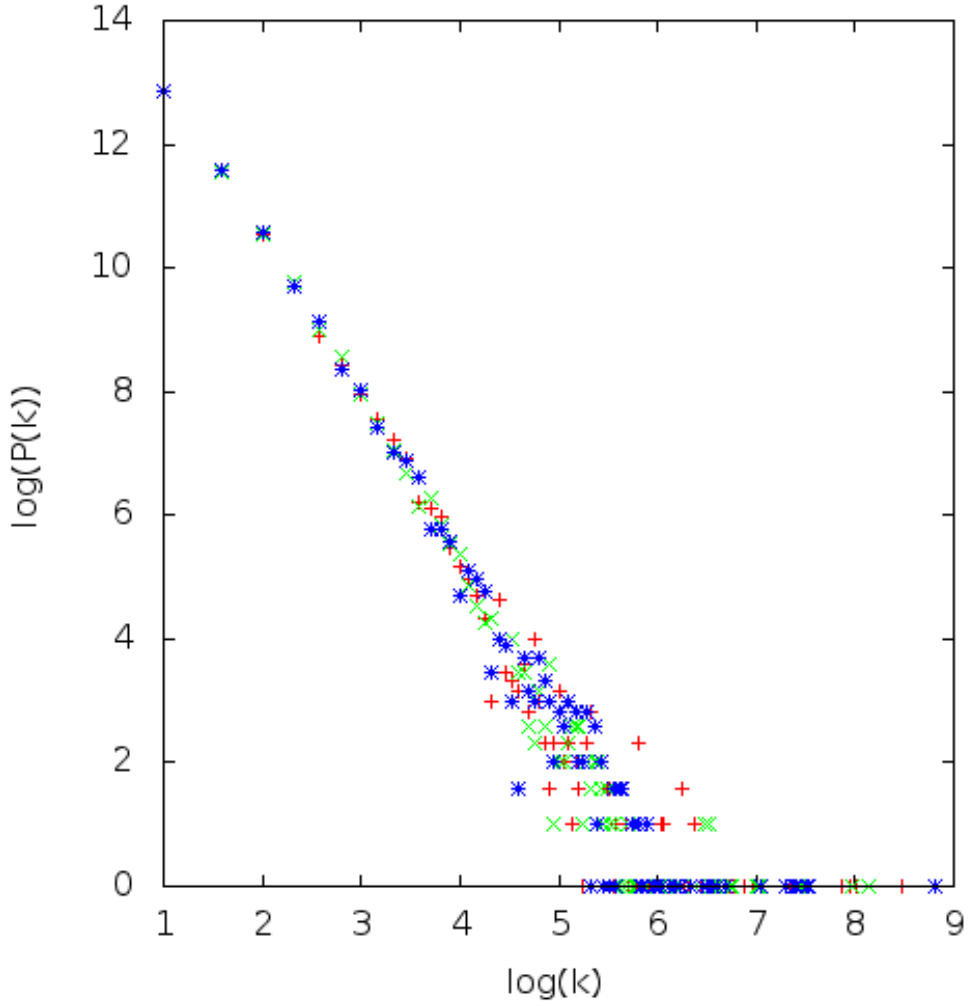

Figura 4.4: Sovrapposizione dei grafici in figure 5.1, 5.2, 5.3. I dati sperimentali ottenuti non evidenziano particolari differenze nelle diverse dimensioni del nucleo iniziale

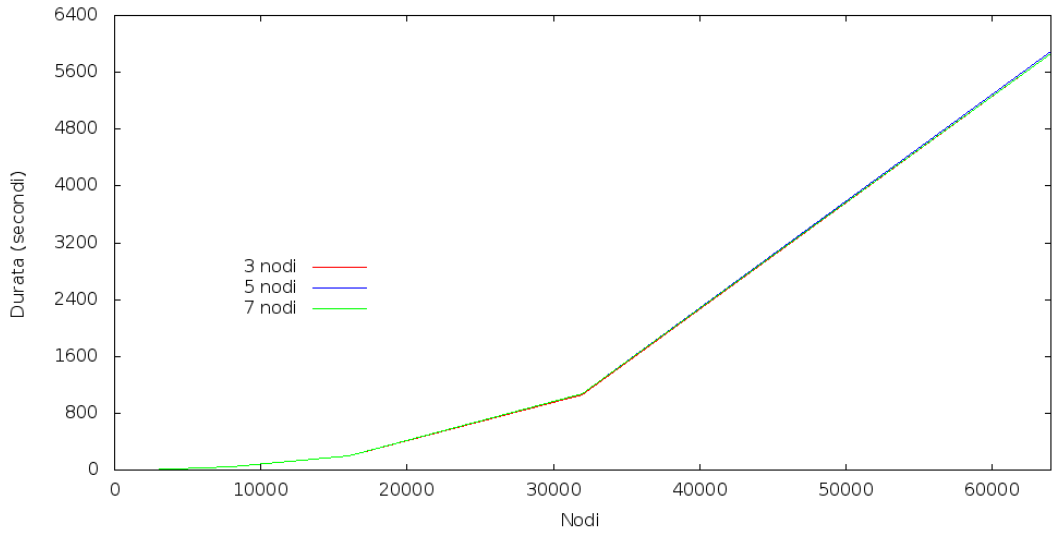

Figura 4.5: Tempi di esecuzione dell'implementazione seriale dell'algoritmo. Il tempo di esecuzione non cresce linearmente con il numero dei nodi

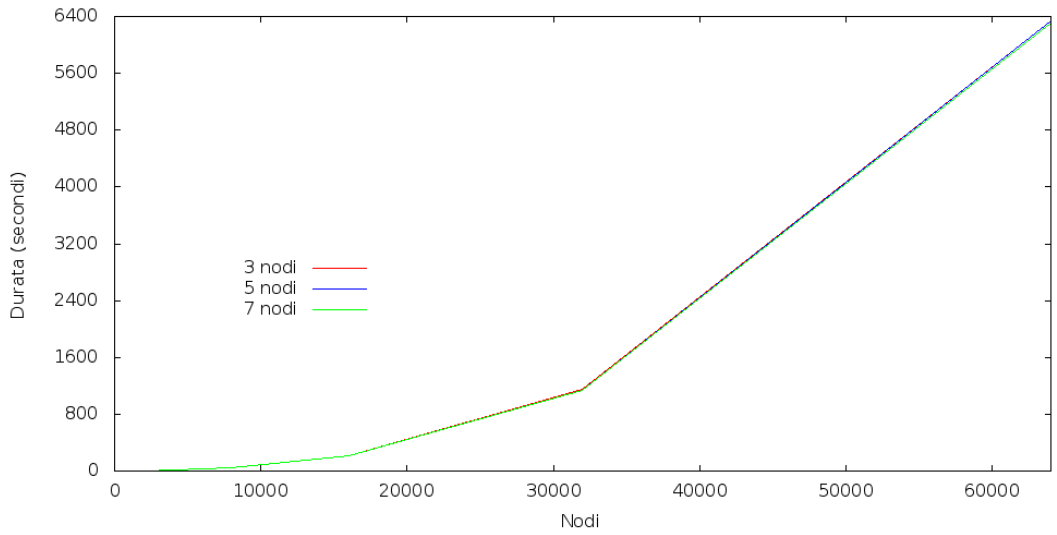

Figura 4.6: Tempi di esecuzione dell'implementazione parallela dell'algoritmo. Il tempo di esecuzione, anche in questo caso, non cresce linearmente con il numero dei nodi

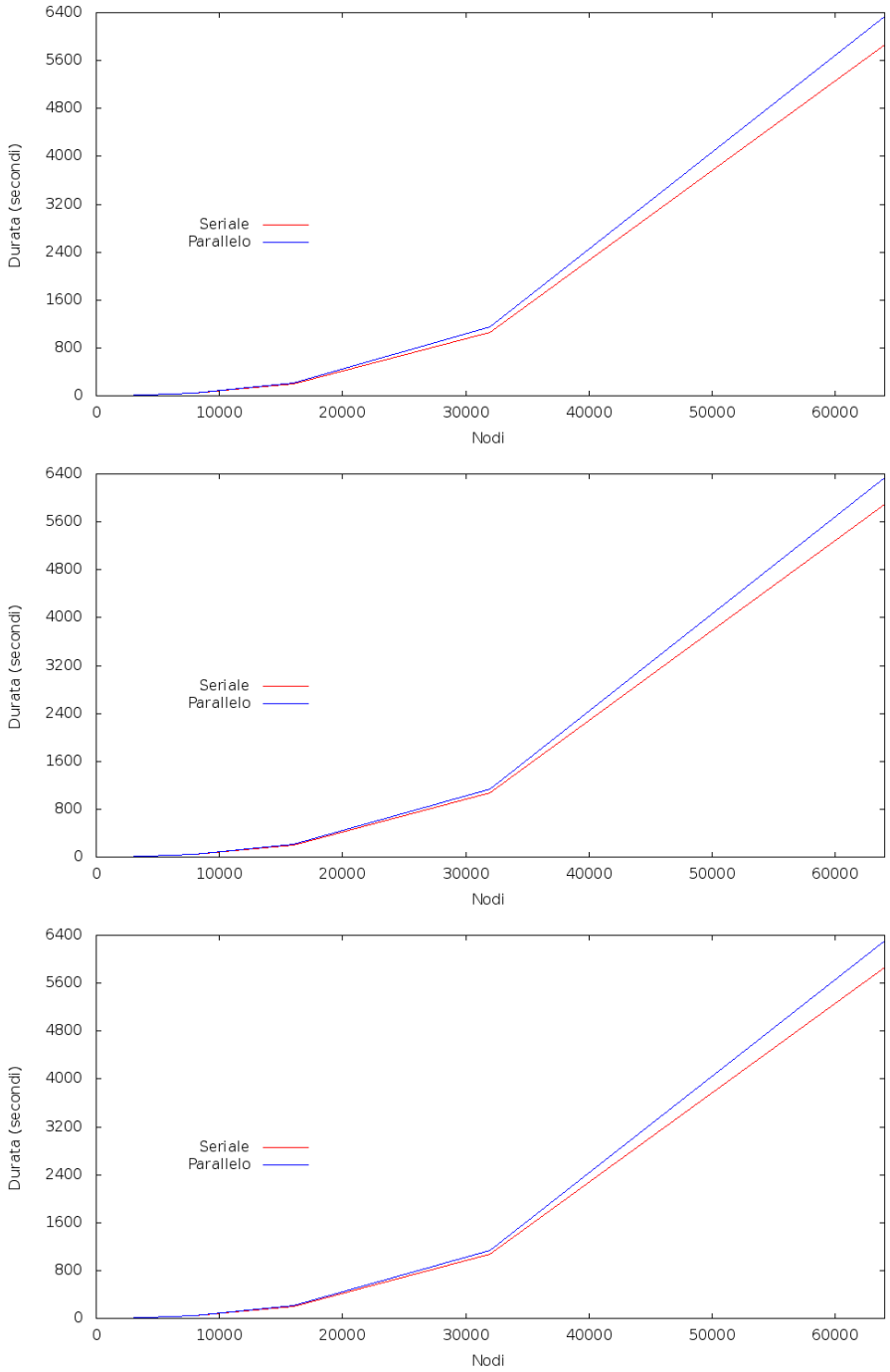

Figura 4.7: Confronto fra i tempi di esecuzione dell'algoritmo seriale e parallelo con nucleo di dimensione 3, 5, 7. Il colore rosso rappresenta l'algoritmo seriale, il blu quello parallelo

Conclusioni

Il concetto di rete viene spesso oggi associato solamente all'informatica. Tuttavia, esso è presente in un'impensabile quantità di ambiti differenti: dalla biologia alla letteratura, dalle scienze sociali all'intrattenimento; vista la loro vastità ed eterogeneità è naturale chiedersi se tutte le interconnessioni dinamiche siano puramente casuali o abbiano alle loro spalle un *fil rouge* comune. A questa domanda hanno tentato di dare risposta, nei tardi anni '50, due scienziati: Paul Erdös ed Alfréd Rényi, con un modello tanto solido nella teoria, quanto fragile per l'impossibilità di verifiche sperimentali. Esso prevede, intuitivamente, che la maggior parte delle reti possa essere modellata come una sorta di agglomerato di bottoni sparpagliati su un pavimento, connessi fra loro casualmente da un filo.

Il progresso tecnologico, tuttavia, ha svelato che, per una notevole quantità di reti reali, la mappa delle connessioni fra i vari nodi è tutt'altro che casuale. Siamo alla fine degli anni '90, quando Albert-László Barabási e Réka Albert, assieme ad altri scienziati, mettono a punto un software che si prefigge lo scopo di mappare una porzione del World Wide Web, attendendo un risultato aderente al modello Erdös-Rényi. I dati raccolti, però, raccontano un'altra storia: si osserva che vi è una ristrettissima minoranza di nodi "superconnessi", mentre la schiacciante maggioranza ha pochissime connessioni; sostanzialmente, un pugno di nodi sorregge l'intero sistema.

A ciò hanno fatto seguito ulteriori ricerche e studi, sia sul comportamento di epidemie all'interno di reti con questa topologia, sia sul modo in cui queste reti possono essere costruite: l'errore fondamentale dei modelli fino a quel

momento esistenti era la loro totale incapacità di modellare la *crescita* delle reti, unita al considerare tutti i nodi ugualmente eligibili per nuove connessioni, non considerando l'*evoluzione* delle reti stesse.

L'algoritmo Barabási-Albert, analizzato in questa dissertazione ed implementato, risponde alle peculiarità topologiche delle reti reali, modellandone l'aspetto in maniera fedele. Tuttavia, per come è formulato, non lascia molto spazio ad interpretazioni implementative, probabilmente anche per via della stringente logica costruttiva alle spalle; ne consegue che, salvo miglioramenti sul fronte della rappresentazione della probabilità di estrazione di un singolo nodo, non vi possano essere, a meno di modifiche concettuali, rilevanti evoluzioni dal punto di vista della velocità di computazione, sicuramente il neo maggiore dell'attuale implementazione. È altresì ovvio che questo documento non abbia alcuna presunzione di correttezza assoluta e che esistano numerose altre possibilità di miglioramento, sia dal punto di vista concettuale che computazionale.

Sicuramente, per quanto possa sembrare superfluo ad un occhio distratto, la conoscenza approfondita della topologia delle reti reali è di fondamentale importanza: con essa, infatti, è possibile non solo analizzare *a posteriori* reti già esistenti, ma generare strumenti che modellino *a priori*, ad esempio, il più probabile comportamento evolutivo di una rete di grandi dimensioni o la diffusione di un'epidemia (sia essa informatica o sociale), per prevenire od arginare il fenomeno su larga scala.

Bibliografia

[1] Albert-László Barabási, Eric Bonabeu. *Scale-free networks*. Scientific American - May 2003 pp.52-59

[2] Albert-László Barabási. *Scale-Free Networks: A Decade and Beyond*. Science 325, 412 (2009)

[3] Albert-László Barabási, Réka Albert. *Statistical mechanics of complex networks*. Reviews of modern physics, volume 74, January 2002

[4] Xiao Fan Wang, Guanrong Chen. *Complex Networks: Small-World, Scale-Free and Beyond*. IEEE Circuits and Systems Magazine. First quarter 2003

[5] Paul Erdös, Alfréd Rényi. *On the evolution of random graphs*.

[6] Albert-László Barabási, Réka Albert, Hawoong Jeong. *Mean-field theory for scale-free random networks*. Physica A 272 (1999) 173-187

[7] Albert-László Barabási, Réka Albert. *Emergence of Scaling in Random Networks*. Science, vol 286, 15 October 1999

[8] Alessandro Vespignani, Romualdo Pastor-Satorras. *Epidemic dynamics and endemic states in complex networks*. Physical Review E, Volume 63, 066117

[9] Albert-László Barabási, Erzsébet Ravasz, Tamás Vicsek. *Deterministic scale-free networks*. Physica A 299 (2001) 559-564

[10] Michele Catanzaro, Marián Boguñá, Romualdo Pastor-Satorras. *Generation of uncorrelated random scale-free networks*. Physical Review E 71, 027103 (2005)

[11] Fereydoun Hormozdiari, Petra Berenbrink, Nataša Pržulj, Cenk Sahinalp. *Not All Scale Free Networks Are Born Equal: The Role of the Seed Graph in PPI Network Emulation*. Lecture Notes in Computer Science, 2007, Volume 4532/2007, 1-13.

[WIKI] `http://en.wikipedia.org/wiki/Scale_invariance`

[IGRAPH] `http://igraph.sourceforge.net/`

[GSL] `http://www.gnu.org/s/gsl/`

[OMP] `http://openmp.org/wp/`

[DOT] `http://www.graphviz.org/content/dot-language`

[GVIZ] `http://www.graphviz.org/`

[GEPHI] `http://gephi.org/`

[GEXF] `http://gexf.net/format/`

[UBI] `http://ubietylab.net/ubigraph/index.html`

[XMLRPC] `http://www.xmlrpc.com/`

[APACHE] `http://www.apache.org/licenses/LICENSE-2.0`

[PADS] `http://pads.cs.unibo.it/dokuwiki/doku.php?id=`

NOTA: tutte le citazioni da testi in lingua straniera sono state effettuate traducendo o riassumendo nel modo semanticamente più fedele possibile.