Aspetti algoritmici del Web

Paolo Boldi

Un documento storico

From: Tim Berners-Lee (timbl@info_.cern.ch)
Subject: WorldWideWeb: Summary
Newsgroups: alt.hypertext
View: (This is the only article in this thread) | Original Format
Date: 1991-08-06 13:37:40 PST

In article <6484@cernvax.cern.ch> I promised to post a short summary of the WorldWideWeb project. Mail me with any queries.

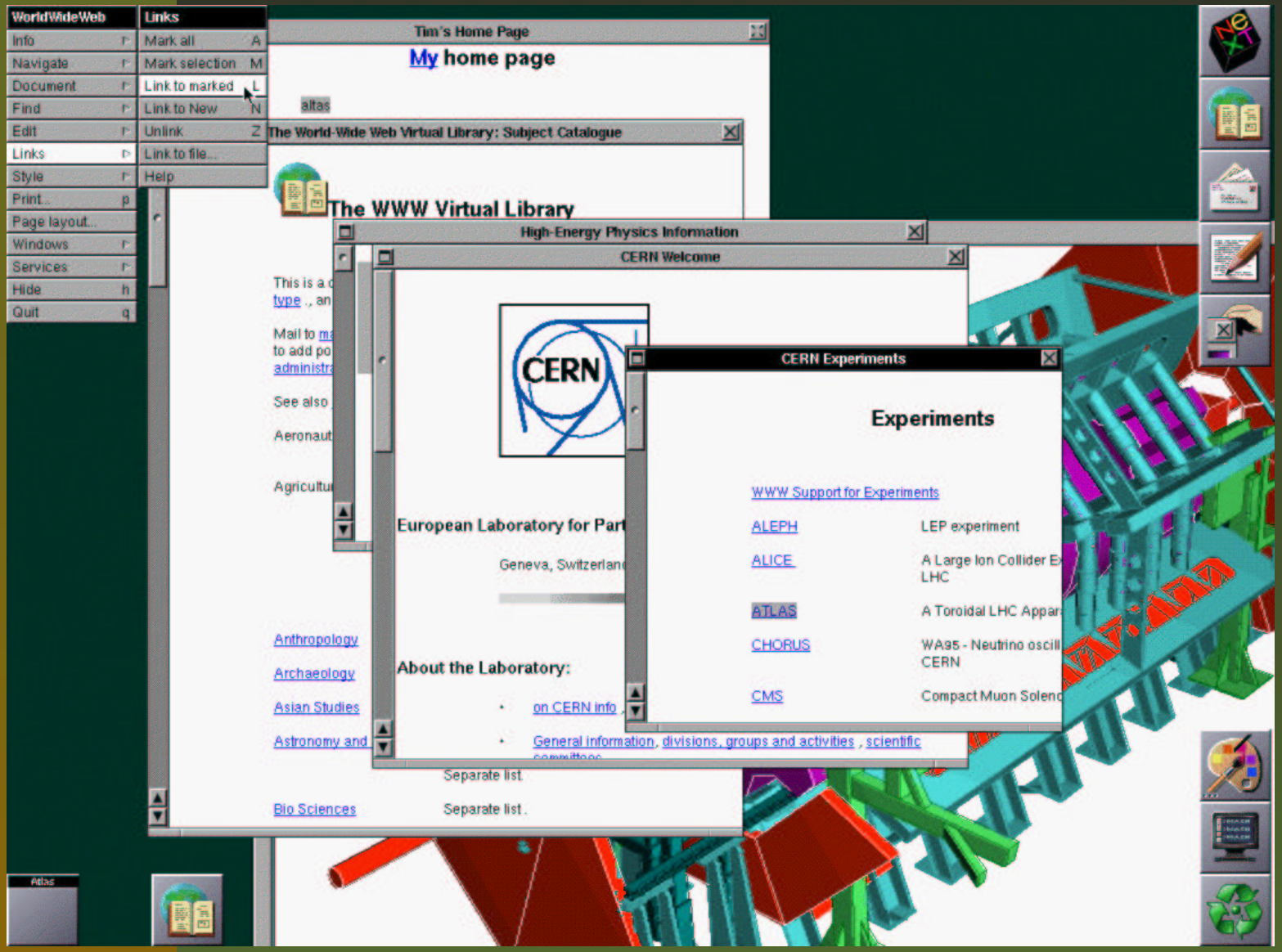
WorldWideWeb - Executive Summary

The WWW project merges the techniques of information retrieval and hypertext to make an easy but powerful global information system.

The project started with the philosophy that much academic information should be freely available to anyone. It aims to allow information sharing within internationally dispersed teams, and the dissemination of information by support groups.

...

Uno screenshot storico



Un po’ di storia: la “Rete”

- Metà degli anni ’60: nasce ARPANET

Un po’ di storia: la “Rete”

- Metà degli anni ’60: nasce ARPANET
- Fine anni ’60: standardizzazione dei protocolli

Un po’ di storia: la “Rete”

- Metà degli anni ’60: nasce ARPANET
- Fine anni ’60: standardizzazione dei protocolli
- Anni ’70: la rete è usata attivamente dalla comunità scientifica (posta elettronica, trasferimento di file)

Un po’ di storia: la “Rete”

- Metà degli anni ’60: nasce ARPANET
- Fine anni ’60: standardizzazione dei protocolli
- Anni ’70: la rete è usata attivamente dalla comunità scientifica (posta elettronica, trasferimento di file)
- 1980: formalizzazione del protocollo TCP/IP

Un po’ di storia: protocollo FTP

- FTP (File Transfer Protocol)

Un po’ di storia: protocollo FTP

- FTP (File Transfer Protocol)
  - standardizzato nel 1971

Un po’ di storia: protocollo FTP

- FTP (File Transfer Protocol)
  - standardizzato nel 1971
  - permette di copiare file da/verso server FTP

Un po’ di storia: protocollo FTP

- FTP (File Transfer Protocol)
  - standardizzato nel 1971
  - permette di copiare file da/verso server FTP
  - accesso mediante login/password (o “anonimo”)

Un po’ di storia: protocollo FTP

- FTP (File Transfer Protocol)
  - standardizzato nel 1971
  - permette di copiare file da/verso server FTP
  - accesso mediante login/password (o “anonimo”)
  - protocollo di elezione per il trasferimento di file negli anni ’70 e ’80

Un po’ di storia: protocollo FTP

- FTP (File Transfer Protocol)
  - standardizzato nel 1971
  - permette di copiare file da/verso server FTP
  - accesso mediante login/password (o “anonimo”)
  - protocollo di elezione per il trasferimento di file negli anni ’70 e ’80
  - alla fine degli anni ’80, circa metà del traffico di rete era dovuto a FTP

Un po’ di storia: protocollo FTP

- FTP (File Transfer Protocol)
  - standardizzato nel 1971
  - permette di copiare file da/verso server FTP
  - accesso mediante login/password (o “anonimo”)
  - protocollo di elezione per il trasferimento di file negli anni ’70 e ’80
  - alla fine degli anni ’80, circa metà del traffico di rete era dovuto a FTP
- FTP richiede che l’utente sappia *a priori* a quale server collegarsi per ottenere i dati desiderati

Un po’ di storia: protocolli di ricerca

- Gopher (University of Minnesota, primi anni ’90) consente di effettuare ricerche sulla base di parole chiave (o per argomento); la ricerca viene effettuata su un insieme di database registrati a partire da un database centrale

Un po’ di storia: protocolli di ricerca

- Gopher (University of Minnesota, primi anni ’90) consente di effettuare ricerche sulla base di parole chiave (o per argomento); la ricerca viene effettuata su un insieme di database registrati a partire da un database centrale
- WAIS (Wide Area Information Service, fine anni ’80) consentiva di effettuare ricerche su un database, e di ottenere in risposta un elenco di file remoti, ordinati per “rilevanza”

Un po’ di storia: protocolli di ricerca

- Gopher (University of Minnesota, primi anni ’90) consente di effettuare ricerche sulla base di parole chiave (o per argomento); la ricerca viene effettuata su un insieme di database registrati a partire da un database centrale
- WAIS (Wide Area Information Service, fine anni ’80) consentiva di effettuare ricerche su un database, e di ottenere in risposta un elenco di file remoti, ordinati per “rilevanza”
- Archie (1990) permetteva di cercare un file su un indice costruito a partire dai server FTP, usando il nome (o un pattern di ricerca)

La nascita del Web

- Ideato da Tim Berners-Lee nel 1989, mentre lavorava al CERN di Ginevra

La nascita del Web

- Ideato da Tim Berners-Lee nel 1989, mentre lavorava al CERN di Ginevra
- Ispirato all'idea di *ipertesto*

La nascita del Web

- Ideato da Tim Berners-Lee nel 1989, mentre lavorava al CERN di Ginevra
- Ispirato all'idea di *ipertesto*
- Berners-Lee è anche l'ideatore del concetto di URL, del protocollo HTTP, del linguaggio HTML, e del primo Web browser (chiamato WorldWideWeb, e realizzato per Next) e Web server

La nascita del Web

- Ideato da Tim Berners-Lee nel 1989, mentre lavorava al CERN di Ginevra
- Ispirato all'idea di *ipertesto*
- Berners-Lee è anche l'ideatore del concetto di URL, del protocollo HTTP, del linguaggio HTML, e del primo Web browser (chiamato WorldWideWeb, e realizzato per Next) e Web server

L’idea

- Il Web è una collezione di documenti

L’idea

- Il Web è una collezione di documenti
- I documenti possono essere in vari formati, ma di preferenza sono *ipertestuali* e contengono collegamenti (link) verso altri documenti

L’idea

- Il Web è una collezione di documenti
- I documenti possono essere in vari formati, ma di preferenza sono *ipertestuali* e contengono collegamenti (link) verso altri documenti
- I documenti sono resi disponibili attraverso un protocollo client/server

L’idea

- Il Web è una collezione di documenti
- I documenti possono essere in vari formati, ma di preferenza sono *ipertestuali* e contengono collegamenti (link) verso altri documenti
- I documenti sono resi disponibili attraverso un protocollo client/server

Ingredienti: URL + HTTP + HTML.

(1) URL

Ciascun documento è identificato da un URL (Universal Resource Locator).

(1) URL

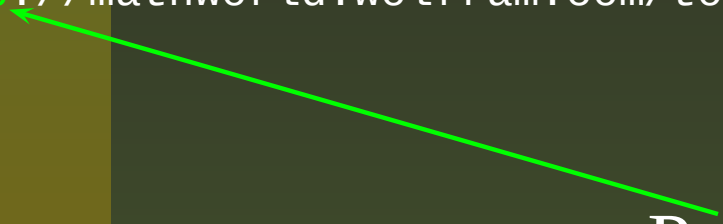
Ciascun documento è identificato da un URL (Universal Resource Locator).

`http://mathworld.wolfram.com/topics/AppliedMathematics.html`

(1) URL

Ciascun documento è identificato da un URL (Universal Resource Locator).

`http://mathworld.wolfram.com/topics/AppliedMathematics.html`



- HTTP: HyperText Transfer Protocol
- Altri protocolli: FTP (File Transfer Protocol), HTTPS (HTTP sicuro), Gopher, WAIS, News,. . .
- Alcuni client consentono di ometterlo (si assume HTTP)

(1) URL

Ciascun documento è identificato da un URL (Universal Resource Locator).

`http://mathworld.wolfram.com/topics/AppliedMathematics.html`

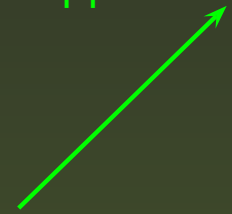
Nome dell'host

- Viene risolto in un indirizzo IP (140.177.205.23) mediante DNS (Domain Name System).
- Può succedere che hostname diversi corrispondano allo *stesso* indirizzo IP (virtual host), ad esempio `gongolo.usr.dsi.unimi.it` e `ioi.dsi.unimi.it` (159.149.147.201)

(1) URL

Ciascun documento è identificato da un URL (Universal Resource Locator).

`http://mathworld.wolfram.com/topics/AppliedMathematics.html`



Pathname

- Può essere pensato come il nome di un file
- È responsabilità del server stabilire come "rispondere" alla richiesta; spesso si tratta semplicemente di recuperare un file dal file system e servirlo al client (in questo caso, esiste un legame diretto fra Pathname e nome completo del file)

(2) HTTP

- È il protocollo di elezione per il Web (circa 89%).

(2) HTTP

- È il protocollo di elezione per il Web (circa 89%).
- Esiste in due versioni (HTTP/1.0 e HTTP/1.1)

(2) HTTP

- È il protocollo di elezione per il Web (circa 89%).
- Esiste in due versioni (HTTP/1.0 e HTTP/1.1)
- Esempio di richiesta:

(2) HTTP

- È il protocollo di elezione per il Web (circa 89%).
- Esiste in due versioni (HTTP/1.0 e HTTP/1.1)
- Esempio di richiesta:

```
GET http://mathworld.wolfram.com/topics/AppliedMathematics.html
HTTP/1.1
```

(2) HTTP (cont.)

Esempio di risposta:

```
HTTP/1.1 200 OK
Date: Fri, 17 May 2002 09:47:47 GMT
Server: Apache/1.3.20 (Unix)
Connection: close
Cache-Control: no-cache
Content-Length: 22085
Content-Type: text/html

<!DOCTYPE HTML PUBLIC -//W3C//DTD HTML 4.0 Transitional//EN»
<html>  ...
```

(2) HTTP (cont.)

Esempio di risposta:

```
HTTP/1.1 200 OK
Date: Fri, 17 May 2002 09:47:47 GMT
Server: Apache/1.3.20 (Unix)
Connection: close
Cache-Control: no-cache
Content-Length: 22085
Content-Type: text/html

<!DOCTYPE HTML PUBLIC -//W3C//DTD HTML 4.0 Transitional//EN»
<html>  ...
```

Status line

- Sto usando HTTP/1.1
- 200 OK: la richiesta è ok

(2) HTTP (cont.)

Esempio di risposta:

```
HTTP/1.1 200 OK
Date: Fri, 17 May 2002 09:47:47 GMT
Server: Apache/1.3.20 (Unix)
Connection: close
Cache-Control: no-cache
Content-Length: 22085
Content-Type: text/html

<!DOCTYPE HTML PUBLIC -//W3C//DTD HTML 4.0 Transitional//EN»
<html>  ...
```

Header `Date`

- La data in cui il server ha evaso la richiesta

(2) HTTP (cont.)

Esempio di risposta:

```
HTTP/1.1 200 OK
Date: Fri, 17 May 2002 09:47:47 GMT
Server: Apache/1.3.20 (Unix)
Connection: close
Cache-Control: no-cache
Content-Length: 22085
Content-Type: text/html

<!DOCTYPE HTML PUBLIC -//W3C//DTD HTML 4.0 Transitional//EN»
<html>  ...
```

Header `Server`

- Identifica il tipo di server

(2) HTTP (cont.)

Esempio di risposta:

```
HTTP/1.1 200 OK
Date: Fri, 17 May 2002 09:47:47 GMT
Server: Apache/1.3.20 (Unix)
Connection: close
Cache-Control: no-cache
Content-Length: 22085
Content-Type: text/html

<!DOCTYPE HTML PUBLIC -//W3C//DTD HTML 4.0 Transitional//EN»
<html>  ...
```

Header `Connection`

- È tipico del protocollo HTTP/1.1
- Indica che la risposta è costituita solo da questo chunk, e che il server chiuderà la comunicazione subito dopo averlo trasmesso

(2) HTTP (cont.)

Esempio di risposta:

```
HTTP/1.1 200 OK
Date: Fri, 17 May 2002 09:47:47 GMT
Server: Apache/1.3.20 (Unix)
Connection: close
Cache-Control: no-cache
Content-Length: 22085
Content-Type: text/html

<!DOCTYPE HTML PUBLIC -//W3C//DTD HTML 4.0 Transitional//EN»
<html>  ...
```

Header `Cache-control`

- Prega il browser di non cachare il contenuto della pagina (tipico di pagine che si modificano spesso)

(2) HTTP (cont.)

Esempio di risposta:

```
HTTP/1.1 200 OK
Date: Fri, 17 May 2002 09:47:47 GMT
Server: Apache/1.3.20 (Unix)
Connection: close
Cache-Control: no-cache
Content-Length: 22085
Content-Type: text/html

<!DOCTYPE HTML PUBLIC -//W3C//DTD HTML 4.0 Transitional//EN»
<html>  ...
```

Header `Content-length`

- Lunghezza in byte del contenuto della pagina

(2) HTTP (cont.)

Esempio di risposta:

```
HTTP/1.1 200 OK
Date: Fri, 17 May 2002 09:47:47 GMT
Server: Apache/1.3.20 (Unix)
Connection: close
Cache-Control: no-cache
Content-Length: 22085
Content-Type: text/html

<!DOCTYPE HTML PUBLIC -//W3C//DTD HTML 4.0 Transitional//EN»
<html>  ...
```

Header `Content-type`

- È il tipo MIME del contenuto
- In questo caso è `text/html` (che vuol dire che la pagina contiene HTML)

(2) HTTP (cont.)

Esempio di risposta:

```
HTTP/1.1 200 OK
Date: Fri, 17 May 2002 09:47:47 GMT
Server: Apache/1.3.20 (Unix)
Connection: close
Cache-Control: no-cache
Content-Length: 22085
Content-Type: text/html

<!DOCTYPE HTML PUBLIC -//W3C//DTD HTML 4.0 Transitional//EN»
<html>  ...
```

Il contenuto

Cosa fa il client?

A fronte di un URL, il client (browser; p.es. IE, Netscape, Mozilla, Amaya. . .)

- Risolve l’hostname in un indirizzo IP (via DNS)

Cosa fa il client?

A fronte di un URL, il client (browser; p.es. IE, Netscape, Mozilla, Amaya. . .)

- Risolve l’hostname in un indirizzo IP (via DNS)
- Contatta il server

Cosa fa il client?

A fronte di un URL, il client (browser; p.es. IE, Netscape, Mozilla, Amaya. . .)

- Risolve l'hostname in un indirizzo IP (via DNS)
- Contatta il server
- Gli invia la richiesta (usando il protocollo specificato, p.es. l'HTTP)

Cosa fa il client?

A fronte di un URL, il client (browser; p.es. IE, Netscape, Mozilla, Amaya. . .)

- Risolve l'hostname in un indirizzo IP (via DNS)
- Contatta il server
- Gli invia la richiesta (usando il protocollo specificato, p.es. l'HTTP)
- Attende una risposta

Cosa fa il client?

A fronte di un URL, il client (browser; p.es. IE, Netscape, Mozilla, Amaya. . .)

- Risolve l'hostname in un indirizzo IP (via DNS)
- Contatta il server
- Gli invia la richiesta (usando il protocollo specificato, p.es. l'HTTP)
- Attende una risposta
- Visualizza il contenuto del documento

Tipi di documenti

Il tipo del documento servito è determinato dallo header `Content-type`, e può essere:

- puro testo

Tipi di documenti

Il tipo del documento servito è determinato dallo header `Content-type`, e può essere:

- puro testo
- testo formattato (HTML, PostScript, PDF, . . . , PPT, Word [sic!], . . .)

Tipi di documenti

Il tipo del documento servito è determinato dallo header `Content-type`, e può essere:

- puro testo
- testo formattato (HTML, PostScript, PDF, . . . , PPT, Word [sic!], . . .)
- multimediale (immagini, suoni, filmati. . .)

Tipi di documenti

Il tipo del documento servito è determinato dallo header `Content-type`, e può essere:

- puro testo
- testo formattato (HTML, PostScript, PDF, . . . , PPT, Word [sic!], . . .)
- multimediale (immagini, suoni, filmati. . .)

Il browser può essere in grado da solo di visualizzare completamente il documento, oppure può farlo usando appositi plugin o *helper applications*.

Tipi di documenti

Il tipo del documento servito è determinato dallo header `Content-type`, e può essere:

- puro testo
- testo formattato (HTML, PostScript, PDF, . . . , PPT, Word [sic!], . . .)
- multimediale (immagini, suoni, filmati. . .)

Il browser può essere in grado da solo di visualizzare completamente il documento, oppure può farlo usando appositi plugin o *helper applications*. Considereremo essenzialmente solo documenti testuali. . .

(3) HTML

- L’HTML (HyperText Markup Language) è un linguaggio per la formattazione di documenti ipertestuali

(3) HTML

- L’HTML (HyperText Markup Language) è un linguaggio per la formattazione di documenti ipertestuali
- È il tipo di documento più comune

(3) HTML

- L’HTML (HyperText Markup Language) è un linguaggio per la formattazione di documenti ipertestuali
- È il tipo di documento più comune
- Un documento HTML può contenere link ipertestuali ad altri URL

Il grafo del Web

Potete pensare all'insieme dei documenti presenti sul Web come a un grafo, in cui:

- i *nodi* sono gli URL;

Il grafo del Web

Potete pensare all'insieme dei documenti presenti sul Web come a un grafo, in cui:

- i *nodi* sono gli URL;
- c'è un *arco* fra il nodo x e il nodo y sse la pagina che corrisponde all'URL x contiene un link verso l'URL y.

Il grafo del Web

Potete pensare all'insieme dei documenti presenti sul Web come a un grafo, in cui:

- i *nodi* sono gli URL;
- c'è un *arco* fra il nodo x e il nodo y sse la pagina che corrisponde all'URL x contiene un link verso l'URL y.

Questo grafo è chiamato *grafo del Web*. Ovviamente, si tratta di un grafo altamente dinamico, che cambia in continuazione.

Dimensioni del Web

Difficili da valutare; comunque, il grafo è *enorme*.

- numero di nodi (=documenti): 2/4 miliardi (escludendo le pagine non accessibili)

Dimensioni del Web

Difficili da valutare; comunque, il grafo è *enorme*.

- numero di nodi (=documenti): 2/4 miliardi (escludendo le pagine non accessibili)
- numero di archi: 60/100 miliardi

Dimensioni del Web

Difficili da valutare; comunque, il grafo è *enorme*.

- numero di nodi (=documenti): 2/4 miliardi (escludendo le pagine non accessibili)
- numero di archi: 60/100 miliardi
- numero di host: 100/200 milioni

Dimensioni del Web

Difficili da valutare; comunque, il grafo è *enorme*.

- numero di nodi (=documenti): 2/4 miliardi (escludendo le pagine non accessibili)
- numero di archi: 60/100 miliardi
- numero di host: 100/200 milioni
- numero di utenti: 500/800 milioni

Richiami: componenti connesse

Dato un grafo orientato $G = (V, E)$, definiamo una relazione $\leftrightarrow$ fra i nodi, ponendo $x \leftrightarrow y$ sse esiste un cammino da x a y *e* un cammino da y a x.

Richiami: componenti connesse

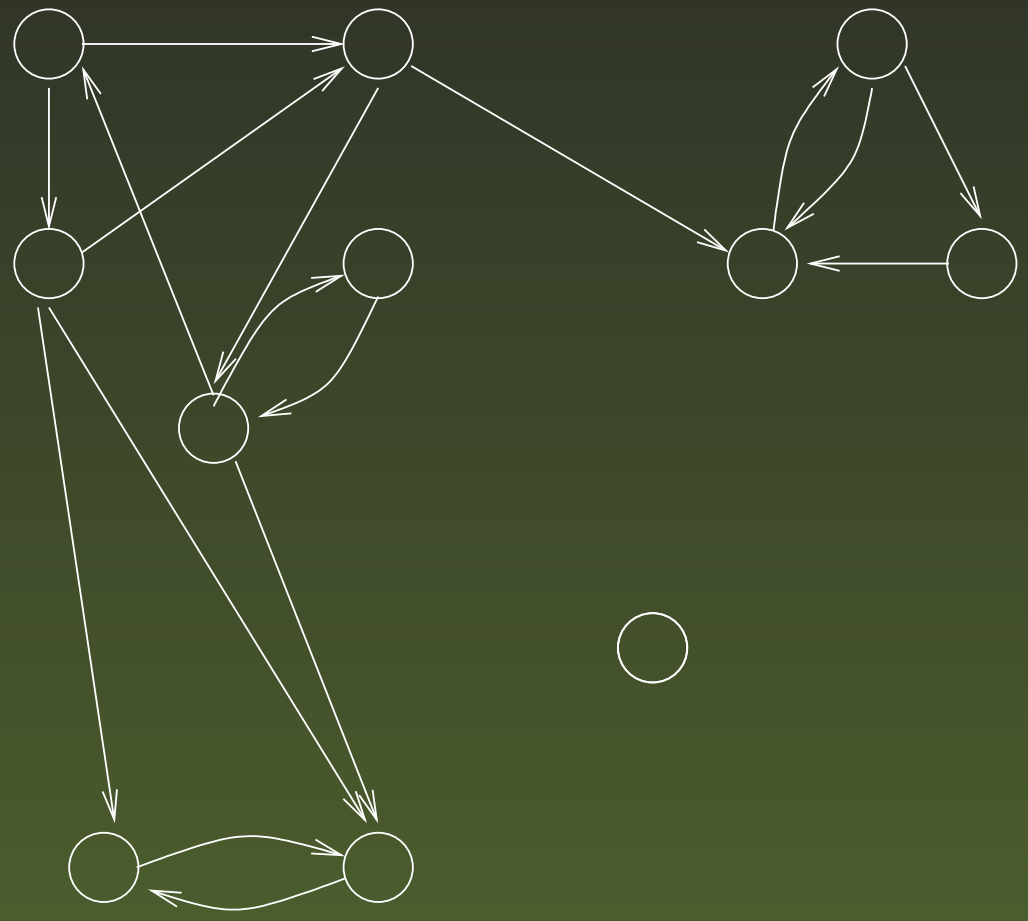
Dato un grafo orientato $G = (V, E)$, definiamo una relazione $\leftrightarrow$ fra i nodi, ponendo $x \leftrightarrow y$ sse esiste un cammino da x a y *e* un cammino da y a x.
La relazione $\leftrightarrow$ è una relazione di equivalenza, le cui classi sono dette *componenti (fortemente) connesse* del grafo.

Richiami: componenti connesse

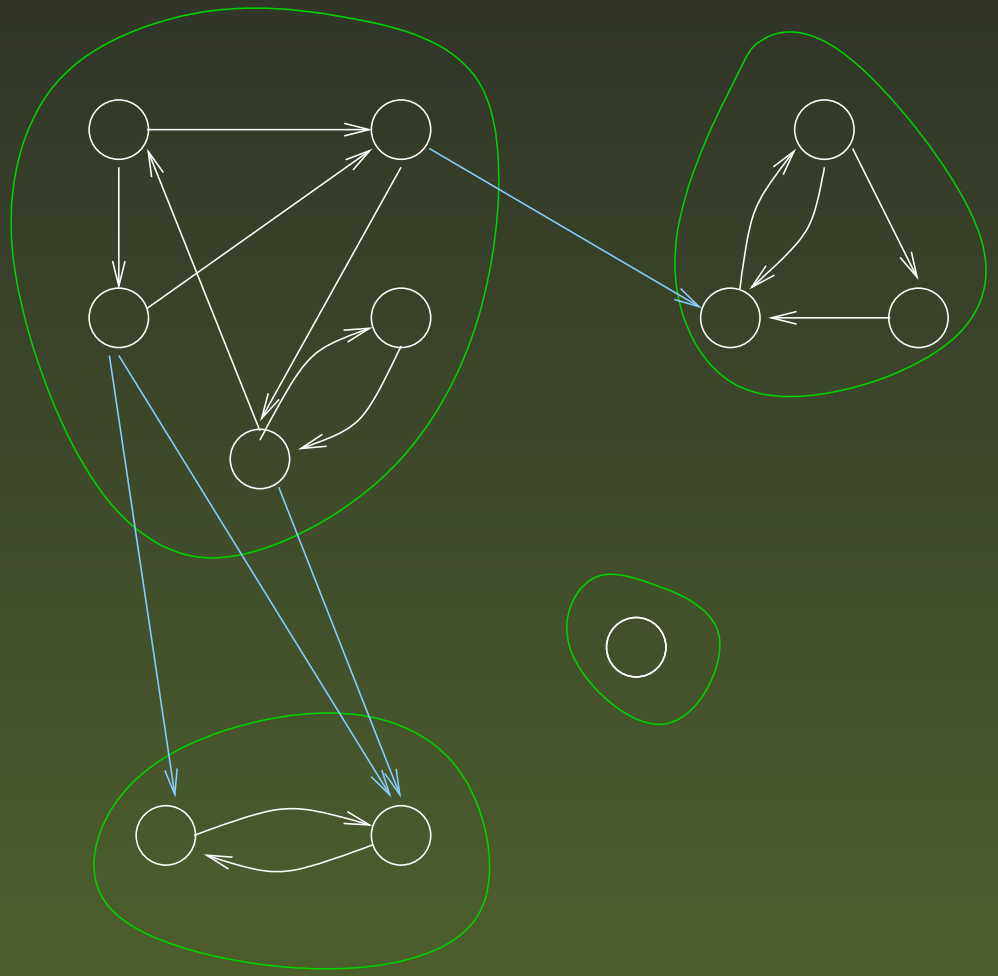
Dato un grafo orientato $G = (V, E)$, definiamo una relazione $\leftrightarrow$ fra i nodi, ponendo $x \leftrightarrow y$ sse esiste un cammino da x a y *e* un cammino da y a x.
La relazione $\leftrightarrow$ è una relazione di equivalenza, le cui classi sono dette *componenti (fortemente) connesse* del grafo.

È possibile costruire il *grafo ridotto* G^*, che ha come nodi le componenti connesse, e ha un arco fra la componente C_1 e la componente C_2 sse esiste un arco che va da un nodo di C_1 a un nodo di C_2.

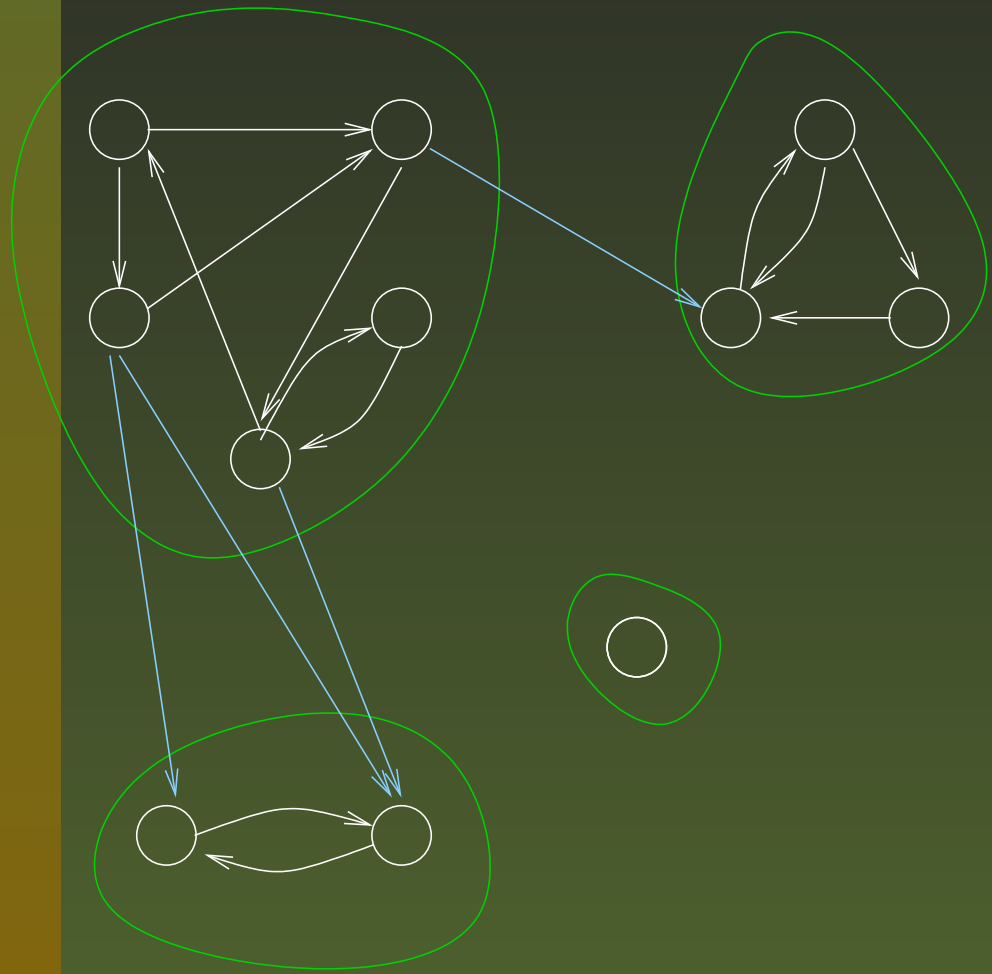
Componenti connesse: un esempio



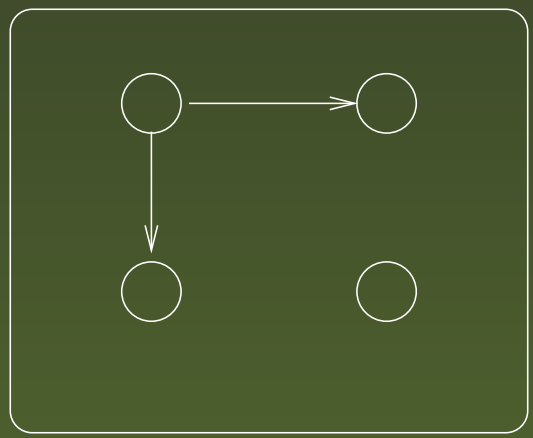
Componenti connesse: un esempio



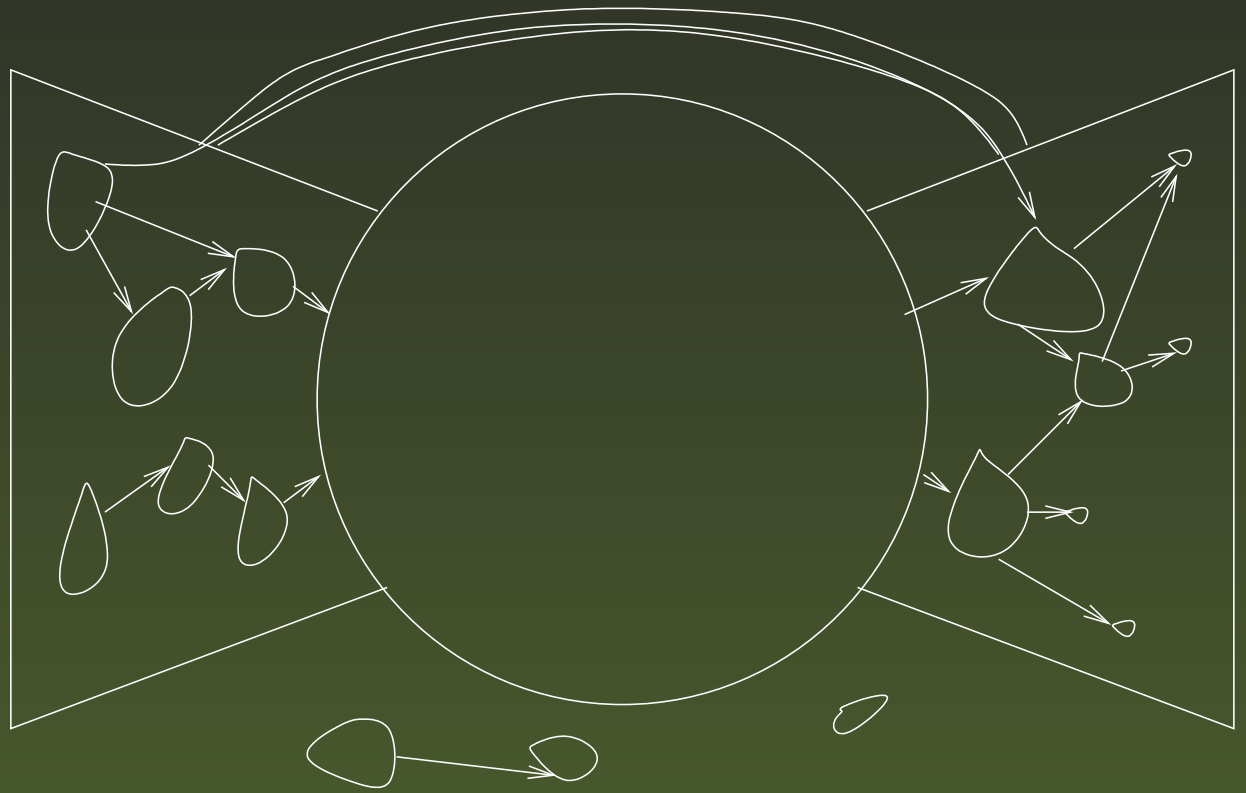
Componenti connesse: un esempio



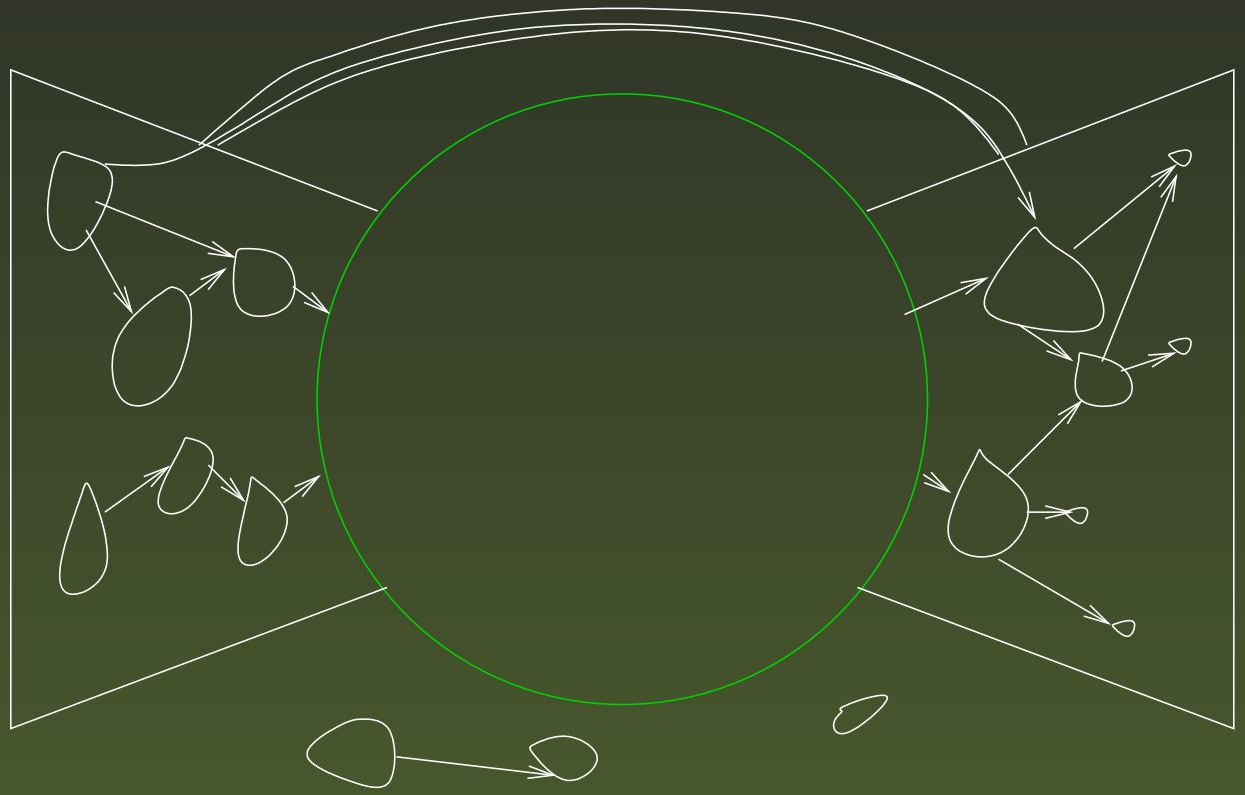
GRAFO RIDOTTO



La struttura “a cravattino” del web



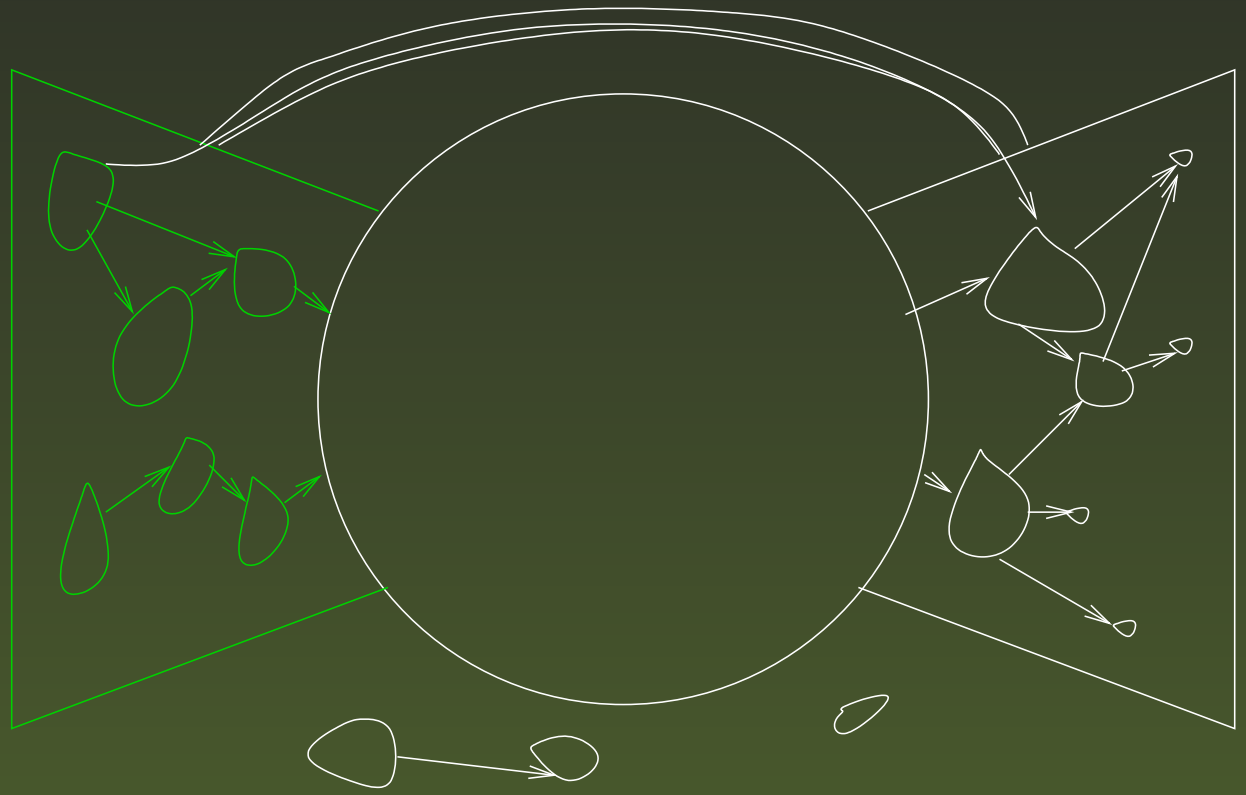
La struttura “a cravattino” del web



Componente gigante

- Comprende circa il 30% delle pagine
- Stime del diametro: orientato=20/30; non orientato=10/17
- “Com’è piccolo il mondo!” (small-world theory)

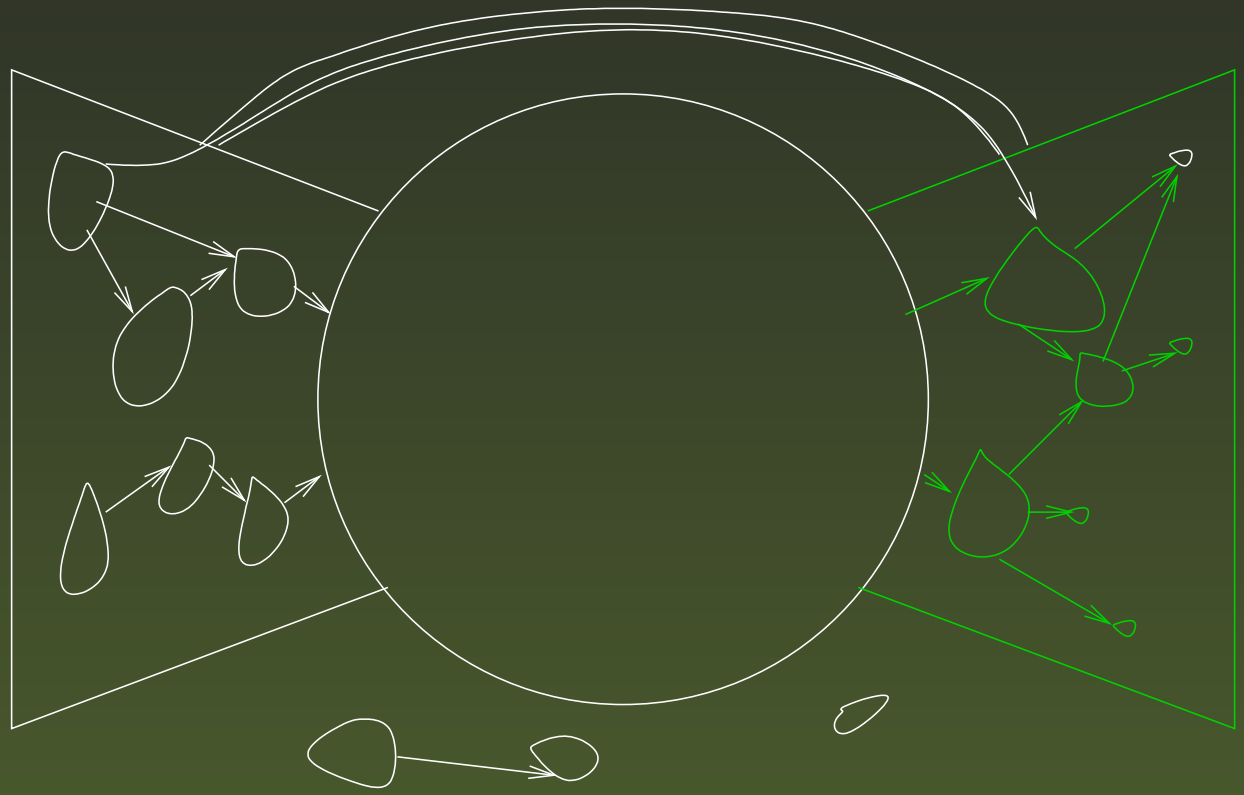
La struttura “a cravattino” del web



Componenti “sorgente” (circa 24%)

- Puntano (direttamente o indirettamente) verso la componente gigante ma…
- …non sono raggiungibili dalla componente gigante
- Sono le pagine “reiette”

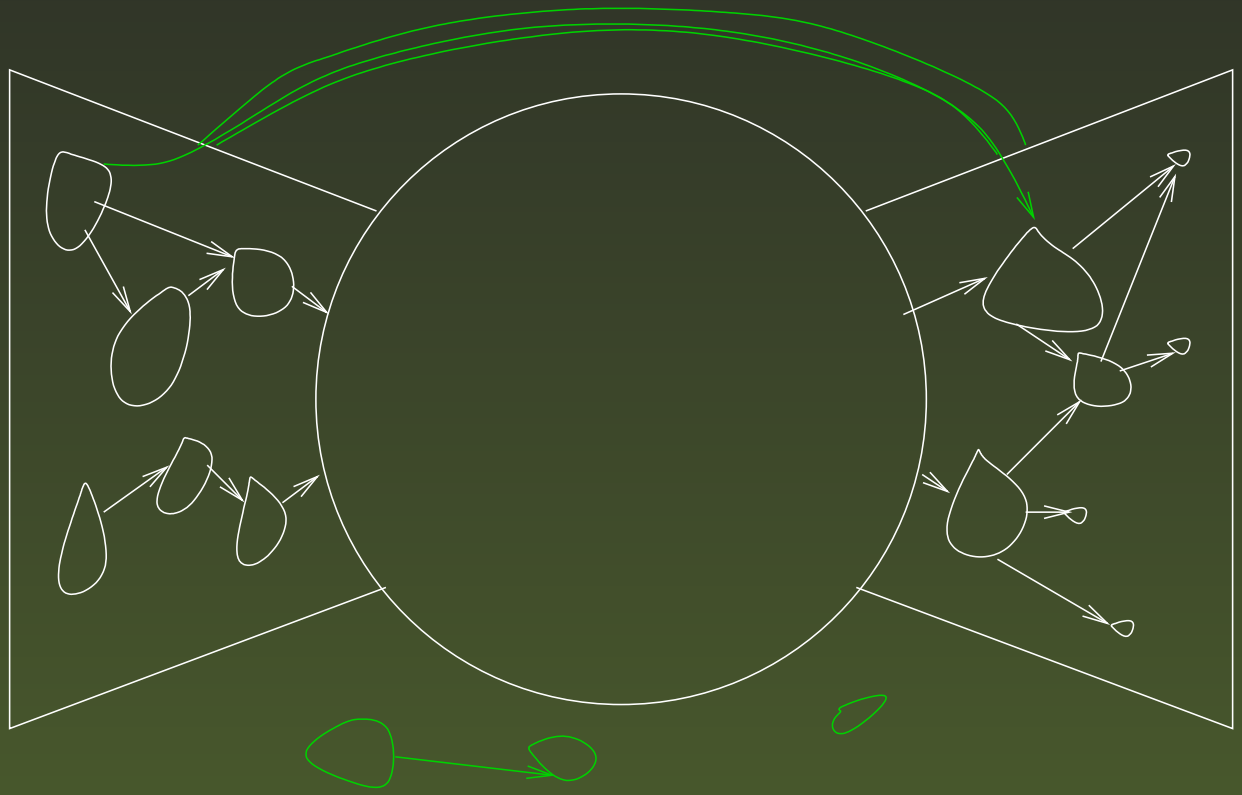
La struttura “a cravattino” del web



Componenti “pozzo” (circa 24%)

- Sono raggiungibili dalla componente gigante ma…
- …da esse non si può tornare indietro
- In questa categoria rientra la maggior parte dei *documenti* (senza link)

La struttura “a cravattino” del web



Componenti “isolate” e tentacoli

- Non sono raggiungibili dalla componente gigante
- Da esse non si raggiunge la componente gigante
- Ci sono collegamenti fra sorgenti e pozzi che non passano per la componente gigante (*tentacoli*)

Come cercare informazioni?

La ricerca di informazioni è diventata sempre più difficile, per vari motivi:

- dimensioni (*too much information* . . .)

Come cercare informazioni?

La ricerca di informazioni è diventata sempre più difficile, per vari motivi:

- dimensioni (*too much information* . . .)
- mancanza di semantica (tentativi di realizzare il *Web semantico*) e struttura

Come cercare informazioni?

La ricerca di informazioni è diventata sempre più difficile, per vari motivi:

- dimensioni (*too much information* . . .)
- mancanza di semantica (tentativi di realizzare il *Web semantico*) e struttura
- qualità di informazione estremamente eterogenea

Come cercare informazioni?

La ricerca di informazioni è diventata sempre più difficile, per vari motivi:

- dimensioni (*too much information* . . .)
- mancanza di semantica (tentativi di realizzare il *Web semantico*) e struttura
- qualità di informazione estremamente eterogenea
- i documenti sono soggetti a rapida modifica

Come cercare informazioni?

La ricerca di informazioni è diventata sempre più difficile, per vari motivi:

- dimensioni (*too much information* . . .)
- mancanza di semantica (tentativi di realizzare il *Web semantico*) e struttura
- qualità di informazione estremamente eterogenea
- i documenti sono soggetti a rapida modifica

. . . circa l'80% degli utenti utilizza abitualmente i motori di ricerca

Com’è fatto un motore di ricerca?

Tre attività logicamente distinte:

- raccolta di dati

Com’è fatto un motore di ricerca?

Tre attività logicamente distinte:

- raccolta di dati
- elaborazione dei dati raccolti

Com’è fatto un motore di ricerca?

Tre attività logicamente distinte:

- raccolta di dati
- elaborazione dei dati raccolti
- risposta alle query degli utenti

Raccolta dati

- Si tratta di recuperare il contenuto delle pagine Web (di solito, limitandosi a quelle testuali)

Raccolta dati

- Si tratta di recuperare il contenuto delle pagine Web (di solito, limitandosi a quelle testuali)
- Viene svolta da un'apposita componente, detta *spider*

Raccolta dati

- Si tratta di recuperare il contenuto delle pagine Web (di solito, limitandosi a quelle testuali)
- Viene svolta da un'apposita componente, detta *spider*
- Problemi da affrontare (fra gli altri)

Raccolta dati

- Si tratta di recuperare il contenuto delle pagine Web (di solito, limitandosi a quelle testuali)
- Viene svolta da un'apposita componente, detta *spider*
- Problemi da affrontare (fra gli altri)
  - Quantità di dati; quantità di banda

Raccolta dati

- Si tratta di recuperare il contenuto delle pagine Web (di solito, limitandosi a quelle testuali)
- Viene svolta da un'apposita componente, detta *spider*
- Problemi da affrontare (fra gli altri)
  - Quantità di dati; quantità di banda
  - Frequenza di aggiornamento

Raccolta dati

- Si tratta di recuperare il contenuto delle pagine Web (di solito, limitandosi a quelle testuali)
- Viene svolta da un'apposita componente, detta *spider*
- Problemi da affrontare (fra gli altri)
  - Quantità di dati; quantità di banda
  - Frequenza di aggiornamento
  - Presenza di materiale nascosto (cfr. cravattino)

Raccolta dati

- Si tratta di recuperare il contenuto delle pagine Web (di solito, limitandosi a quelle testuali)
- Viene svolta da un'apposita componente, detta *spider*
- Problemi da affrontare (fra gli altri)
  - Quantità di dati; quantità di banda
  - Frequenza di aggiornamento
  - Presenza di materiale nascosto (cfr. cravattino)
  - Standard non rispettati

Raccolta dati

- Si tratta di recuperare il contenuto delle pagine Web (di solito, limitandosi a quelle testuali)
- Viene svolta da un'apposita componente, detta *spider*
- Problemi da affrontare (fra gli altri)
  - Quantità di dati; quantità di banda
  - Frequenza di aggiornamento
  - Presenza di materiale nascosto (cfr. cravattino)
  - Standard non rispettati
  - Spider traps!

Elaborazione dati

Obiettivi:

- Estrarre informazioni (parsing)

Elaborazione dati

Obiettivi:

- Estrarre informazioni (parsing)
- Rilevare la presenza di duplicati (o quasi duplicati) dovuti al mirroring

Elaborazione dati

Obiettivi:

- Estrarre informazioni (parsing)
- Rilevare la presenza di duplicati (o quasi duplicati) dovuti al mirroring
- Rilevare la presenza di spamming

Elaborazione dati

Obiettivi:

- Estrarre informazioni (parsing)
- Rilevare la presenza di duplicati (o quasi duplicati) dovuti al mirroring
- Rilevare la presenza di spamming
- Indicizzare i dati

Elaborazione dati

Obiettivi:

- Estrarre informazioni (parsing)
- Rilevare la presenza di duplicati (o quasi duplicati) dovuti al mirroring
- Rilevare la presenza di spamming
- Indicizzare i dati
- Eventualmente: calcolare informazioni necessarie per il ranking

Risposta alle query

Diversi modi di rispondere alle query:

- Ricerche testuali sofisticate (*ricetta AND (pasta NEAR pesto)*)

Risposta alle query

Diversi modi di rispondere alle query:

- Ricerche testuali sofisticate (*ricetta AND (pasta NEAR pesto)*)
- Uso di suggerimenti ontologici

Risposta alle query

Diversi modi di rispondere alle query:

- Ricerche testuali sofisticate (*ricetta AND (pasta NEAR pesto)*)
- Uso di suggerimenti ontologici
- Riconoscimento linguistico

Risposta alle query

Diversi modi di rispondere alle query:

- Ricerche testuali sofisticate (*ricetta AND (pasta NEAR pesto)*)
- Uso di suggerimenti ontologici
- Riconoscimento linguistico
- Analisi del profilo utente (logging di query, bookmarks ecc.)

Risposta alle query

Diversi modi di rispondere alle query:

- Ricerche testuali sofisticate (*ricetta AND (pasta NEAR pesto)*)
- Uso di suggerimenti ontologici
- Riconoscimento linguistico
- Analisi del profilo utente (logging di query, bookmarks ecc.)
- Sistemi di categorizzazione automatica

Raccolta

Com’è fatto uno spider

Uno spider deve raccogliere pagine, essenzialmente effettuando una visita del grafo del web.

Com’è fatto uno spider

Uno spider deve raccogliere pagine, essenzialmente effettuando una visita del grafo del web.
Inizia la raccolta a partire da una (o più) pagine, dette *seme* di raccolta.

Com’è fatto uno spider

Uno spider deve raccogliere pagine, essenzialmente effettuando una visita del grafo del web.
Inizia la raccolta a partire da una (o più) pagine, dette *seme* di raccolta.

Il problema della *coverage*: anche i migliori motori di ricerca coprono solo il 16/20% della rete (in parte, a causa del fenomeno della *dark net*).

Spidering in profondità

```
function visit(P) {
   if (P nuova) {
      scarica e memorizza P
      for (P′: URL che compare in P)
         visit(P′)
   }
}

visit(P_seme)
```

Spidering in ampiezza

```
function visit() {
    Q=coda vuota
    accoda(Q,P_seme)
    while (Q non vuota) {
        estrai una pagina P da Q
        if (P nuova) {
            scarica e memorizza P
            for (P′: URL che compare in P)
                accoda(Q,P)
        }
    }
}
```

Vantaggi/svantaggi

- Visita in profondità: richiede meno memoria (nessuna coda)

Vantaggi/svantaggi

- Visita in profondità: richiede meno memoria (nessuna coda)
- Visita in ampiezza: recupera prima le pagine di alta qualità; non “insiste” su uno host; è meno suscettibile alle trappole

Vantaggi/svantaggi

- Visita in profondità: richiede meno memoria (nessuna coda)
- Visita in ampiezza: recupera prima le pagine di alta qualità; non “insiste” su uno host; è meno suscettibile alle trappole

Problemi:

- parsing robusto (preparsing?)
- rispettare `robots.txt`
- evitare di “bombardare” uno host

Parallelizzare la visita! (problemi di sincronizzazione. . .)

Snapshot e rinfresco

Quanto tempo occorre per ottenere uno snapshot del web? Dati ufficiosi: Google ha qualche centinaio di macchine di spidering, e un tempo di snapshot di circa 20/30 giorni.

Snapshot e rinfresco

Quanto tempo occorre per ottenere uno snapshot del web? Dati ufficiosi: Google ha qualche centinaio di macchine di spidering, e un tempo di snapshot di circa 20/30 giorni.
Uno studio del 1998 dava una percentuale fra di 1.5/6% di link “morti” nei maggiori motori di ricerca.

Snapshot e rinfresco

Quanto tempo occorre per ottenere uno snapshot del web? Dati ufficiosi: Google ha qualche centinaio di macchine di spidering, e un tempo di snapshot di circa 20/30 giorni.
Uno studio del 1998 dava una percentuale fra di 1.5/6% di link “morti” nei maggiori motori di ricerca.

Ovviamente, occorre mantenere i dati ottenuti aggiornati, rinfrescando periodicamente la visita. Quando scade una pagina? (Uso di header? analisi statistiche? hot pages?) Come organizzare le pagine per il rinfresco? (Code di priorità?)

Elaborazione

Il problema

Nella fase di elaborazione occorre indicizzare i documenti recuperati. L'indicizzazione deve consentire in modo efficiente di rispondere alle query.

Il problema

Nella fase di elaborazione occorre indicizzare i documenti recuperati. L'indicizzazione deve consentire in modo efficiente di rispondere alle query.

In particolare: l'indicizzazione deve permettere il ranking dei documenti!

Il ranking

Dato un insieme $\mathcal{P}$ di pagine e una query Q, definire una funzione $r_Q : \mathcal{P} \rightarrow \mathbf{R}$ che associ, ad ogni pagina, un numero reale (rank), che indica il grado di *rilevanza* di quella pagina a fronte di quella query.

Il ranking

Dato un insieme $\mathcal{P}$ di pagine e una query Q, definire una funzione $r_Q : \mathcal{P} \rightarrow \mathbf{R}$ che associ, ad ogni pagina, un numero reale (rank), che indica il grado di *rilevanza* di quella pagina a fronte di quella query.

Tecniche di ranking basate su:

- analisi del contenuto testuale (Altavista)
- analisi della struttura dei link (Google)

Latent Semantic Indexing

È una tecnica di ranking basata sul contenuto testuale. È adottata da Altavista.

Latent Semantic Indexing

È una tecnica di ranking basata sul contenuto testuale. È adottata da Altavista.
Sia t il numero di termini considerati (presi, p.es., da un dizionario, o tutte le parole incontrate nelle pagine raccolte).

Ad ogni pagina P è associato un vettore di t elementi $\vec{d}_P$ dove

$$(d_P)_j = \text{n. occorrenze del termine } j \text{ in } P$$

Latent Semantic Indexing (cont.)

Ad ogni query Q si associa un analogo vettore $\vec{d}_Q$, che ha un 1 in corrispondenza dei termini che compaiono nella query, e uno 0 altrimenti.
Si tratta di determinare l'affinità fra Q e D:

$$\cos \widehat{d_D d_Q} = \frac{\vec{d}_D \times \vec{d}_Q}{\|d_D\| \|d_Q\|}$$

Latent Semantic Indexing (cont.)

Assunzione ingenua del LSI:

> una pagina è autorevole su un argomento se i termini relativi a quell'argomento vi compaiono spesso

Non è vera (p.es. FIAT) ed è suscettibile allo spamming.

Latent Semantic Indexing (cont.)

Assunzione ingenua del LSI:

> una pagina è autorevole su un argomento se i termini relativi a quell'argomento vi compaiono spesso

Non è vera (p.es. FIAT) ed è suscettibile allo spamming. Si possono migliorare le prestazioni estendendo il testo della pagina con i testi delle ancore che puntano alla pagina (ed, eventualmente, il loro contesto).

Latent Semantic Indexing (cont.)

Assunzione ingenua del LSI:

> una pagina è autorevole su un argomento se i termini relativi a quell'argomento vi compaiono spesso

Non è vera (p.es. FIAT) ed è suscettibile allo spamming. Si possono migliorare le prestazioni estendendo il testo della pagina con i testi delle ancore che puntano alla pagina (ed, eventualmente, il loro contesto).

Inoltre, il LSI funziona bene solo su query multiple, ma la maggior parte delle query sono semplici (una, due, al massimo tre parole).

PageRank: caratteristiche

PageRank è un algoritmo di ranking con le seguenti caratteristiche:

- assegna a ciascuna pagina i un rank R_i in modo *statico*, cioè indipendente dalla query: data una query Q, si determineranno le pagine che soddisfano la query, e queste pagine verranno ordinate in base al loro rank;

PageRank: caratteristiche

PageRank è un algoritmo di ranking con le seguenti caratteristiche:

- assegna a ciascuna pagina i un rank R_i in modo *statico*, cioè indipendente dalla query: data una query Q, si determineranno le pagine che soddisfano la query, e queste pagine verranno ordinate in base al loro rank;
- determina l'importanza di una pagina *esclusivamente* sulla base dei link, e non del contenuto testuale: si basa sull'idea che il contenuto *non è autodescrittivo*, e che il conferimento di importanza di una pagina è un processo *esogeno*.

PageRank: caratteristiche

PageRank è un algoritmo di ranking con le seguenti caratteristiche:

- assegna a ciascuna pagina i un rank R_i in modo *statico*, cioè indipendente dalla query: data una query Q, si determineranno le pagine che soddisfano la query, e queste pagine verranno ordinate in base al loro rank;
- determina l'importanza di una pagina *esclusivamente* sulla base dei link, e non del contenuto testuale: si basa sull'idea che il contenuto *non è autodescrittivo*, e che il conferimento di importanza di una pagina è un processo *esogeno*.

È alla base dell'algoritmo di ranking usato da Google.

PageRank: l’idea

Una pagina è tanto più importante quanto più numerose sono le pagine che la puntano.

PageRank: l'idea

Una pagina è tanto più importante quanto più numerose sono le pagine che la puntano.
Se R_i indica l'importanza (rango) di una pagina, essa distribuisce la propria importanza in modo uniforme alle pagine che punta.

$$R_i = \sum_{j \to i} \frac{R_j}{N_j}$$

dove $j \to i$ indica la presenza di un link da j a i, e N_j è il numero di link contenuti nella pagina j.

PageRank: l'idea

Una pagina è tanto più importante quanto più numerose sono le pagine che la puntano.
Se R_i indica l'importanza (rango) di una pagina, essa distribuisce la propria importanza in modo uniforme alle pagine che punta.

$$R_i = \sum_{j \to i} \frac{R_j}{N_j}$$

dove $j \to i$ indica la presenza di un link da j a i, e N_j è il numero di link contenuti nella pagina j.

Esiste una (unica) soluzione all'equazione di ricorrenza?

Solo se il grafo è fortemente connesso!

PageRank: la formula

Per garantire che il grafo sia fortemente connesso, si introduce un fattore che corrisponde a introdurre dei "link random" al grafo:

$$R_i = (1-\alpha)\sum_{j \to i} \frac{R_j}{N_j} + \alpha \frac{1}{N}$$

dove N è il numero di pagine.

PageRank: la formula

Per garantire che il grafo sia fortemente connesso, si introduce un fattore che corrisponde a introdurre dei "link random" al grafo:

$$R_i = (1-\alpha)\sum_{j \to i} \frac{R_j}{N_j} + \alpha \frac{1}{N}$$

dove N è il numero di pagine.
Il rango della pagina i è determinato in parte (cioè, per una frazione $1-\alpha$) dalle pagine che puntano i, e in parte (frazione α) è acquisito "gratuitamente" (come per effetto della presenza di archi da tutte le pagine alla pagina i).

PageRank: la formula

Per garantire che il grafo sia fortemente connesso, si introduce un fattore che corrisponde a introdurre dei "link random" al grafo:

$$R_i = (1-\alpha)\sum_{j \to i} \frac{R_j}{N_j} + \alpha \frac{1}{N}$$

dove N è il numero di pagine.
Il rango della pagina i è determinato in parte (cioè, per una frazione $1-\alpha$) dalle pagine che puntano i, e in parte (frazione α) è acquisito "gratuitamente" (come per effetto della presenza di archi da tutte le pagine alla pagina i).

$\alpha \in [0, 1]$: di solito $\alpha \approx 0.15$ (fattore di spargimento).

PageRank: il problema dei pozzi

La formula di PageRank si può calcolare iterativamente, a partire da

$$\vec{R} = (1/N, 1/N, \ldots, 1/N)$$

(tutte le pagine hanno la stessa importanza) e applicando la formula, che “redistribuisce” l’importanza delle pagine.

PageRank: il problema dei pozzi

La formula di PageRank si può calcolare iterativamente, a partire da

$$\vec{R} = (1/N, 1/N, \ldots, 1/N)$$

(tutte le pagine hanno la stessa importanza) e applicando la formula, che “redistribuisce” l’importanza delle pagine.

Ad ogni passo, viene mantenuta somma 1?

PageRank: il problema dei pozzi

$$\sum_i R_i^{t+1} = \sum_i \left[(1-\alpha) \sum_{j \to i} \frac{R_j^t}{N_j} + \alpha \frac{1}{N} \right] = (1-\alpha) \sum_i \sum_{j \to i} \frac{R_j^t}{N_j} + \alpha$$

PageRank: il problema dei pozzi

$$\sum_i R_i^{t+1} = \sum_i \left[(1-\alpha) \sum_{j \to i} \frac{R_j^t}{N_j} + \alpha \frac{1}{N} \right] = (1-\alpha) \sum_i \sum_{j \to i} \frac{R_j^t}{N_j} + \alpha$$

Per ogni nodo j con *almeno un arco uscente*, il fattore R_j^t / N_j viene sommato per ciascun arco uscente (ce ne sono N_j in tutto):

$$\sum_i R_i^{t+1} = (1-\alpha) \sum_{j \text{ con un arco uscente}} R_j^t + \alpha$$

La somma è 1 solo se *non ci sono pozzi* (cioè, nodi senza archi uscenti). Altrimenti, la somma sarà < 1:

i pozzi assorbono importanza dalle pagine senza restituirla al sistema!

PageRank: eliminazione dei pozzi

Per risolvere il problema, si può operare in molti modi diversi:

- per ogni pozzo i, si aggiungono archi fittizi da i ad ogni altra pagina (cosicché i pozzi “cedano” uniformemente la loro importanza a tutte le pagine);

PageRank: eliminazione dei pozzi

Per risolvere il problema, si può operare in molti modi diversi:

- per ogni pozzo i, si aggiungono archi fittizi da i ad ogni altra pagina (cosicché i pozzi “cedano” uniformemente la loro importanza a tutte le pagine);
- equivalentemente, ad ogni passo di PageRank, si aggiunge a tutti gli R_i una stessa quantità in modo che la somma rimanga 1;

PageRank: eliminazione dei pozzi

Per risolvere il problema, si può operare in molti modi diversi:

- per ogni pozzo i, si aggiungono archi fittizi da i ad ogni altra pagina (cosicché i pozzi “cedano” uniformemente la loro importanza a tutte le pagine);
- equivalentemente, ad ogni passo di PageRank, si aggiunge a tutti gli R_i una stessa quantità in modo che la somma rimanga 1;
- si eliminano iterativamente i pozzi e si applica PageRank al resto del grafo; i pozzi vanno poi reintegrati alla fine.

PageRank: interpretazione stocastica

L'algoritmo di PageRank può essere interpretato come un processo stocastico. Assumiamo un *navigatore probabilistico* che naviga sul grafo del Web (senza pozzi) nel seguente modo:

- parte da una pagina i a caso;

PageRank: interpretazione stocastica

L'algoritmo di PageRank può essere interpretato come un processo stocastico. Assumiamo un *navigatore probabilistico* che naviga sul grafo del Web (senza pozzi) nel seguente modo:

- parte da una pagina i a caso;
- con probabilità $(1 - \alpha)$ segue uno dei link della pagina corrente;

PageRank: interpretazione stocastica

L'algoritmo di PageRank può essere interpretato come un processo stocastico. Assumiamo un *navigatore probabilistico* che naviga sul grafo del Web (senza pozzi) nel seguente modo:

- parte da una pagina i a caso;
- con probabilità $(1 - \alpha)$ segue uno dei link della pagina corrente;
- con probabilità α si muove verso una pagina a caso.

PageRank: interpretazione stocastica

L'algoritmo di PageRank può essere interpretato come un processo stocastico. Assumiamo un *navigatore probabilistico* che naviga sul grafo del Web (senza pozzi) nel seguente modo:

- parte da una pagina i a caso;
- con probabilità $(1 - \alpha)$ segue uno dei link della pagina corrente;
- con probabilità α si muove verso una pagina a caso.

Il rank R_i è la frazione di tempo trascorsa dal navigatore probabilistico nella pagina i.

PageRank: vantaggi/svantaggi

- viene calcolato con grande efficienza: il processo iterativo converge entro pochi passi;

PageRank: vantaggi/svantaggi

- viene calcolato con grande efficienza: il processo iterativo converge entro pochi passi;
- va calcolato una volta sola, in modo indipendente dalla query;

PageRank: vantaggi/svantaggi

- viene calcolato con grande efficienza: il processo iterativo converge entro pochi passi;
- va calcolato una volta sola, in modo indipendente dalla query;
- è possibile costruire insiemi di pagine “artefatte” che influiscono sul ranking;

PageRank: vantaggi/svantaggi

- viene calcolato con grande efficienza: il processo iterativo converge entro pochi passi;
- va calcolato una volta sola, in modo indipendente dalla query;
- è possibile costruire insiemi di pagine “artefatte” che influiscono sul ranking;
- l’indipendenza dalla query è un limite!

PageRank: vantaggi/svantaggi

- viene calcolato con grande efficienza: il processo iterativo converge entro pochi passi;
- va calcolato una volta sola, in modo indipendente dalla query;
- è possibile costruire insiemi di pagine “artefatte” che influiscono sul ranking;
- l’indipendenza dalla query è un limite!

Di solito va “aggiustato” usando un secondo ranking, basato sul contenuto (p.es., LSI).

HITS: idea

PageRank considera autorevoli pagine che sono molto puntate, anche se magari trattano argomenti eterogenei.

HITS: idea

PageRank considera autorevoli pagine che sono molto puntate, anche se magari trattano argomenti eterogenei.
Idea: considerare due categorie di pagine:

- *autorità*, cioè pagine autorevoli per la query che ci interessa;
- *punti focali* (hub), cioè pagine che puntano a pagine autorevoli.

HITS: idea

PageRank considera autorevoli pagine che sono molto puntate, anche se magari trattano argomenti eterogenei.
Idea: considerare due categorie di pagine:

- *autorità*, cioè pagine autorevoli per la query che ci interessa;
- *punti focali* (hub), cioè pagine che puntano a pagine autorevoli.

Esiste un principio di mutuo rinforzo fra autorità e hub: un'autorità (per la query Q) è una pagina puntata da molti hub; uno hub è una pagina che punta a molte autorità.

HITS: formule

Associamo ad ogni pagina i due pesi: a_i (il livello di autorevolezza di i) e h_i (il livello di "hubbità" di i):

$$a_i = \sum_{j \to i} h_j$$

$$h_i = \sum_{i \to j} a_j$$

Ad ogni passo, i vettori $\vec{a}$ e $\vec{h}$ vengono rinormalizzati a 1.

HITS: quale grafo?

Da quale grafo dobbiamo partire per applicare HITS?

HITS: quale grafo?

Da quale grafo dobbiamo partire per applicare HITS? Si deve trattare di un grafo “piccolo” ma che contenga sia le autorità che gli hub per la query di riferimento.

- si parte dall’insieme di pagine P_Q che soddisfano la query
- si aggiungono gli outlink (cioè le pagine v tali che $(u, v) \in E$ per qualche $u \in P_Q$)

HITS: quale grafo?

Da quale grafo dobbiamo partire per applicare HITS? Si deve trattare di un grafo “piccolo” ma che contenga sia le autorità che gli hub per la query di riferimento.

- si parte dall’insieme di pagine P_Q che soddisfano la query
- si aggiungono gli outlink (cioè le pagine v tali che $(u, v) \in E$ per qualche $u \in P_Q$)
- si aggiungono gli inlink (cioè le pagine v tali che $(v, u) \in E$ per qualche $u \in P_Q$)

HITS: quale grafo?

Da quale grafo dobbiamo partire per applicare HITS? Si deve trattare di un grafo “piccolo” ma che contenga sia le autorità che gli hub per la query di riferimento.

- si parte dall’insieme di pagine P_Q che soddisfano la query
- si aggiungono gli outlink (cioè le pagine v tali che $(u, v) \in E$ per qualche $u \in P_Q$)
- si aggiungono gli inlink (cioè le pagine v tali che $(v, u) \in E$ per qualche $u \in P_Q$)
- si tolgono i link navigazionali (cioè, quelli intra-host)

HITS: quale grafo?

Da quale grafo dobbiamo partire per applicare HITS? Si deve trattare di un grafo “piccolo” ma che contenga sia le autorità che gli hub per la query di riferimento.

- si parte dall’insieme di pagine P_Q che soddisfano la query
- si aggiungono gli outlink (cioè le pagine v tali che $(u, v) \in E$ per qualche $u \in P_Q$)
- si aggiungono gli inlink (cioè le pagine v tali che $(v, u) \in E$ per qualche $u \in P_Q$)
- si tolgono i link navigazionali (cioè, quelli intra-host)

HITS: interpretazione matriciale

Sia A la matrice di adiacenza del grafo che stiamo considerando (quello a cui applichiamo HITS). Le equazioni di ricorrenza si possono riscrivere come:

$$\begin{aligned} \vec{a} &= A^T \vec{h} = (A^T A)\vec{a} \\ \vec{h} &= A\vec{a} = (AA^T)\vec{h} \end{aligned}$$

HITS: interpretazione matriciale

Sia A la matrice di adiacenza del grafo che stiamo considerando (quello a cui applichiamo HITS). Le equazioni di ricorrenza si possono riscrivere come:

$$\begin{aligned} \vec{a} &= A^T \vec{h} = (A^T A)\vec{a} \\ \vec{h} &= A\vec{a} = (AA^T)\vec{h} \end{aligned}$$

Infatti, ad esempio, $a_i = \sum_{j \to i} h_j$ e

$$(A^T \vec{h})_i = \text{riga } i \text{ di } A^T \cdot \vec{h} = \text{colonna } i \text{ di } A \cdot \vec{h} = \sum_{j \to i} h_j .$$

Fatto

Fatto 1 *Sia M una matrice $n \times n$ con autovalori $\lambda_1, \ldots, \lambda_n$ tali che:*

$$|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \ldots |\lambda_n|.$$

Sia $\vec{\omega}_i$ l'autovettore corrispondente all'autovalore λ_i (cioè, $M\vec{\omega}_i = \lambda_i \vec{\omega}_i$), e sia $\vec{v}$ un vettore a componenti positive. Allora

$$\lim_{k \to \infty} M^k \vec{v}$$

è un vettore parallelo a $\vec{\omega}_1$.

Dimostrazione

È sempre possibile scrivere $\vec{v}$ come

$$\vec{v} = a_1\vec{\omega}_1 + \cdots + a_n\vec{\omega}_n.$$

Ora

$$M^k\vec{v} = a_1M^k\vec{\omega}_1 + \cdots + a_nM^k\vec{\omega}_n = a_1\lambda_1^k\vec{\omega}_1 + \cdots + a_n\lambda_n^k\vec{\omega}_n =$$

$$= a_1\lambda_1^k\left(\omega_1 + \frac{a_2}{a_1}\left(\frac{\lambda_2}{\lambda_1}\right)^k\vec{\omega}_2 + \cdots + \frac{a_n}{a_1}\left(\frac{\lambda_n}{\lambda_1}\right)^k\vec{\omega}_n\right)$$

Dimostrazione

È sempre possibile scrivere $\vec{v}$ come

$$\vec{v} = a_1\vec{\omega}_1 + \cdots + a_n\vec{\omega}_n.$$

Ora

$$M^k\vec{v} = a_1M^k\vec{\omega}_1 + \cdots + a_nM^k\vec{\omega}_n = a_1\lambda_1^k\vec{\omega}_1 + \cdots + a_n\lambda_n^k\vec{\omega}_n =$$

$$= a_1\lambda_1^k\left(\omega_1 + \frac{a_2}{a_1}\left(\frac{\lambda_2}{\lambda_1}\right)^k\vec{\omega}_2 + \cdots + \frac{a_n}{a_1}\left(\frac{\lambda_n}{\lambda_1}\right)^k\vec{\omega}_n\right)$$

Per $k \to \infty$, i fattori $(\frac{\lambda_i}{\lambda_1})^k$ vanno a zero (essendo $|\lambda_1| > |\lambda_i|$) e quindi

$$\lim_{k\to\infty} M^k\vec{v} = a_1\lambda_1^k\vec{\omega}_1 \parallel \vec{\omega}_1$$

Quindi...

...sotto ipotesi abbastanza generali, essendo

$$\vec{a} = (A^T A)\vec{a}$$
$$\vec{h} = (AA^T)\vec{h}$$

risulta che $\vec{a}$ e $\vec{h}$ convergono agli autovettori principali di $A^T A$ e AA^T, rispettivamente.

Quindi...

...sotto ipotesi abbastanza generali, essendo

$$\vec{a} = (A^T A)\vec{a}$$
$$\vec{h} = (AA^T)\vec{h}$$

risulta che $\vec{a}$ e $\vec{h}$ convergono agli autovettori principali di $A^T A$ e AA^T, rispettivamente.

Anche questo algoritmo può essere modificato tenendo conto di un fattore di spargimento.

HITS: interpretazione stocastica

L'algoritmo HITS può essere interpretato come un processo stocastico. Assumiamo un *navigatore probabilistico* che naviga sul grafo di HITS e che si può trovare in due stati $+$ e $-$:

- parte da una pagina i a caso e in uno stato a caso;

HITS: interpretazione stocastica

L'algoritmo HITS può essere interpretato come un processo stocastico. Assumiamo un *navigatore probabilistico* che naviga sul grafo di HITS e che si può trovare in due stati $+$ e $-$:

- parte da una pagina i a caso e in uno stato a caso;
- con probabilità α si muove verso una pagina a caso e sceglie in modo uniforme uno stato;

HITS: interpretazione stocastica

L'algoritmo HITS può essere interpretato come un processo stocastico. Assumiamo un *navigatore probabilistico* che naviga sul grafo di HITS e che si può trovare in due stati $+$ e $-$:

- parte da una pagina i a caso e in uno stato a caso;
- con probabilità α si muove verso una pagina a caso e sceglie in modo uniforme uno stato;
- con probabilità $(1-\alpha)$, se è nello stato $+$ segue un link a caso nella pagina corrente; se è nello stato $-$ sceglie a caso una pagina che punti a quella corrente.

HITS: interpretazione stocastica

L'algoritmo HITS può essere interpretato come un processo stocastico. Assumiamo un *navigatore probabilistico* che naviga sul grafo di HITS e che si può trovare in due stati $+$ e $-$:

- parte da una pagina i a caso e in uno stato a caso;
- con probabilità α si muove verso una pagina a caso e sceglie in modo uniforme uno stato;
- con probabilità $(1-\alpha)$, se è nello stato $+$ segue un link a caso nella pagina corrente; se è nello stato $-$ sceglie a caso una pagina che punti a quella corrente.

L'autorevolezza a_i è (proporzionale al) numero di volte che si è saltati nella pagina i essendo nello stato $+$.

HITS: pregi e difetti

- Essendo query-dependent, è molto più preciso di PageRank. . .

HITS: pregi e difetti

- Essendo query-dependent, è molto più preciso di PageRank. . .
- . . . ma il calcolo dell'autorevolezza va fatto all'atto della query, e richiede di determinare un grafo la cui dimensione è di solito notevole!

Individuazione di comunità di interesse

Il problema

I motori di ricerca danno risultati poco utilizzabili. Il ranking ne migliora solo limitatamente la qualità.

Il problema

I motori di ricerca danno risultati poco utilizzabili. Il ranking ne migliora solo limitatamente la qualità. Individuazione di comunità di interesse virtuali (=insieme di pagine che si occupano dello stesso argomento). Utilizzi:

- (pre-)categorizzazione automatica per portali gerarchici (*à la* Yahoo)
- realizzazione di *focused crawlers*.

Richiami: reti, flusso

Sia $G = (V, E)$ un grafo orientato, e si assuma che ad ogni arco $(i, j) \in E$ sia associata una *capacità* $c(i, j) \geq 0$. Siano inoltre $s, t \in V$.

Richiami: reti, flusso

Sia $G = (V, E)$ un grafo orientato, e si assuma che ad ogni arco $(i, j) \in E$ sia associata una *capacità* $c(i, j) \geq 0$. Siano inoltre $s, t \in V$.

Un *flusso* è un'assegnazione di valori $f(i, j)$ agli archi in modo tale che:

- $\forall (i, j) \in E, 0 \leq f(i, j) \leq c(i, j)$.
- se $i \notin \{s, t\}$, $\sum_{i \to j} f(i, j) = \sum_{j \to i} f(j, i)$.

Il valore $\sum_{s \to i} f(s, i) = \sum_{i \to t} f(i, t)$ è detto *(valore del) flusso* f.

Richiami: tagli

Un *taglio* in un grafo $G = (V, E)$ è una bipartizione di V in due insiemi (V_1, V_2); se $s, t \in V$, un *taglio* s/t è un taglio (S, T) con $s \in S$ e $t \in T$.

Richiami: tagli

Un *taglio* in un grafo $G = (V, E)$ è una bipartizione di V in due insiemi (V_1, V_2); se $s, t \in V$, un *taglio* s/t è un taglio (S, T) con $s \in S$ e $t \in T$.
Il *valore del taglio* (S, T) è

$$\sum_{i \in S, j \in T, i \to j} c(i, j).$$

Richiami: tagli

Un *taglio* in un grafo $G = (V, E)$ è una bipartizione di V in due insiemi (V_1, V_2); se $s, t \in V$, un *taglio* s/t è un taglio (S, T) con $s \in S$ e $t \in T$.
Il *valore del taglio* (S, T) è

$$\sum_{i \in S, j \in T, i \to j} c(i, j).$$

Teorema 1 (Max-flow, min-cut) *Il massimo valore di flusso coincide con il minimo valore di taglio.*

Richiami: flussi e tagli

Per trovare il minimo taglio, basta trovare il massimo flusso e poi visitare il grafo in profondità a partire dalla sorgente. Gli archi con flusso uguale alla capacità (archi saturati) costituiscono il taglio minimo.

Perché i tagli?

Cos’è una comunità virtuale?

Perché i tagli?

Cos'è una comunità virtuale?
È un insieme di nodi del grafo C tali che, per ogni $v \in C$, ci sono più link da v verso nodi di C che non verso nodi di $V \setminus C$.

Perché i tagli?

Cos'è una comunità virtuale?
È un insieme di nodi del grafo C tali che, per ogni $v \in C$, ci sono più link da v verso nodi di C che non verso nodi di $V \setminus C$.

Vogliamo determinare comunità conoscendo prima alcuni membri. Vogliamo evitare soluzioni banali (p.es. $C = V$).

Metodo di Flakes/Lawrence/Giles

Supponiamo di voler determinare una comunità che contenga un membro noto s.

- Scegliamo un nodo t che non sta nella comunità ma che è raggiungibile da tutti i nodi della comunità.

Metodo di Flakes/Lawrence/Giles

Supponiamo di voler determinare una comunità che contenga un membro noto s.

- Scegliamo un nodo t che non sta nella comunità ma che è raggiungibile da tutti i nodi della comunità.
- Calcoliamo il min-cut s/t (capacità unitarie).

Metodo di Flakes/Lawrence/Giles

Supponiamo di voler determinare una comunità che contenga un membro noto s.

- Scegliamo un nodo t che non sta nella comunità ma che è raggiungibile da tutti i nodi della comunità.
- Calcoliamo il min-cut s/t (capacità unitarie).
- La *candidata comunità* è l'insieme dei nodi C che si trovano *prima* del taglio.

Metodo di Flakes/Lawrence/Giles

Supponiamo di voler determinare una comunità che contenga un membro noto s.

- Scegliamo un nodo t che non sta nella comunità ma che è raggiungibile da tutti i nodi della comunità.
- Calcoliamo il min-cut s/t (capacità unitarie).
- La *candidata comunità* è l'insieme dei nodi C che si trovano *prima* del taglio.
- La candidata comunità è una comunità posto che $s^{\#}$ e $t^{\#}$ siano $\geq |C|$, dove

$$s^{\#} = \text{num. archi da } s \text{ a } C \setminus s;$$

$$t^{\#} = \text{num. archi da } V \setminus (C \cup t) \text{ a } t.$$

Note

- Scelta di s: Un insieme di pagine rilevanti per la comunità “condensato” in un nodo virtuale.

Note

- Scelta di s: Un insieme di pagine rilevanti per la comunità “condensato” in un nodo virtuale.
- Scelta di t: Una piccola collezione di portali web e hub “condensata” in un nodo virtuale.

Note

- Scelta di s: Un insieme di pagine rilevanti per la comunità “condensato” in un nodo virtuale.
- Scelta di t: Una piccola collezione di portali web e hub “condensata” in un nodo virtuale.
- Che algoritmo? Con Ford-Fulkerson, complessità $O(|V||E|^2)$. Ma esistono algoritmi più efficienti (pre-flush, Edmonds e Karp, etc.) e che non richiedono di tenere il grafo in memoria!

Note

- Scelta di s: Un insieme di pagine rilevanti per la comunità “condensato” in un nodo virtuale.
- Scelta di t: Una piccola collezione di portali web e hub “condensata” in un nodo virtuale.
- Che algoritmo? Con Ford-Fulkerson, complessità $O(|V||E|^2)$. Ma esistono algoritmi più efficienti (pre-flush, Edmonds e Karp, etc.) e che non richiedono di tenere il grafo in memoria!

Il web come small-world

Richiami

Dato un grafo (non orientato) $G = (V, E)$, chiamiamo *densità* δ di G il valore $|E|/\binom{|V|}{2}$.

Richiami

Dato un grafo (non orientato) $G = (V, E)$, chiamiamo *densità* δ di G il valore $|E|/\binom{|V|}{2}$.

Dati $x, y \in V$, indichiamo con $d(x, y)$ la lunghezza del cammino minimo che va da x a y (∞ se tale cammino non esiste).

Cammino caratteristico

Dato un grafo $G = (V, E)$ fortemente connesso, definiamo la *(lunghezza del) cammino caratteristico* come la distanza media fra due nodi:

$$L_G = \frac{\sum_{x \neq y} d(x, y)}{\binom{|V|}{2}}$$

Cammino caratteristico

Dato un grafo $G = (V, E)$ fortemente connesso, definiamo la *(lunghezza del) cammino caratteristico* come la distanza media fra due nodi:

$$L_G = \frac{\sum_{x \neq y} d(x, y)}{\binom{|V|}{2}}$$

L_G misura il "diametro medio" del grafo, cioè quanto sono distanti in media i nodi.

Coefficiente di clustering

Dato un grafo $G = (V, E)$, e un nodo $v \in V$, definiamo

$C(v) =$ frazione di vicini di v che sono adiacenti fra loro

e chiamiamo

$$C_G = \frac{\sum_x C(x)}{|V|}.$$

Coefficiente di clustering

Dato un grafo $G = (V, E)$, e un nodo $v \in V$, definiamo

$C(v) =$ frazione di vicini di v che sono adiacenti fra loro

e chiamiamo

$$C_G = \frac{\sum_x C(x)}{|V|}.$$

C_G misura quanto è "localmente denso" il grafo (quanti dei miei amici sono amici fra loro?).

Definizione

Fissati n e δ, considerate un grafo casuale con n nodi e densità δ. Indichiamo con $\hat{L}(n, \delta)$ e $\hat{C}(n, \delta)$ i valori attesi di L_G e C_G.

Definizione

Fissati n e δ, considerate un grafo casuale con n nodi e densità δ. Indichiamo con $\hat{L}(n, \delta)$ e $\hat{C}(n, \delta)$ i valori attesi di L_G e C_G.

Un grafo G con n nodi e densità δ è uno *small-world* sse L_G è vicina a $\hat{L}(n, \delta)$ (o poco inferiore), ma C_G è molto più grande di $\hat{C}(n, \delta)$.

Esempi

Esempi di small-world:

- esperimento di Milgram

Esempi

Esempi di small-world:

- esperimento di Milgram
- numero di Erdős

Esempi

Esempi di small-world:

- esperimento di Milgram
- numero di Erdős
- Kevin Bacon Graph

Esempi

Esempi di small-world:

- esperimento di Milgram
- numero di Erdős
- Kevin Bacon Graph
- connessioni neurali del *Caenorhabditis elegans*

Esempi

Esempi di small-world:

- esperimento di Milgram
- numero di Erdős
- Kevin Bacon Graph
- connessioni neurali del *Caenorhabditis elegans*
- reti di distribuzione elettrica

Esempi

Esempi di small-world:

- esperimento di Milgram
- numero di Erdős
- Kevin Bacon Graph
- connessioni neurali del *Caenorhabditis elegans*
- reti di distribuzione elettrica
- il Web

SW: fra regolari e random

- Un grafo altamente regolare tende ad avere alto coefficiente di clustering e alta lunghezza di cammino caratteristico.

SW: fra regolari e random

- Un grafo altamente regolare tende ad avere alto coefficiente di clustering e alta lunghezza di cammino caratteristico.
- Un grafo casuale ha un cammino caratteristico molto breve, ma ha basso coefficiente di clustering.

SW: fra regolari e random

- Un grafo altamente regolare tende ad avere alto coefficiente di clustering e alta lunghezza di cammino caratteristico.
- Un grafo casuale ha un cammino caratteristico molto breve, ma ha basso coefficiente di clustering.
- Uno small world è una via di mezzo: ha alto coefficiente di clustering e cammino caratteristico breve.

Come si costruisce uno SW

È possibile costruire in modo casuale uno SW? Sì, in molti modi diversi. La costruzione di Strogatz-Watts:

- n vertici su un cerchio; ogni vertice è collegato a tutti i nodi entro distanza $k = \frac{\delta(n-1)}{2}$;

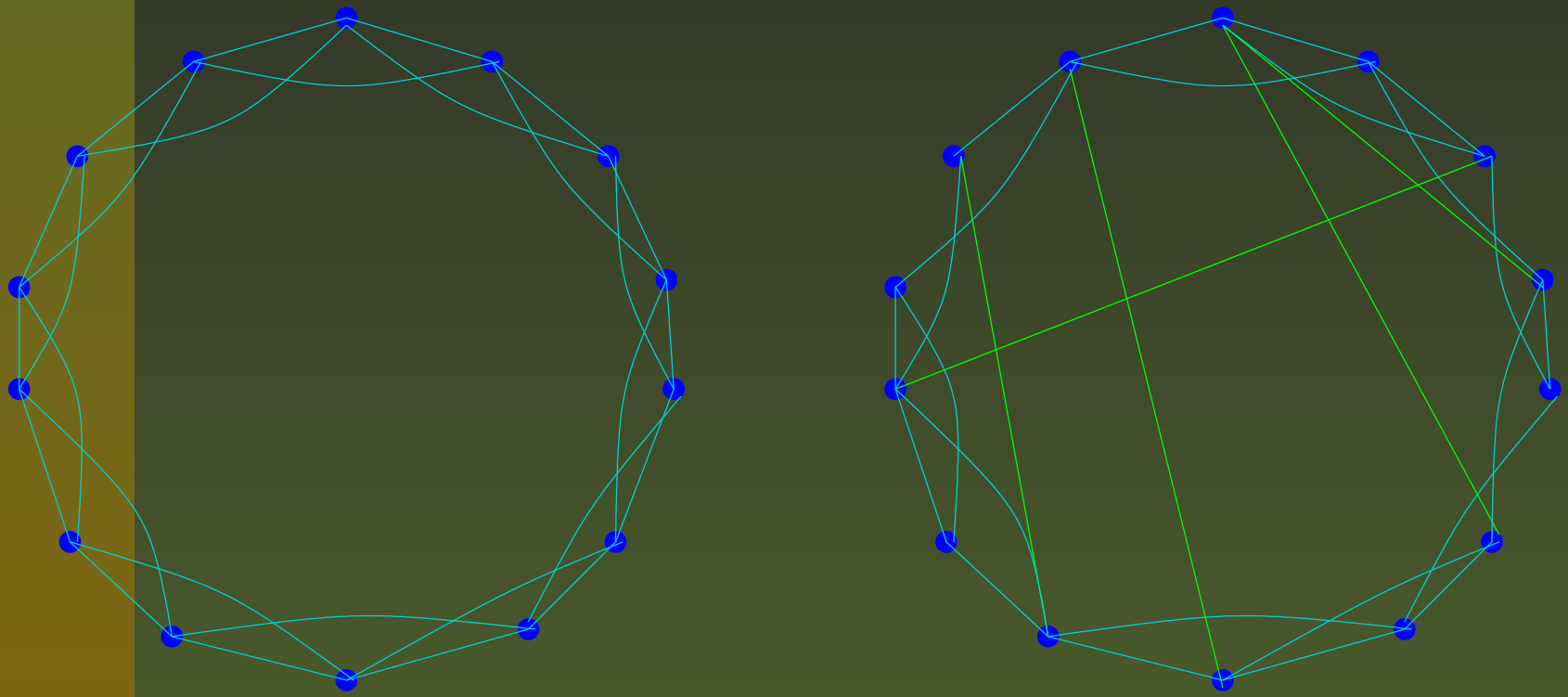
Come si costruisce uno SW

È possibile costruire in modo casuale uno SW? Sì, in molti modi diversi. La costruzione di Strogatz-Watts:

- n vertici su un cerchio; ogni vertice è collegato a tutti i nodi entro distanza $k = \frac{\delta(n-1)}{2}$;
- una frazione $p \in [0, 1]$ di archi vengono "rediretti": cioè, cancellati e riaggiunti al grafo fra nodi scelti a caso.

Esempio

In questo esempio $n = 14$ e $\delta = 4/13$.



Ponendo $p = 1/4$ (cioè, redirigendo $28/4 = 7$ archi)...

Valori di p

Quando $p = 0$, la rete è altamente regolare (è una *circolante*). Quando $p = 1$, la rete è casuale: