SAPIENZA
UNIVERSITÀ DI ROMA

# Conteggio esatto ed approssimato di cicli da quattro in grafi non diretti

**Facoltà di Ingegneria dell'informazione, informatica e statistica**
**Corso di laurea in Informatica**
Cattedra di Algoritmi e strutture dati

**Candidato**
**Marco Mulas**
**1205518**

Responsabile
Prof. Angelo Monti

A/A 2012/2013

*A mia madre,*
*ad Angela,*
*a Giulia,*
*alla mia fidanzata e*
*a Te ovunque tu sia*

*Un ringraziamento sentito al mio relatore, il prof.Monti,*
*per la sua infinita pazienza e*
*per il tempo dedicatomi e*

*ai miei colleghi*
*Damiano, Domenico, Giulia, Marco F., Marco R., Marco S.*
*per lo studio che assieme abbiamo condiviso e*
*per avermi sopportato.*

*To iterate is human,*
*to recurse divine*
*(L.Peter Deutsch)*

*L'unico modo per essere davvero soddisfatti*
*sarà fare un gran bel lavoro*
*e l'unico modo per fare un gran bel lavoro*
*è amare quello che fate.*
*(Steve Jobs)*

*L'informatica non riguarda i computer*
*più di quanto l'astronomia riguardi i telescopi.*
*(Edsger W. Dijkstra)*

# Indice

**1 Introduzione** **1**

**2 Definizioni** **3**

**3 Lavori precedenti** **6**

**4 Algoritmi per il conteggio esatto dei quadrati** **8**

4.1 Ricerca esaustiva . . . . . 8

4.2 Conteggio con tecnica *backtracking* . . . . . 10

4.3 Conteggio mediante *2path* . . . . . 12

**5 Algoritmi approssimanti** **14**

5.1 Campionamento mediante quattro vertici . . . . . 15

5.2 Campionamento mediante un *2path* e un vertice . . . . . 17

5.3 Campionamento mediante un *3path* . . . . . 19

**6 Risultati sperimentali** **21**

6.1 Datasets . . . . . 22

6.1.1 Lavoro di preprocessing . . . . . 22

6.2 Risultati sperimentali degli algoritmi esatti . . . . . 24

6.2.1 Commenti . . . . . 26

6.3 Risultati sperimentali degli algoritmi approssimanti . . . . . 27

6.3.1 Commenti . . . . . 31

**7 Conclusioni e sviluppi futuri** **32**

# Capitolo 1

# Introduzione

In questa relazione di tirocinio verrà trattato il conteggio esatto ed approssimato di cicli da quattro in grafi non diretti e verranno descritti algoritmi più o meno efficienti per ciascuno scopo.

Per conteggiare approssimativamente i cicli da quattro in un grafo non diretto ci si servirà di algoritmi di conteggio denominati ($\delta$-$\epsilon$)-approssimanti basati su campionamento, vale a dire delle routine che saranno in grado di estrapolare dal grafo un particolare sottografo da quattro vertici e verificare se esso corrisponda ad un ciclo da quattro. La ($\delta$-$\epsilon$)-approssimazione, con una particolare scelta dei parametri $\delta$ e $\epsilon$, ci garantirà che con una probabilità maggiore o uguale a 1-$\delta$ la stima di conteggio prodotta sarà all'interno di un intervallo di raggio $\epsilon$ che avrà come centro l'esatto conteggio dei cicli da quattro, il quale verrà prodotto con gli algoritmi esatti sopracitati.

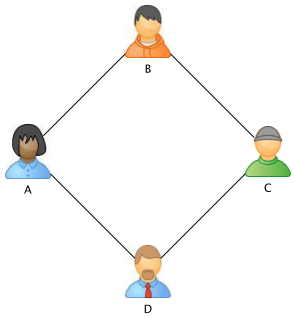
I grafi rappresentano un potente mezzo matematico per poter rappresentare la realtà, come, ad esempio, il *World Wide Web*, le reti stradali, le reti di collaborazioni e, non ultime, le reti sociali.

All'interno di una rete sociale un ciclo da quattro rappresenta un sottografo in cui una ipotetica informazione parte dall'utente $A$ percorrendo gli utenti $B$, $C$ e $D$, per poi tornare all'utente $A$ di partenza (Figura 1.1). In una rete ferroviaria, ad esempio, un ciclo da quattro rappresenta una porzione di rete in cui un treno locale parte da una stazione, percorre altre tre stazioni per poi raggiungere la stazione di partenza.

In questo caso verranno usati grafi che rappresentano sistemi autonomi, reti di co-

municazione email, reti di collaborazioni per articoli scientifici, ma anche grafi generati randomicamente.

Figura 1.1: Esempio di ciclo da 4 in un grafo non diretto




All'interno di questa relazione di tirocinio verranno date nel Capitolo 2 le definizioni necessarie al fine di migliorar la comprensione della relazione stessa, nel Capitolo 3 verranno enunciati e descritti i lavori precedenti e affini a questo tirocinio.

Successivamente verranno descritti degli algoritmi per il conteggio esatto di quadrati (Capitolo 4), il cui risultato servirà d'ausilio ad algoritmi usati per il conteggio approssimato di quadrati (Capitolo 5).

I risultati prodotti mediante gli algoritmi esatti ed approssimanti verranno riportati e commentati nel Capitolo 6.

Infine il Capitolo 7 verrà dedicato ai commenti conclusivi del tirocinio e ai lavori futuri.

# Capitolo 2

# Definizioni

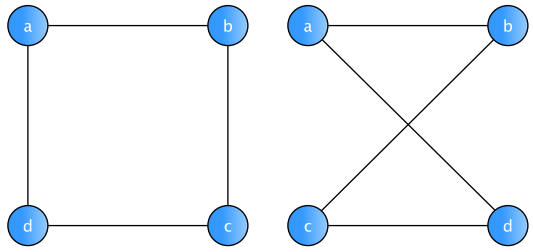
Per migliorare la comprensione degli algoritmi che andremmo ad analizzare vengono date le seguenti definizioni e inoltre viene scelto il grafo $G$=($\{a,\ b,\ c,\ d\}$,$\{\{a,b\}$, $\{a,d\}$ ,$\{b,c\}$, $\{b,d\}$, $\{c,d\}\}$ per esemplificare le definizioni al fine di ottenere una corretta comprensione:

> *Definizione 2.1:* Chiameremo *quadrato* (Figura 2.1) una qualsiasi sequenza $(a,b,c,d)$ del grafo $G$ se e solo se essa forma un ciclo di lunghezza quattro in $G$, ossia se e solo se $\{a,b\}$, $\{b,c\}$, $\{c,d\}$, $\{d,a\} \in E$.

Nel grafo $G$ di esempio la sequenza $(a,b,c,d)$ forma un *quadrato* poichè $\{a,b\}$, $\{b,c\}$, $\{c,d\}$, $\{d,a\} \in E$.

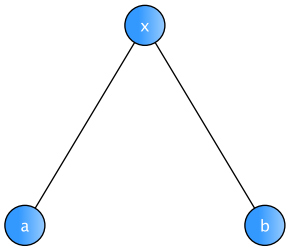
Si può facilmente notare che se $(a,b,c,d)$ è un quadrato allora le tre sequenze $(b,c,d,a)$, $(c,d,a,b)$, $(d,a,b,c)$, così come le quattro sequenze $(d,c,b,a)$, $(a,d,c,b)$, $(b,a,d,c)$, $(c,b,a,d)$ identificano lo stesso quadrato.

Figura 2.1: Esempi di *quadrato*



*Definizione 2.2:* Chiameremo *2path* (Figura 2.2) un qualsiasi cammino di lunghezza due all'interno del grafo. Diremo quindi che la terna $(a,x,b)$ è un *2path* di $G$ se e solo se $\{a,x\},\{x,b\} \in E$.

Figura 2.2: *2path*




Ad esempio uno dei *2path* di $G$ può essere

$$\{b,a,d\}$$

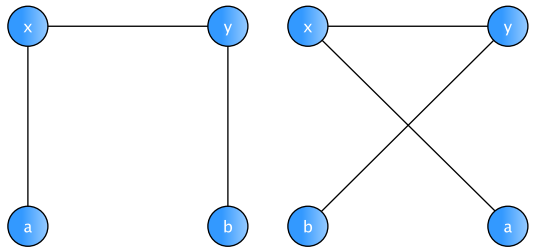
poichè $\{b,a\}$, $\{a,d\} \in G$.

Per come è stato definito è possibile rappresentare lo stesso *2path* $(a,x,b)$ anche nella sua lettura da destra a sinistra *(b,x,a)*. Poichè in questo tirocinio si consideranno solo grafi non diretti l'istanza di un *2path* che va da un nodo $a$ in un nodo $b$ implica l'istanza di un *2path* che va dal nodo $b$ al nodo $a$. Quindi se $G$ ha il *2path* $(a,x,b)$ allora avrà anche il *2path* $(b,x,a)$.

A volte verrà tralasciato della terna $(a,x,b)$ il nodo $x$, poichè all'interno di quel determinato contesto non è importante come informazione e quindi verrà rappresentato il *2path* con $\{a,b\}$, semplicemente indicandone gli estremi.

*Definizione 2.3:* Chiameremo *3path* (Figura 2.3) un qualsiasi cammino di lunghezza tre all'interno del grafo. Diremo quindi che la quadrupla $(a,x,y,b)$ è un *3path* di $G$ se e solo se $\{a,x\}$, $\{x,\mathrm{y}\}$, $\{y,b\} \in E$.

Figura 2.3: Esempi di *3path*




Ad esempio uno dei *3path* all'interno di $G$ può essere

$$\{b,a,d,\mathrm{c}\}$$

poichè $\{b,a\}$, $\{a,d\}$, $\{d,c\} \in G$.

Per come è definito lo stesso *3path* $(a,x,y,b)$ è possibile rappresentarlo anche come $(b,y,x,a)$ per lo stesso motivo già esplicitato per il *2path*.

Nella Tabella 2.1 seguente vengono descritte le notazioni usate più frequentemente all'interno di questo tirocinio con relativa descrizione:

Tabella 2.1: Notazioni

| | |
|---|---|
| $V$ | insieme dei vertici di un grafo $G$ |
| E | insieme degli archi di un grafo $G$ |
| $P$ | insieme dei *2path* di un grafo $G$ |
| $Q$ | insieme dei *quadrati* di un grafo $G$ |
| E[$\beta$] | valore atteso della variabile aleatoria $\beta$ |
| $d_v$ | grado del nodo $v$ in un grafo $G$ |
| $d$ | grado massimo di un grafo $G$ |
| *adj(a)* | lista di adiacenti del nodo $a \in V$ |

# Capitolo 3

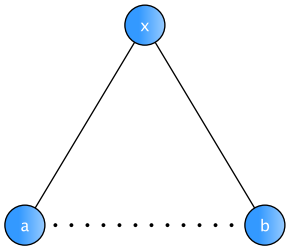
# Lavori precedenti

Il conteggio approssimato dei cicli da 4 è una generalizzazione del conteggio approssimato dei cicli da 3, chiamati per semplicità *clique di ordine 3* o *triangoli*.

Il problema del conteggio dei triangoli è stato ampiamente studiato e per questo problema sono stati proposti diversi algoritmi sia esatti che approssimanti.

Nell'articolo [13] viene proposto un algoritmo di approssimazione che prevede un campionamento che utilizza il *2path*: esso seleziona a caso un *2path* $(a,x,b)$ all'interno di $G$ e verifica l'esistenza dell'arco $\{a,b\}$ in $E$ al fine di completare il ciclo. Nel caso in cui esista verrà restituito il valore 1, altrimenti verrà restituito 0 (Figura 3.1).

Figura 3.1: Campionamento per il conteggio di triangoli mediante *2path*




Questo tipo di algoritmo, basato su campionamento, fa parte della famiglia degli algoritmi ($\delta$-$\epsilon$)-approssimanti che verrà descritta nel Capitolo 5.

Di seguito viene riportato lo pseudocodice:

**Algoritmo 3.1** Campionamento per il conteggio di triangoli mediante *2path*

INPUT: G=(V,E)

$\beta \leftarrow 0$
Scegli a caso un *2path* $(a,x,b)$
if $\{a,b\} \in E$ then
  $\beta \leftarrow 1$

OUTPUT: $\beta$

L'algoritmo di campionamento per il conteggio approssimato di *quadrati* mediante *2path* e un vertice, che viene descritto nella Sezione 5.2, è frutto di una generalizzazione dell'algoritmo di campionamento per il conteggio approssimato di triangoli appena descritto.

Il conteggio approssimato di *quadrati* all'interno di un grafo viene ripreso, in parte dagli stessi autori di [13], in [8] in cui viene descritto un algoritmo generico per poter contare i minori di ordine 3 e 4 all'interno di un grafo diretto e non, ma non viene descritto un modo particolare per contare i *quadrati*.

# Capitolo 4

# Algoritmi per il conteggio esatto dei quadrati

Sono stati scelti tre algoritmi per il conteggio esatto dei quadrati. Per ogni algoritmo viene dato un codice identificativo, riportata una breve descrizione, un'analisi asintotica e uno pseudocodice.

## 4.1 Ricerca esaustiva

Il primo algoritmo, che identificheremo con [RES], è un banale algoritmo che, sul grafo $G$=($V$,$E$), calcola per ogni sequenza ($a$,$b$,$c$,$d$) in $G$ se sono presenti gli archi {a,b}, {b,c}, {c,d}, {a,d} in $E$ necessari per chiudere un quadrato.

Poichè le sequenze da considerare sono $\Theta(n^4)$, la complessità risultante sarà $\Omega(n^4)$. Nel caso in cui il grafo sia di modeste dimensioni, cosicchè possa essere tenuto in memoria centrale e rappresentato tramite matrice di adiacenza, il controllo della presenza o meno di un arco richiederà O(1), quindi la complessità dell'algoritmo risulterà $\Theta(n^4)$.

Tutto ciò ci suggerisce che tale algoritmo presenterà prestazioni inaccettabili già per grafi con $n$ relativamente piccolo.

Di seguito viene riportato lo pseudocodice dell'algoritmo:

**Algoritmo 4.1** RES

```
INPUT: G=(V,E)
count ← 0
for each a ∈ V
  for each b ∈ V (b≠a)
    for each c ∈ V (c≠b, c≠a)
      for each d ∈ V (d≠c, d≠b, d≠a)
        if {a,b},{b,c},{c,d},{d,a} ∈ E then
          count ← count + 1
OUTPUT: count/8
```

Il numero dei quadrati restituito dall'algoritmo è $^{count}/_{8}$, poichè, fissati i vertici $a$, $b$, $c$, $d$, in $V$ le sequenze $(a,b,c,d)$, $(b,c,d,a)$, $(c,d,a,b)$ e $(d,a,b,c)$ e le sequenze $(d,c,b,a)$, $(a,d,c,b)$, $(b,a,d,c)$ e $(c,b,a,d)$, ottenute leggendo da destra a sinistra le quattro sequenze riportate precedentemente, identificano lo stesso quadrato.

## 4.2 Conteggio con tecnica *backtracking*

Il secondo algoritmo per contare esattamente i quadrati, che indicheremo con [BTR], è un algorimo più raffinato di [RES] in quanto usa la tecnica del *backtracking*: man mano che si va a costruire la sequenza $(a,b,c,d)$ si controlla se esistono nell'ordine gli archi {a,b}, {b,c}, {c,d}, {a,d} in $E$. Nel caso in cui uno di questi sia inesistente in $E$, risulterebbe inutile continuare con la costruzione della sequenza poichè anche le successive sequenze che si andranno a costruire non andranno a produrre quadrati.

Inizialmente il primo nodo si potrà scegliere fra tutti gli $a \in V$ e questo costa $\Theta(n)$. Per ogni nodo $a$ generato si visitano tutti quanti gli adiacenti $b$ che per definizione sono $d_a$. Lo stesso verrà fatto per il terzo nodo, sia esso $c$, che verrà scelto fra gli adiacenti di $b$ e per il quarto, sia esso $d$, che verrà scelto fra gli adiacenti di $c$. Infine, solo nel caso in cui esista l'arco $\{a,d\}$ in $E$, si può affermare che la sequenza $(a,b,c,d)$ forma un quadrato e questo costa O(1) se il grafo è stato rappresentato mediante matrice di adiacenza, altrimenti costa $O(max.\{d_a, d_d\})$ nel caso in cui si è usata una rappresentazione mediante liste di adiacenza.

Per come è stato descritto l'algoritmo, conviene rappresentare il grafo mediante liste di adiacenza per la costruzione della sequenza e usare la versione in matrice di adiacenza per controllare l'esistenza dell'arco $\{a,d\}$ in $E$.

La complessità dell'algoritmo per quanto detto in precedenza è pari a $O(n+m)$, per la creazione della matrice di adiacenza a partire dalle liste di adiacenza, più $O(nd^3)$, dovuto all'esecuzione dell'algoritmo.

Nel caso pessimo, ovvero quando $d=\Theta(n)$, questa implementazione da $O(n^4)$, che non rappresenterebbe un guadagno rispetto al tempo asintotico dell'algoritmo naive [RES]. Nel caso in cui i grafi rappresentano istanze reali, con dunque $d=O(1)$, l'algoritmo avrà complessità $O(m)$.

Di seguito viene riportato lo pseudocodice:

**Algoritmo 4.2** BTR

```
INPUT: G=(V,E)
count ← 0
for each a ∈ V_G
  for each b ∈ adj(a)
    for each c ∈ adj(b) (c≠a)
      for each d ∈ adj(c) (d≠b, d≠a)
        if {d,a} ∈ E then
        count ← count + 1
OUTPUT: count/8
```

Il numero dei quadrati restituito dall'algoritmo è $^{count}/_{8}$ per lo stesso motivo riportato per [RES].

## 4.3 Conteggio mediante *2path*

Il terzo algoritmo, identificato con [DPT], per contare i quadrati sfrutta i *2path*.

Due distinti *2path* con stessi estremi individuano un quadrato e conseguentemente $k$ ($\geq 2$) distinti *2path* con gli stessi estremi individuano $\binom{k}{2}$ quadrati. Infine il numero di quadrati restituito dall'algoritmo è la metà dei quadrati contati, poichè ogni quadrato $(a,b,c,d)$ viene conteggiato due volte, prima a causa dei due *2path* con estremi $(a,c)$ e successivamente per i due *2path* con estremi $(b,d)$.

Di seguito viene riportato lo pseudocodice:

**Algoritmo 4.3** DPT

INPUT: G=(V,E)
$count \leftarrow 0$
$P \leftarrow \varnothing$
for each *2path* $(a,x,b)$ in $G$
    $P \leftarrow P \bigcup (a,b)$
for each $(c,d) \in P$
    calcola la molteplicità $k$ di $(c,d)$ in $P$
if $k \geq 2$ then
    $count \leftarrow count + \binom{k}{2}$
OUTPUT: $count/2$

La lista $P$ viene costruita in tempo O($|P|$) scorrendo le liste di adiacenza del grafo $G$. Successivamente P viene ordinata rispetto al primo e poi al secondo valore delle coppie di vertici che contiene e questo richiede tempo O($|P|$ *log* $|P|$). In $P$ ora coppie con stessi estremi compaiono in posizioni adiacenti di conseguenza per calcolare le moltiplicità delle coppie basta scorrere $P$ e questo richiede tempo O($|P|$).

Per quanto detto la complessità risulterà O($|P|$ *log* $|P|$). Si osservi che $|P|$ è il numero dei *2path*, vale a dire che

$$|P| = \sum_{v \in V} \frac{d_v(d_v-1)}{2} = O(nd^2) \tag{4.1}$$

Di conseguenza la complessità è

$$\mathrm{O}(nd^2 \ log \ n)$$

Quindi nel caso pessimo, ovvero quando $d{=}\Theta(n)$, questa implementazione avrà tempo $\mathrm{O}(n^3 \ log \ n)$, che comunque rappresenta un guadagno rispetto al tempo asintotico dell'algoritmo naive [RES]. Nel caso in cui i grafi rappresentano istanze reali, con dunque $d{=}\mathrm{O}(1)$, questo tipo l'algoritmo avrà complessità $\mathrm{O}(n \ log \ n)$.

# Capitolo 5

# Algoritmi approssimanti

Gli algoritmi approssimanti, in contrapposizione a quelli esatti, propongono una stima del conteggio esatto.

In questo capitolo verranno descritte tre diverse ($\delta$-$\epsilon$)-approssimazioni per il problema del conteggio dei quadrati in un grafo.

*Definizione 5.1.* Un algoritmo probabilistico per un problema di conteggio è ($\delta$-$\epsilon$)-*approssimante* se $\forall\epsilon,\delta \in (0,1)$ produce con una probabilità maggiore o uguale a 1-$\delta$ una stima $\widetilde{X}$ che dista al più un fattore $\epsilon$ dal conteggio esatto di $X$. Vale a dire:

$$P\left(\left|\widetilde{X} - X\right| > \epsilon X\right) < \delta$$

Per dimostrare che gli algoritmi proposti sono ($\delta$-$\epsilon$)-approssimanti si farà uso del seguente teorema la cui prova si basa sulla *disuguaglianza di Chernoff-Hoeffding* [7].

**Teorema 1** *Siano $\beta_1$, $\beta_2$,...,$\beta_S$ variabili aleatorie identicamente distribuite a valori nell'intervallo [0,1] con valore medio E[$\beta$] se*

$$S \geq \frac{3}{\epsilon^2}\frac{1}{E[\beta]}log\frac{2}{\delta} \text{ allora}$$

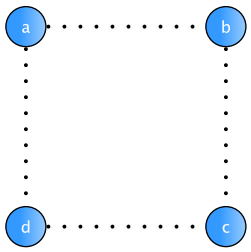
$$P\left(\left|\frac{1}{S}\sum_{i=1}^{S}\beta_i - E[\beta]\right| > \epsilon E[\beta]\right) < \delta$$

ovvero che gli S campionamenti producono una ($\delta$-$\epsilon$)-approssimazione di E[$\beta$].

## 5.1 Campionamento mediante quattro vertici

Questo algoritmo, che per semplicità denomineremo [CQV], utilizza il campionamento mediante una quadrupla di vertici $(a,b,c,d)$ (Figura 5.1) con $a$, $b$, $c$ e $d$ presi con ugual probabilità in $V$ e la successiva ricerca degli archi $\{a,b\}$, $\{b,c\}$, $\{c,d\}$, $\{d,a\}$ all'interno di $E$.

Figura 5.1: Campionamento CQV




Di seguito viene proposto lo pseudocodice dell'algoritmo di campionamento:

**Algoritmo 5.1** Campionamento CQV

INPUT: G=(V,E)

$\beta \leftarrow 0$
Scegli a caso una possibile quadrupla $(a,\ b,\ c,\ d)$
if $\{a,b\}$, $\{b,c\}$, $\{c,d\}$, $\{d,a\} \in E$ then
$\beta \leftarrow 1$

OUTPUT: $\beta$

**Lemma 1** *L'algoritmo di campionamento* CQV *restituisce una $\beta$ tale che*

$$E[\beta] = \frac{8Q}{24\binom{n}{4}} = \frac{Q}{3\binom{n}{4}}$$

La prova del lemma si basa sul fatto che ciascuna quadrupla ha la stessa probabilità $\frac{1}{24\binom{n}{4}}$ di essere campionata e solo $8Q$ delle quadruple faranno sì che verra restituito $\beta$=1.

L'algoritmo che fa uso di questo tipo di campionamento producendo una ($\delta$-$\epsilon$)-approssimazione lavora come segue: esegue

$$S \geq \frac{9}{\epsilon^2}\,\frac{\binom{n}{4}}{Q}\,log\,\frac{2}{\delta}$$

campionamenti ricavando i valori $\beta_1$,$\beta_2$,...,$\beta_S$ e restituisce come stima di $Q$ il valore

$$\tilde{Q} = 3\binom{n}{4}\left(\frac{1}{S}\sum_{i=1}^{S}\beta_i\right) \tag{5.1}$$

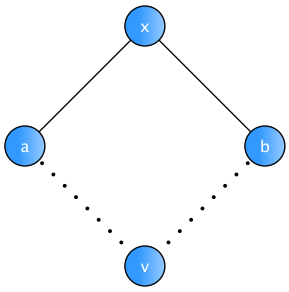
La prova che è effettivamente è una ($\delta$-$\epsilon$)-approssimazione è la seguente: dal Teorema 1 ricaviamo che il valore $\frac{1}{S}\sum_{i=1}^{S}\beta_i$ è una ($\delta$-$\epsilon$)-approssimazione di E[$\beta$] e che per il Lemma 1 vale $\frac{Q}{3\binom{n}{4}}$.

Per come viene calcolato nell'equazione 5.1 ricaviamo che la stima $\widetilde{Q}$ è una ($\delta$-$\epsilon$)-approssimazione di $Q$.

## 5.2 Campionamento mediante un *2path* e un vertice

Questo algoritmo, che chiameremo [CPV], utilizza il campionamento mediante un qualunque *2path* $TP$=($a$,$x$,$b$) e un qualunque vertice $v$ che non sia uno dei vertici di $TP$ (Figura 5.2e la ricerca degli archi $\{a,v\}$ e $\{b,v\}$ in $E$. Anche in questo caso, come nel campionamento [CQV], gli elementi del campionamento vengono presi con ugual probabilità in $G$.

Figura 5.2: Campionamento CPV




Di seguito viene riportato lo pseudocodice dell'algoritmo di campionamento:

**Algoritmo 5.2** Campionamento CPV

INPUT: G=(V,E)

$\beta \leftarrow 0$
Scegli a caso un *2path* ($a$,$x$,$b$) e un vertice $v$ t.c $v \notin \{a,x,b\}$
if $\{v,b\}$ e $\{v,a\} \in E$ then
  $\beta \leftarrow 1$

OUTPUT: $\beta$

**Lemma 2** *L'algoritmo di campionamento* CPV *restituisce una* $\beta$ *tale che*

$$E[\beta] = \frac{4Q}{|P|(n-3)}$$

La prova del lemma si basa sul fatto che ciascuno dei $|P|(n$-3) campioni ha la stessa probabilità di essere campionato e di questi solo $4Q$ restituiscono $\beta$=1.

L'algoritmo che fa uso di questo tipo di campionamento produce una ($\delta$-$\epsilon$)-approssimazione lavora come segue: esegue

$$S \geq \frac{3}{\epsilon^2} \frac{|P|(n-3)}{4Q} \; log \; \frac{2}{\delta}$$

campionamenti ricavando i valori $\beta_1$,$\beta_2$,...,$\beta_S$ e restituisce come stima di $Q$ il valore

$$\tilde{Q} = \left( \frac{1}{S} \sum_{i=1}^{S} \beta_i \right) \frac{|P|(n-3)}{4} \tag{5.2}$$

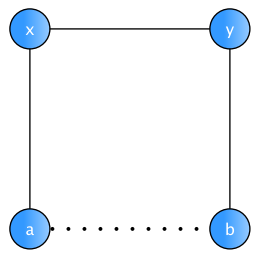
La prova che è effettivamente è una ($\delta$-$\epsilon$)-approssimazione è la seguente: dal Teorema 1 ricaviamo che il valore $\frac{1}{S}\sum_{i=1}^{S} \beta_i$ è una ($\delta$-$\epsilon$)-approssimazione di E[$\beta$] che per il Lemma 2 vale $\frac{4Q}{|P|(n-3)}$.

Di conseguenza per come viene calcolato nell'equazione 5.2 ricaviamo che la stima di $\widetilde{Q}$ è una ($\delta$-$\epsilon$)-approssimazione di $Q$.

## 5.3 Campionamento mediante un *3path*

Questo algoritmo, che viene denominato [CTP], propone il campionamento mediante un qualsiasi *3path* $P$=($a$,$x$,$y$,$b$) presente in $G$ (Figura 5.3) e la ricerca dell'arco $\{a,b\}$ in $E$.

Figura 5.3: Campionamento CTP




Viene riportato di seguito lo pseudocodice di questo algoritmo:

**Algoritmo 5.3** Campionamento CTP

INPUT: G=(V,E)

$\beta \leftarrow 0$
Scegli a caso un arco {x,y} in $E$
Fra gli $adj(x)$ scegli a caso un vertice $a$
Fra gli $adj(y)$ scegli a caso un vertice $b$
if $a \neq y$ AND $b \neq x$ AND $\{a,b\} \in E_G$ then
  $\beta \leftarrow \dfrac{d_X \cdot d_y}{4}$

OUTPUT: $\beta$

**Lemma 3** *L'algoritmo di campionamento* CTP *restituisce una* $\beta$ *tale che*

$$E[\beta] = \frac{Q}{m}$$

La prova del lemma si basa sul fatto che la probabilità che nel campionamento [CTP] venga scelto l'arco $(x,y)$ e poi ottenuta la quaterna $(a,x,y,b)$ è $\dfrac{1}{2m \cdot d_x \cdot d_y}$ da cui segue che

$$E[\beta] = \sum_{(a,b,c,d) con (a,d) \in E} \frac{1}{2m \cdot d_x \cdot d_y} \cdot \frac{d_x \cdot d_y}{4} = \frac{8Q}{8m} = \frac{Q}{m}$$

L'algoritmo che fa uso di questo tipo di campionamento produce una ($\delta$-$\epsilon$)-approssimazione lavora come segue: esegue

$$S \geq \frac{3}{\epsilon^2} \frac{md^2}{Q} \, log \, \frac{2}{\delta}$$

campionamenti ricavando i valori $\beta_1$,$\beta_2$,...,$\beta_S$ e restituisce come stima di $Q$ il valore

$$\tilde{Q} = m \left( \frac{1}{S} \sum_{i=1}^{S} \beta_i \right) \tag{5.3}$$

Per dimostrare che il risultato è una ($\delta$-$\epsilon$)-approssimazione non possiamo usare direttamente il Teorema 1 poichè le variabili $\beta_1$,..., $\beta_S$ non assumono necessariamente valori nell'intervallo [0,1].

Consideriamo allora le S variabili $\beta_1^{'}$,..., $\beta_S^{'}$ con $\beta_i^{'}$=$\frac{B_i}{d^2}$. Si nota che $\beta_i$<$d^2$ da cui $\beta_i^{'}$<1, quindi su queste nuove variabili è possibile applicare il Teorema 1: dal Lemma 3 ricaviamo che E[$\beta^{'}$]=$\frac{Q}{md^2}$ e applicando ora il Teorema 1 si può affermare che $\frac{1}{S}\sum_{i=1}^{S} \beta_i^{'}$ è una ($\delta$-$\epsilon$)-approssimazione di E[$\beta^{'}$], di conseguenza $\frac{1}{S}\sum_{i=1}^{S} \beta_i$ è una ($\delta$-$\epsilon$)-approssimazione di E[$\beta$] perchè $\frac{1}{S}\sum_{i=1}^{S} \beta_i = \frac{d^2}{S}\sum_{i=1}^{S} \beta_i^{'}$.

Di conseguenza per come viene calcolato nell'equazione 5.3 ricaviamo che la stima di $\widetilde{Q}$ è una ($\delta$-$\epsilon$)-approssimazione di $Q$.

# Capitolo 6

# Risultati sperimentali

Questa sezione riporta i risultati sperimentali che sono stati ottenuti, prima per il calcolo esatto dei quadrati e successivamente per il calcolo approssimato. I codici sono stati scritti in C e compilati con il compilatore gcc versione 4.2. Gli esperimenti sono stati fatti su un *Macbook Pro Late 2011* [5] con a disposizione un *Intel Core i7* da quattro core con 2.2 Ghz, 8 Gb di Ram DDR3 a 1333Mhz, 80Gb di spazio libero su Solid State Drive.

Per il calcolo del tempo impiegato dagli algoritmi esatti è stata usata la funzione *time* [1] di UNIX tenendo conto del tempo *user* e *sys*, mentre per il calcolo del tempo impiegato dagli algoritmi approssimanti è stata utilizzata la funzione *time* [2] della libreria *C <time.h>*.

Questa scelta è stata fatta poichè nel codice degli algoritmi approssimanti viene eseguita una subroutine necessaria per creare il grafo in memoria affinchè la routine dell'algoritmo lo possa manipolare. Negli algoritmi esatti l'intero codice rappresenta l'algoritmo, conseguentemente viene più comodo usare la funzione *time* della libreria *C,* andando a prendere in considerazione il tempo usato per le subroutine *user mode* sommato al tempo usato per le subroutine in *kernel mode.*

## 6.1 Datasets

I grafi usati provengono quasi tutti dal *Large Network Dataset* della *Stanford University*, sono scaricabili da [4] e sono grafi il cui numero di archi arriva fino a circa $400k$.

### 6.1.1 Lavoro di preprocessing

Non tutti i grafi usati etichettavano i propri nodi da 0 a $|V|$-1, così un primo lavoro che è stato effettuato è stato quello di rappresentare tutti i grafi usati in questo tirocinio sotto quest'ottica.

Un secondo lavoro è stato quello di eliminare i cappi, ovvero quegli archi del tipo $(a,a)$, da ogni grafo.

#### Grafi utilizzati

- *Erdős–Rényi* $_{300,0.035}$ [14]: un grafo randomico costruito sul modello di *Erdős–Rényi,* la cui costruzione si è realizzata mediante la libreria C presente in [3] e secondo i parametri $n$ e $p$[1] pari rispettivamente a 300 e 0.035.
  Si sono scelti questi due valori per i parametri per far vedere che l'algoritmo [RES] usa tempi accettabili solo con grafi di piccolissima taglia e già con grafi di taglia poco più grande (ad esempio il più piccolo dei *as*AAAAMMGG, *as*19971109) i tempi diventano del tutto inaccettabili.

- *as*AAAAMMGG [10]: tre instantanee di un sistema autonomo. Ogni sistema autonomo comunica il flusso di scambi ai propri vicini e così si è potuto ricostruire questa rete di scambi tramite i BGP logs.
  È stato scelto questo insieme di tre grafi per le loro modeste dimensioni e per vedere le stesse caratteristiche di una stessa struttura crescente.

- *Email-Enron* [6][9]: il grafo rappresenta una rete di comunicazioni via email, originalmente pubblica e postata sul web dalla Federal Energy Regulatory Commission durante le sue investigazioni.
  I nodi corrispondono agli indirizzi email e un arco $\{i, j\}$ corrisponde all'invio di un'e-mail fra $i$ e $j$.
  Questo tipo di grafo si è scelto per un numero discreto di nodi e archi.

[1] $n$ è il numero di vertici e $p$ è la probabilità con la quale un arco possa congiungere due nodi $a$ e $b$

- *Erdős–Rényi* $_{3011,0.045}$ [14]: un grafo randomico costruito sul modello di *Erdős–Rényi,* secondo i parametri $n$ e $p$ pari rispettivamente a 3011 e 0.045.
  Si sono scelti questi due precisi parametri per vedere come l'ordinamento utilizzato per il conteggio preciso, per un $d$ sufficientemente grande, ha un costo considerevolmente più alto e per effettuare ciò verranno messi in comparazione i tempi su questo grafo con i tempi sul grafo *as19971109* che appunto ha 3011 nodi.

- *Dblp.ungraph* [11]: il *DBLP Computer Science Bibliography* ha fornito un elenco di articoli pubblicati di carattere informatico. Da questo elenco viene costruito un grafo i cui sono sono gli autori e tra gli autori è presente un arco se i due hanno almeno un articolo in comune.
  Questo grafo, oltre ad essere il primo grafo descritto che supera il milione di archi, è stato scelto prevedendo un alto numero di quadrati al suo interno.

## 6.2 Risultati sperimentali degli algoritmi esatti

Nella Tabella 6.1 sono riportate succintamente alcune delle caratteristiche salienti del dataset utilizzato. Alcune già note (vertici e archi), altre calcolate qui per la prima volta grazie agli algoritmi di conteggio esatto implementati (*2path*, quadrati).

Tabella 6.1: Dataset

| *Nome grafo* | *Vertici* | *Archi* | *d medio* | $\|P\|$ | *Quadrati* |
|---|---|---|---|---|---|
| Erdős–Rényi $_{300,0.035}$ | 300 | 1.590 | 10,6 | 16.857 | 1.636 |
| as19971109 | 3.011 | 5.343 | 3,5 | 468.720 | 61.245 |
| as19990923 | 5.803 | 11.823 | 4,1 | 1.659.840 | 241.790 |
| as20000102 | 6.474 | 7.391 | 4,1 | 2.069.787 | 299.317 |
| Erdős–Rényi $_{3011,0.045}$ | 3.011 | 204.141 | 135,6 | 27.671.163 | 42.253.872 |
| Email-Enron | 36.692 | 183.831 | 10,0 | 25.566.893 | 36.262.229 |
| Dblp.ungraph | 317.080 | 1.049.866 | 6,6 | 21.780.888 | 55.107.654 |

Nella Tabella 6.2 si riportano i tempi di computazione $T_{RES}$, $T_{BTR}$, $T_{DPT}$, dove è stato possibile calcolarli, dei tre algoritmi usati per il conteggio esatto.

Tabella 6.2: Tempi di computazione per algoritmi esatti

| *Nome Grafo* | $T_{RES}$ | $T_{BTR}$ | $T_{DPT}$ |
|---|---|---|---|
| Erdős–Rényi$_{300,0.035}$ | 17,84s | 0,033s | 0,007s |
| as19971109 | >5000s | 6,77s | 0,136s |
| as19990923 | ? | 46,37s | 0,491s |
| as20000102 | ? | 11,33s | 0,629s |
| Erdős–Rényi$_{3011,0.045}$ | ? | 639,93s | 9,354s |
| Email-Enron | ? | ? | 8,918s |
| Dblp.ungraph | ? | ? | 8,562s |

Nella Figura 6.1 un grafico ad istogramma riporta i dati della Tabella 6.2 in confronto.

Figura 6.1: Tempi di computazione per algoritmi esatti a confronto

### 6.2.1 Commenti

Come evidenziato dalla Tabella 6.1 i grafi che catturano istanze reali, tipicamente, hanno un grado $d$ medio molto basso se paragonato al numero di vertici. La sola eccezione è rappresentata dal grafo *Erdős–Rényi*$_{3011,0.045}$ costruito sul modello di Erdős–Rényi, il quale volutamente è stato costruito per ottenere un $d$ asintoticamente vicino ad $n$. Questo grafo è stato considerato per mostrare come l'analisi teorica del Capitolo 4, riguardante gli algoritmi per il conteggio esatto, venga confermata.

Principalmente sul tempo usato per calcolare esattamente i quadrati con il metodo dei *2path* grava asintoticamente il tempo usato per ordinare le coppie di *2path* trovati all'interno del grafo che, quando $d$ è asintoticamente vicino ad $n$ è $O(n^3 \ log \ n)$, mentre per $d$=O(1) è pari ad $O(n \ log \ n)$.

Questa sperimentazione ha evidenziato proprio questa differenza. Nei grafi *as19971109* e *Erdős–Rényi*$_{3011,0.045}$ notiamo dalla Tabella 6.1 uno stesso numero di nodi, ma un differente numero di archi. La Tabella 6.2 mostra come i tempi di computazione, facendo fuoco sul dettaglio sopra riportato, aumentano vertiginosamente (il tempo impiegato per calcolare il grafo *Erdős–Rényi*$_{3011,0.045}$ è circa 65 volte più grande rispetto a quello usato per lo stesso calcolo sul grafo *as19971109*) andando a confermare l'analisi teorica.

Si nota dalla Figura 6.1 e dalla Tabella 6.2 come non sempre si è potuto usare l'algoritmo [RES] in quanto molto inefficiente e i tempi di calcolo sono considerevoli già per il grafo più piccolo e come, in tutti i grafi selezionati per gli esperimenti, l'algoritmo [BTR] risulta meno efficiente dell'algoritmo [DPT].

## 6.3 Risultati sperimentali degli algoritmi approssimanti

La Tabella 6.3 descrive il conteggio approssimato e i tempi d'esecuzione, riportati in millisecondi, dell'algoritmo di campionamento [CPV]. Come insiemi di campione vengono scelti campioni con cardinalità $20k$, $50k$, $100k$ e $500k$ e i risultati riportati nella tabella sono frutto di almeno trenta test mediati.

Nella tabella verrà usato $\tilde{Q}$ per indicare il numero di quadrati stimati, *dev.* per indicare il valore assoluto dello spostamento in percentuale di $\tilde{Q}$ da $Q$, ovvero *dev.* $= \left|\frac{Q-\tilde{Q}}{Q}\right|$, e infine $T$ per indicare il tempo di esecuzione.

Tabella 6.3: Conteggio approssimato e tempi di computazione con campionamento [CPV]

| *Nome Grafo* | *20k* | | | *50k* | | | *100k* | | | *500k* | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $\tilde{Q}$ | *dev.* | $T$ | $\tilde{Q}$ | *dev.* | $T$ | $\tilde{Q}$ | *dev.* | $T$ | $\tilde{Q}$ | *dev.* | $T$ |
| E.R.$_{300,0.035}$ | 1.585 | 3,1% | 7 | 1.637 | 0,1% | 17 | 1.641 | 0,3% | 32,6 | 1.635 | 0,1% | 159 |
| as19971109 | 65.561 | 7,0% | 26 | 57.101 | 6,8% | 67 | 57.806 | 5,6% | 135 | 61.846 | 1,0% | 69 |
| as19990923 | 259.931 | 7,5% | 48 | 250.304 | 3,5% | 120 | 247.897 | 2,5% | 242 | 246.934 | 2,1% | 1.216 |
| as20000102 | 244.433 | 18,3% | 54 | 294.659 | 1,6% | 132 | 303.030 | 1,2% | 256 | 296.199 | 1,0% | 1.332 |
| E.R.$_{3011,0.045}$ | 40.431.332 | 4,3% | 27 | 41.700.664 | 1,3% | 64 | 41.754.767 | 1,2% | 132 | 41.908.751 | 0,8% | 663 |
| Email-Enron | 32.830.831 | 9,5% | 37 | 34.550.541 | 4,7% | 90 | 34.785.047 | 4,1% | 180 | 35.848.140 | 1,1% | 886 |
| dblp.ungraph | 50.070.085 | 9,1% | 30 | 50.645.603 | 8,1% | 50 | 58.127.339 | 5,5% | 84 | 54.098.713 | 1,8% | 355 |

La Tabella 6.4 descrive invece il conteggio approssimato e i tempi d'esecuzione, anche in questo caso riportati in millisecondi, dell'algoritmo di campionamento [CTP]. Come insiemi di campione vengono scelti campioni con cardinalità 100, $2k$, $5k$ e $10k$ e anche in questo caso i risultati sono frutto di almeno trenta test mediati. In questa tabella $\widetilde{Q}$, *dev.* e $T$ assumono lo stesso significato già descritto per la Tabella 6.3.

Tabella 6.4: Conteggio approssimato e tempi di computazione con campionamento [CTP]

| *Nome Grafo* | 100 | | | *2k* | | | *5k* | | | *10k* | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $\bar{Q}$ | *dev.* | $T$ | $\bar{Q}$ | *dev.* | $T$ | $\bar{Q}$ | *dev.* | $T$ | $\bar{Q}$ | *dev.* | $T$ |
| E.R.$_{300,0.035}$ | 1.367 | 16,4% | 0,1 | 1.704 | 4,2% | 0,5 | 1.679 | 2,6% | 1,4 | 1.622 | 0,9% | 2,8 |
| as19971109 | 51.774 | 15,5% | 0,4 | 61.328 | 0,14% | 1,5 | 61.178 | 0,11% | 3,9 | 61.186 | 0,10% | 9,5 |
| as19990923 | 210.686 | 12,9% | 0,3 | 256.973 | 6,3% | 2,3 | 233.935 | 3,2% | 5,4 | 244.177 | 1,0% | 10,4 |
| as20000102 | 355.584 | 18,8% | 0,5 | 290.912 | 2,8% | 2,4 | 291.124 | 2,7% | 5,7 | 294.972 | 1,5% | 41,5 |
| E.R.$_{3011,0.045}$ | 40.431.332 | 12,6% | 2,0 | 43.964.020 | 4% | 4,2 | 43.308.160 | 2,5% | 7,9 | 43.080.098 | 1,9% | 13,8 |
| Email-Enron | 32.608.862 | 10,1% | 2,5 | 35.658.315 | 1,7% | 5,3 | 36.801.054 | 1,5% | 10,1 | 36.574.898 | 0,9% | 17,9 |
| dblp.ungraph | 51.303.277 | 6,9% | 16,1 | 53.931.490 | 2,1% | 17,7 | 55.676.569 | 1,0% | 20,9 | 55.108.124 | 0,001% | 25,2 |

Nei risultati sperimentali non verrà riportata la sperimentazione dell'algoritmo di campionamento [CQV], il quale verrà usato solo al momento della comparazione di questo con i rimanenti algoritmi di campionamento su determinati grafi.

La Tabella 6.5 riporta lo *speedup*. Descrive quindi per ogni grafo il rapporto che c'è fra il tempo $T_{DPT}$ utilizzato per il calcolo esatto dall'algoritmo esatto [DPT] e il tempo utilizzato per il calcolo approssimato dall'algoritmo di campionamento [CPV], ovvero *speedup* $= \frac{T_{DPT}}{T_{CPV}}$, per ogni campione scelto.

Tabella 6.5: *Speedup* per campionamento [CPV]

| *Nome Grafo* | *20k* | *50k* | *100k* | *500k* |
|---|---|---|---|---|
| Erdős–Rényi$_{300,0.035}$ | 1,00 | 2,44 | 4,66 | 22,65 |
| as19971109 | 0,20 | 0,50 | 1,00 | 5,07 |
| as19990923 | 0,10 | 0,25 | 0,49 | 2,48 |
| as20000102 | 0,30 | 0,74 | 1,43 | 7,44 |
| Erdős–Rényi$_{3011,0.045}$ | 0,003 | 0,007 | 0,014 | 0,071 |
| Email-Enron | 0,004 | 0,010 | 0,020 | 0,099 |
| Com-dblp.ungraph | 0,005 | 0,008 | 0,015 | 0,071 |

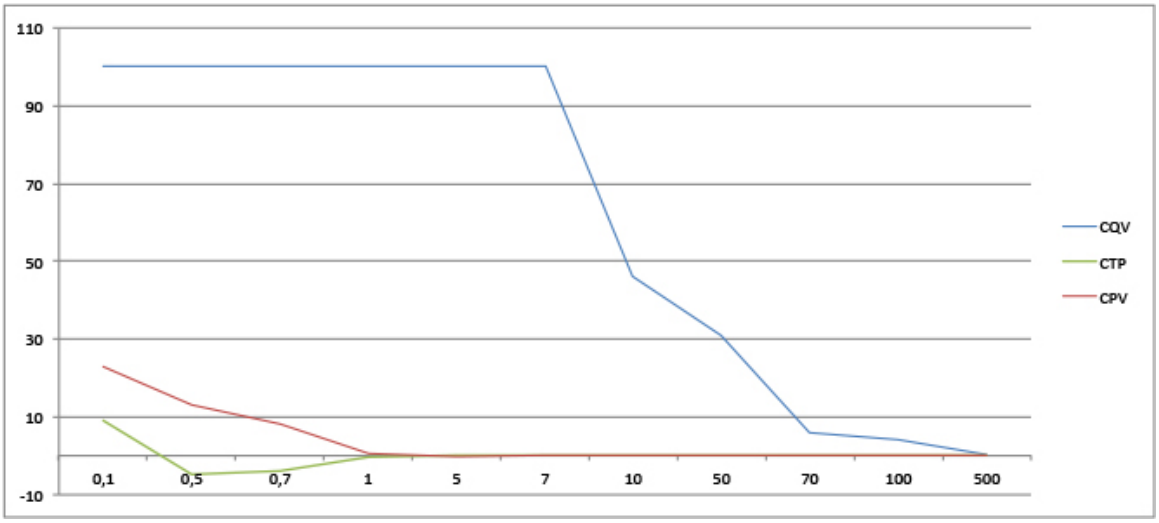
Allo stesso modo la Tabella 6.6 riporta lo *speedup* dell'algoritmo di campionamento [CTP], dove in questo caso lo speedup indica il rapporto fra il tempo $T_{DPT}$ utilizzato per il calcolo esatto dall'algoritmo esatto [DPT] e il tempo utilizzato per il calcolo approssimato dall'algoritmo di campionamento [CTP], ovvero *speedup* $= \frac{T_{DPT}}{T_{CTP}}$, per ogni campione scelto.

Tabella 6.6: Speedup per campionamento [CTP]

| *Nome Grafo* | 100 | *2k* | *5k* | *10k* |
|---|---|---|---|---|
| Erdős–Rényi$_{300,0.035}$ | 0,0143 | 0,0831 | 0,2066 | 0,4057 |
| as19971109 | 0,0029 | 0,0116 | 0,0289 | 0,0697 |
| as19990923 | 0,0006 | 0,0047 | 0,0109 | 0,0211 |
| as20000102 | 0,0028 | 0,0136 | 0,0318 | 0,0673 |
| Erdős–Rényi$_{3011,0.045}$ | 0,0002 | 0,0005 | 0,0009 | 0,0015 |
| Email-Enron | 0,0003 | 0,0006 | 0,0011 | 0,0195 |
| Com-dblp.ungraph | 0,0018 | 0,0020 | 0,0024 | 0,0029 |

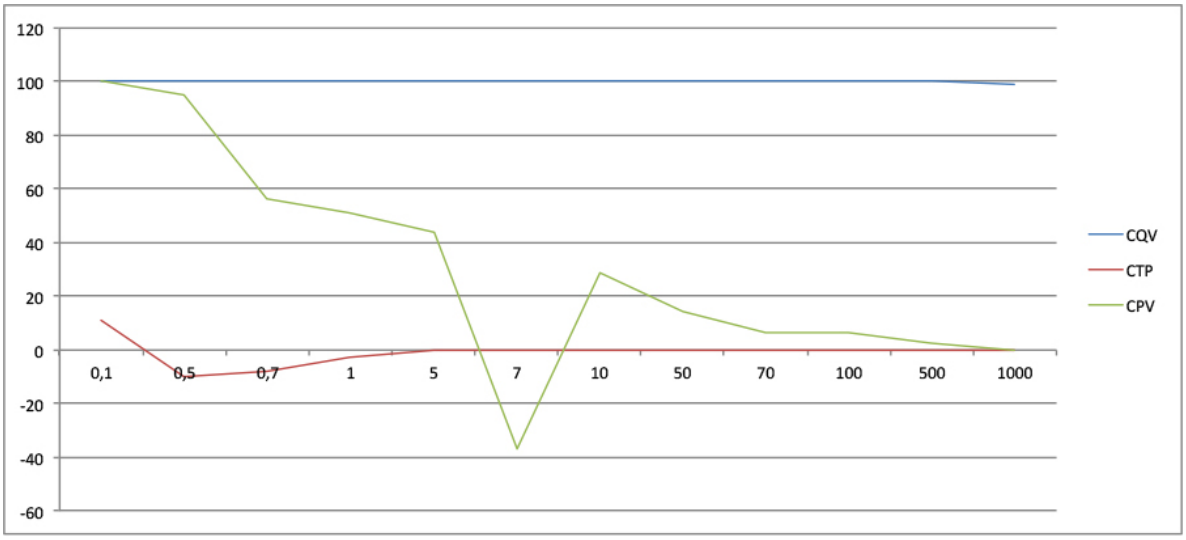
La Figura 6.2 mette a confronto i tempi di convergenza sul grafo Erdős–Rényi$_{300,0.035}$ dei tre algoritmi di campionamento presentati. Nel grafico viene rappresentato con le ascisse il tempo espresso in millisecondi, mentre con le ordinate viene rappresentata la *dev.* calcolata come nel caso della Tabella 6.3. In questo caso viene dato in input agli algoritmi i tempi di esecuzione e viene restituita la *dev.* . Per questo particolare test vengono scelti come input i seguenti tempi espressi in millisecondi: 0.1, 0.5, 0.7, 1, 5, 7, 10, 20, 50, 100, 200 e 500.

Figura 6.2: Confronto di convergenza fra i tre algoritmi sul grafo *Erdős–Rényi*$_{300,0.035}$



Nella Figura 6.3 vengono messi a confronto i tempi di convergenza sul grafo as19971109 dei tre algoritmi di campionamento presentati. Non varia la rappresentazione del grafico e la modalità degli esperimenti, caratteristiche già enunciate per la descrizione della Figura 6.2. Per quanto riguarda i test, invece, viene aggiunto il test che ha come input tempo 1 secondo.

Figura 6.3: Confronto di convegenza fra [CTP] e [CPV] sul grafo *as19971109*



### 6.3.1 Commenti

Dalle Tabelle 6.3 e 6.4 si evince come un conteggio più accurato si ottiene facendo un campionamento con campioni man mano in quantità crescente. D'altro canto dalle Tabelle 6.5 e 6.6 si evince come, un campionamento eccessivamente grande, si rischia di raggiungere tempi di calcolo simili a quelli usati per il conteggio esatto, il che andrebbe a vanificare il vantaggio del conteggio approssimato.

Si nota nella Tabella 6.3 come il campionamento [CPV] di cardinalità $100k$ permette di ottenere una disceta approssimazione.

Per questo tipo di campionamento la Tabella 6.5 mette in luce che per grafi piccoli (meno di $20k$ archi) il tempo usato per il conteggio approssimato si avvicina molto al tempo usato per il conteggio esatto (si può notare nella tabella che mediante campionamento [CPV] tutti i test sul grafo Erdős–Rényi$_{300,0.035}$ superano/eguagliano il tempo usato per il conteggio esatto), problema che non emerge per i grafi più grandi: per il grafo Erdős–Rényi$_{3011,0.045}$ si riesce ad ottenere un'approssimazione pari all'1,3% usando solo $50k$ campioni e un tempo pari allo 0,7% del tempo usato per il calcolo esatto.

Dalla Tabella 6.4 si nota, invece, come il campionamento [CTP] ottiene un'ottima approssimazione già con campioni di cardinalità $5k$ e la Tabella 6.6 mostra come per tutti i test effettuati viene sempre utilizzata una piccola frazione del tempo usato per il conteggio esatto.

I test effettuati per costruire le Figure 6.2 e 6.3 hanno lo scopo di mettere a confronto i tempi di convergenza degli algoritmi approssimanti.

Dalla Figura 6.2 si evince che l'algoritmo di campionamento [CQV] per produrre una buona approssimazione richiede almeno 70 millisecondi.

Gli altri algoritmi di campionamento con 5 millisecondi a disposizione hanno già raggiunto un'ottima approssimazione.

Dalla Figura 6.3 appare evidente che l'algoritmo [CTP] ha tempi di convergenza molto più rapidi dell'algoritmo [CPV]. Sul grafo in esame poco più di un millisecondo è sufficiente all'algoritmo [CTP] per ottenere un'ottima approssimazione, mentre l'algoritmo [CPV] ne richiede una quantità 200 volte superiore per produrre una soluzione della stessa qualità.

# Capitolo 7

# Conclusioni e sviluppi futuri

In questo tirocinio si sono presentati e confrontati tre algoritmi ($\delta$-$\epsilon$)-approssimanti, basati su diverse tecniche di campionamento, per il conteggio approssimato di quadrati su un grafo. Dai risultati sperimentali, a conferma dell'analisi teorica, risulta più performante l'algoritmo basato sul campionamento dei *3path*, il quale, sul grafo più grande, riesce a raggiungere una deviazione dello 0,001% anche con un campione di modeste dimensioni.

Gli algoritmi di conteggio approssimanti sono utili nel caso in cui non si è interessati necessariamente al conteggio preciso, bensì ad un'ottima stima indicativa. Un buon algoritmo approssimante per un problema di conteggio deve restituire un valore che non sia troppo lontano dal valore esatto e produrre esso in tempi accettabili.

Si può facilmente notare da [4] come tra le informazioni disponibili per un grafo non ci sia il numero di *quadrati,* il quale risulta fondamentale ai fini di questo tirocinio. Conseguentemente si son dovuti costruire algoritmi esatti appositi (Capitolo 4) al fine di reperire questa informazione.

L'algoritmo più efficiente per il conteggio esatto di quadrati, denominato [DPT] (Sezione 4.3), si basa principalmente sulla creazione di una struttura d'ausilio, la lista dei *2path*, per calcolare poi definitivamente il numero di quadrati presenti all'interno del grafo.

Come si è visto dall'equazione 4.1 della Sezione 4.3, il numero di *2path* è $O(nd^2)$ e che quindi dipende fortemente, nel caso di istanze reali in cui tipicamente $d$=O(1), dal numero di vertici presenti all'interno del grafo. Conseguentemente, nel caso di un grafo avente un numero di vertici abbastanza grande, la memoria principale della macchina potrebbe non

accogliere la lista dei *2path* del grafo in questione. Si passa quindi da un modello di memoria interna ad un *modello di memoria esterna*, in cui le operazioni che riguardavano i *2path*, ordinamento e conteggio, vengono effettuate su file.

La lista $P$ viene costruita, anche in questo caso in tempo O($|P|$), scorrendo le liste di adiacenza del grafo $G$. Questa volta ogni elemento viene salvato all'interno di un file, poichè presupponiamo che la memoria centrale non potrebbe accogliere l'intera lista $P$.

Per poter effettuare il sort su memoria esterna viene utilizzato un mergesort iterativo a *k-vie* descritto in [12], il cui costo è pari a

$$O(\frac{n}{B} log_{M/B} \frac{n}{M}) \tag{7.1}$$

in cui:

- $n$ sono gli elementi da ordinare, nel nostro caso $|P|$
- $M$ è la memoria principale della macchina messa a disposizione per l'algoritmo
- $B$ è la dimensione dei blocchi con cui si è divisa la lista $P$

Per quanto detto $|P|$ è uguale a O($nd^2$) e conseguentemente per 7.1 il tempo totale di questa implementazione è pari a

$$\mathrm{O}(\frac{nd^2}{B} log_{M/B} \frac{n}{M})$$

in cui nel caso peggiore, ovvero quando $d$=O($n$) si ha che il costo totale è pari a O$(\frac{n^3}{B} log_{M/B} \frac{n}{M})$, mentre nel caso peggiore, ovvero quando $d$=O(1) si ha che il costo totale è pari a O$(\frac{n}{B} log_{M/B} \frac{n}{M})$.

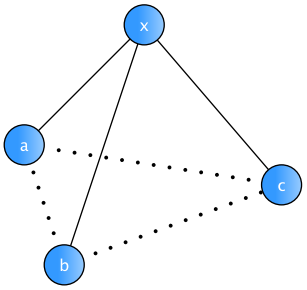
Nel modello di memoria esterna il conteggio dei *2path* aventi stessi estremi viene effettuato nella stessa maniera con cui si operava per l'implementazione su memoria principale.

Un altro aspetto che sarebbe interessante studiare è il calcolo approssimato di *clique di ordine 4* andando a generalizzare gli algoritmi di campionamento [CTP] e [CQV]: anzichè utilizzare il *2path* viene utilizzata una struttura, che verrà chiamata per semplicità *grappolo da 3*.

*Definizione 7.1.* Un *grappolo da 3 su* $x$ è una coppia $(x,\{a,\ b,\ c\})$ formata da un elemento $x \in V$ t.c. $d_v \geq 3$ e un insieme $\{a,\ b,\ c\}$, dove $a,\ b,\ c \in adj(x)$

L'algoritmo di campionamento che usa il *grappolo da 3* andrebbe a scegliere un qualunque *grappolo da 3* $(x,\ \{a,\ b,\ c\})$ e successivamente andrebbe a controllare se gli archi $\{a,b\}$, $\{b,c\}$, $\{c,a\}$ siano presenti in $E$.

Figura 7.1: Campionamento mediante *grappolo da 3*



# Bibliografia


[1] Funzione time in unix. URL: http://it.wikipedia.org/wiki/Time_(Unix).

[2] Funzione time nella libreria <time.h>. URL: http://www.cplusplus.com/reference/ctime/.

[3] Igraph. URL: http://igraph.sourceforge.net.

[4] Large network dataset of stanford university. URL: http://snap.stanford.edu/data/.

[5] Macbookpro late 2011 15-inch. URL: http://support.apple.com/kb/SP644?viewlocale=it_IT.

[6] Y. Yang B. Klimmt. Introducing the enron corpus. *CEAS conference*, 2004.

[7] Herman Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *The Annals of Mathematical Statistics*, 23(4):493–507, 1952.

[8] A. Gionis S. Leonardi I. Bordino, D. Donato. Mining large networks with subgraph counting. *Data Mining, 2008. ICDM '08. Eighth IEEE International Conference on*, pages 737 – 742, December 2008.

[9] A. Dasgupta M. Mahoney J. Leskovec, K. Lang. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 6:29–123, 2009.

[10] C. Faloutsos J. Leskovec, J. Kleinberg. Graphs over time: Densification laws, shrinking diameters and possible explanations. *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2005.

[11] J. Leskovec J. Yang. Defining and evaluating network communities based on ground-truth. *ICDM*, 2012.

[12] P. Sanders K. Mehlhorn. *Algorithms and data structures: The basic toolbox*. Springer, 2009.

[13] S. Leonardi A. Marchetti-Spaccamela C.Sohler L.S Buriol, G.Frahling. Counting triangles in data streams. In *Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 253–262. ACM, 2006.

[14] A. Rényi P. Erdős. On random graphs. *Publicationes Mathematicae*, 6:290–297, 1959.