UNIVERSITA' DEGLI STUDI
ROMA TRE

Facoltà di Ingegneria
Corso di Laurea in Ingegneria Informatica

Metodi e strumenti per l'analisi dell'evoluzione dei rapporti tra fornitori e fruitori di servizi di connettività in Internet

Candidato:
Massimo Rimondini

Relatore:
prof. Giuseppe Di Battista

Correlatori:
Maurizio Patrignani
Maurizio Pizzonia

28 Maggio 2003

Alla mia famiglia, per avermi sopportato durante il difficile periodo della stesura di questo lavoro;

a Gabriele, per avermi dato la possibilità di collaborare con lui, costruendo così insieme la nostra cultura ed una straordinaria amicizia;

ad Ivan e Alexandra, per avermi rallegrato con i loro scherzi e dispetti e per avermi saputo aiutare con idee geniali che solo loro avrebbero potuto avere;

ad Alessandra, per il suo fondamentale appoggio morale, che è stato determinante in molte fasi del nostro ciclo di studi;

a tutti i miei amici dell'Università, perché è grazie all'amicizia che ho con loro, al loro sostegno ed al loro brio che ho potuto portare a termine questo percorso.

Indice

Elenco delle figure

Introduzione

La rete Internet è stata in questi ultimi anni oggetto di un notevole numero di analisi finalizzate a cercare di catturarne i comportamenti intrinseci con l'obiettivo di migliorarne sempre più il funzionamento. Queste analisi hanno coperto vari aspetti, sia sul fronte della definizione di nuovi teoremi che ne governano le proprietà, sia su quello della valutazione *sul campo* delle sue prestazioni in situazioni particolari.

In questo lavoro si affronta un problema relativamente recente, con degli interessanti risvolti di carattere sia concettuale che operativo. Gli attori in gioco sono i fornitori ed i fruitori di servizi di connettività. Come accade in molti altri contesti, il livello di astrazione su cui ci si può porre per prendere in considerazione un problema può essere diverso a seconda delle esigenze del caso. In questa situazione si assumerà che sia i fornitori, che i fruitori siano grosse organizzazioni a livello regionale o nazionale. Ciascuna di queste organizzazioni prende il nome di *Autonomous System*, ed è strutturata in modo tale da apparire all'esterno come entità gestita in modo uniforme e coerente sia dal punto di vista commerciale che da quello del routing.

I rapporti di cui si parla nel titolo sono rapporti di carattere commerciale, ovvero veri e propri contratti che i fruitori stipulano con i fornitori per ottenere l'accesso all'intera rete Internet. In realtà, esistono molte possibili sfumature in queste relazioni, al punto che spesso nessuna delle organizzazioni in gioco può considerarsi esclusivamente fornitrice od utilizzatrice del servizio di connettività.

La conoscenza di questi rapporti risulta particolarmente utile a chi, per esempio, volesse instaurare nuove relazioni, poiché sarebbe in grado di scegliere l'alternativa che offre il servizio più adeguato alle proprie esigenze. Tuttavia, quasi nessuna organizzazione rende pubbliche le informazioni sulla natura dei propri contratti, e così non è possibile conoscere questi rapporti mediante un semplice processo di consultazione. D'altra parte, anche per quello che riguarda le informazioni di routing, spesso si evita di rendere esplicite le scelte fatte, perché da esse sarebbe possibile risalire ai rapporti che le hanno determinate.

Per queste ragioni, sono stati introdotti diversi algoritmi per dedurre in modo automatico e ragionevolmente affidabile le relazioni esistenti tra Autonomous Systems, basandosi esclusivamente sulle informazioni scambiate tra di essi. In sede di definizione di questi algoritmi sono stati affrontati diversi problemi di carattere teorico, che rendono il procedimento di inferenza fondato su solide basi concettuali. È accaduto invece abbastanza raramente di porsi di fronte al problema con un atteggiamento più pragmatico, mirato a cogliere i comportamenti degli algoritmi in situazioni particolari.

In questa trattazione si introduce una serie di elementi che contribuiscono alla definizione di una metodologia per analizzare l'evoluzione dei rapporti commerciali nel tempo. L'obiettivo che ci si pone è quantomeno duplice: da un lato si vuole introdurre un procedimento sistematico per studiare tale evoluzione, intendendo in tal modo arricchire il significato dei risultati di un processo di inferenza; dall'altro, si vuole cercare di convalidare la qualità delle relazioni calcolate con un opportuno algoritmo, per capire quanto esso sia affidabile e quanto, dunque, risulti significativa l'informazione temporale ad esse aggiunta.

In seguito ad opportune fasi di analisi e progettazione, viene inoltre implementato un ambiente che consenta di produrre delle informazioni molto dettagliate circa il comportamento delle relazioni inferite. Viene inoltre effettuata una serie di sperimentazioni su un algoritmo di inferenza in particolare, utilizzando come contesto tre situazioni particolarmente significative dal punto di vista commerciale. Al termine di ciascuna sperimentazione viene utilizzato l'ambiente prodotto al fine di trarre delle conclusioni sul comportamento dell'algoritmo scelto.

Per rendere possibili queste sperimentazioni viene preliminarmente effettuata un'accurata ottimizzazione dell'algoritmo scelto, che lo rende molto più veloce ed applicabile in modo massiccio, come richiesto dalle prove in questione.

Il resto di questo documento è organizzato nel seguente modo:

- nel capitolo 1 vengono richiamate alcune nozioni basilari per poter affrontare il problema in esame, riguardanti il protocollo BGP ed i suoi meccanismi di funzionamento; in esso viene inoltre illustrata una breve panoramica dei tipi di rapporto che, a tutt'oggi, risultano governare la maggior parte dei contratti; infine, viene qui presentata anche una serie di motivazioni che giustificano lo studio dell'evoluzione delle relazioni, nonché gli obiettivi con cui tale studio viene affrontato;

- nel capitolo 2 viene formalizzato in modo rigoroso il problema dell'inferenza di relazioni tra Autonomous System; vengono poi presentati i tre approcci finora utilizzati per risolverlo, descrivendo per ciascuno di essi i presupposti su cui si basa e l'algoritmo che utilizza;

- nel capitolo 3 viene motivata la necessità di introdurre un ambiente che consenta di effettuare lo studio dell'evoluzione delle relazioni in modo sistematico, chiaro e rigoroso; vengono analizzate le caratteristiche che tale ambiente dovrebbe avere e si forniscono delle linee progettuali per la sua realizzazione; infine, viene descritta una sua implementazione che va sotto il nome di TORQUE, della quale si presenta anche un esempio di utilizzo completo;

- nel capitolo 4 viene scelto l'algoritmo di inferenza delle relazioni da utilizzare successivamente in fase di sperimentazione; la scelta ricade sulla soluzione presentata in [20], della quale viene fornita una descrizione dettagliata; in questo capitolo viene inoltre analizzata a fondo un'implementazione esistente di tale algoritmo, al file di ricercare in essa operazioni migliorabili dal punto di vista dell'efficienza; infine, viene presentata una versione ottimizzata dello stesso algoritmo, che impiega per l'esecuzione meno del 20% del tempo impiegato dalla versione precedente;

- nel capitolo 5 viene illustrato il processo di sperimentazione operato utilizzando l'algoritmo di inferenza descritto nel capitolo precedente, applicato a dati relativi a periodi particolarmente interessanti dal punto di vista commerciale; le prove effettuate sono tre: la prima analizza il comportamento della versione non ottimizzata dell'algoritmo su dati immediatamente disponibili ed utilizzati nella soluzione proposta da [8]; la seconda considera i dati del server Oregon Route Views ([22]), in un arco temporale di una settimana nella quale non abbiano avuto luogo eventi commerciali di rilievo; la terza considera ancora i dati di Route Views, ma stavolta isolando due settimane con un certo numero di cambiamenti commerciali significativi;

- nel capitolo 6 vengono infine brevemente tratte delle conclusioni sul comportamento dell'algoritmo, e viene presentato un elenco riassuntivo dei nuovi problemi sollevati in ogni capitolo e rimasti, per il momento, irrisolti.

Capitolo 1

Background ed obiettivi

In questo capitolo vengono presentati alcuni concetti fondamentali, necessari per comprendere il problema che viene affrontato; vengono inoltre illustrate le finalità dell'analisi e le motivazioni che rendono questo tema di particolare interesse nell'ambito delle reti di computer.

1.1 Definizione di Autonomous System

L'insieme dei calcolatori che compongono Internet nel suo complesso può essere considerato su scala più o meno larga. La visione più capillare corrisponde a considerare come entità distinta ciascun calcolatore che sia connesso alla rete; man mano che si eleva il livello di astrazione è poi possibile utilizzare modelli che rappresentano l'interconnessione di componenti di dimensioni sempre maggiori (*LAN*, *WAN*, ecc.). Il problema che viene qui affrontato richiede di utilizzare una visione di livello molto alto, poiché necessita di poter associare degli accordi di natura prettamente commerciale (tipicamente non relativi a piccoli gruppi di calcolatori) a particolari comportamenti adottati nell'utilizzo di una rete.

Il livello di astrazione utilizzato nel resto di questa trattazione corrisponde, dunque, a considerare l'intera rete Internet come suddivisa in gruppi di calcolatori che occupano tipicamente aree geografiche di dimensioni notevoli (intere regioni o addirittura stati). Questi gruppi prendono il nome di *Autonomous Systems*, e vengono generalmente definiti ([6]) come insiemi di router[1] sotto il controllo di un'unica amministrazione, che utilizzano un unico protocollo (*IGP, Interior Gateway Protocol*) e metriche comuni per l'instradamento interno dei pacchetti, e che comunicano con router di altri Autonomous Systems mediante un altro protocollo (*EGP, Exterior Gateway Protocol*).

[1] Questa definizione, e quelle presentate nel seguito, utilizzano il termine *router* piuttosto che *calcolatore* perché ciò che veramente interessa per individuare un Autonomous System è il comportamento delle macchine al suo interno dal solo punto di vista dell'instradamento dei dati.

In realtà, con il passare del tempo, questa definizione è stata gradualmente rilassata, sicché ora viene tipicamente classificato come Autonomous System anche un insieme di router che utilizzi più protocolli IGP. Più correttamente, dunque, si può affermare che

Definizione 1.1 (Autonomous System) *Un Autonomous System è un insieme di router sotto il controllo di un'unica amministrazione, che appare agli altri Autonomous Systems come un gruppo di destinazioni organizzate coerentemente e che adotta e mantiene, al proprio interno, un'unica politica di routing.*

Il concetto di Autonomous System è nato proprio per consentire a ciascuna organizzazione di mantenere in uso i propri protocolli di routing senza doversi necessariamente adeguare a quelli delle organizzazioni con cui essa comunica.

A ciascun Autonomous System viene associato un numero $1 \leq ASn \leq 65535$; i numeri identificativi degli Autonomous System sono univoci in tutto il mondo (ad eccezione dell'intervallo $64512 \leq ASn \leq 65534$, utilizzato per scopi privati), e vengono gestiti globalmente dalla *IANA* (*Internet Assigned Numbers Authority*) [1]. L'assegnazione e la gestione a livello regionale di questi numeri è cura di 4 grandi organizzazioni, ciascuna avente competenze su una specifica area geografica[2] (*RIR, Regional Internet Registries*):

- APNIC (*Asia Pacific Network Information Centre*) [2], che copre gli stati dell'Asia meridionale e sud-orientale, oltre ad alcune zone dell'Oceano Pacifico (fino all'Australia e alla Nuova Zelanda);
- ARIN (*American Registry for Internet Numbers*) [3], che amministra l'America Settentrionale, parte delle zone Caraibiche e l'Africa sub-equatoriale;
- LACNIC (*Latin American and Caribbean Internet Addresses Registry*) [4], che controlla l'America Latina e le zone Caraibiche;
- RIPE NCC (*Réseaux IP Européens Network Coordination Centre*) [5], cui competono l'Europa, l'Asia centrale e centro-orientale e le zone dell'Africa al di sopra dell'Equatore.

A loro volta, queste organizzazioni allocano blocchi di indirizzi IP a registri locali (*LIR, Local Internet Registries*), i quali infine li assegnano ai singoli utenti che ne facciano richiesta.

1.2 Il protocollo di routing BGP

Come già messo in evidenza, l'analisi che è oggetto di questa trattazione richiede che ci si ponga ad un livello di astrazione tale da considerare la rete Internet come interconnessione di Autonomous Systems. D'altra parte, è abbastanza immediato idenfiticare fornitori e fruitori di servizi di connettività proprio con gli Autonomous Systems, e l'offerta di tali servizi con la possibilità di farli comunicare tra loro. Poiché tale comunicazione è attualmente resa possibile

[2] Una lista completa dei Paesi assegnati a ciascuna organizzazione può essere consultata all'indirizzo `http://www.arin.net/library/internet_info/countries.html`, mentre per informazioni riguardo all'allocazione degli intervalli di numerazione degli Autonomous Systems è possibile far riferimento al file `http://www.iana.org/assignments/as-numbers`.

dall'utilizzo del protocollo *BGP* (*Border Gateway Protocol*), è proprio sui meccanismi di funzionamento di quest'ultimo che si fondano tutti gli approcci finora adottati per affrontare il problema in esame.

1.2.1 Gestione degli Autonomous Systems in BGP

I router che fanno parte di un Autonomous System vengono generalmente classificati in *interior router* ed *exterior* (o *border*) *router*. I primi vengono utilizzati per l'instradamento dei dati nell'ambito di uno stesso Autonomous System, e fanno uso di protocolli generalmente denominati IGP; i secondi sono invece configurati per dialogare con altri exterior router di altri Autonomous Systems, e dunque per consentire la comunicazione tra macchine appartenenti ad Autonomous Systems diversi. È compito di questi ultimi mantenere aggiornate tutte le informazioni riguardanti la reciproca raggiungibilità tra le macchine di ciascun Autonomous System, e ciò avviene per mezzo del protocollo BGP.

1.2.2 Principio di funzionamento del protocollo

Il protocollo BGP (definito rigorosamente in [6]) fa parte della tipologia di protocolli chiamati *distance vector*. Con questa espressione si individua un insieme di protocolli accomunati dal seguente meccanismo di funzionamento:

1. all'atto dell'accensione, ogni router conosce soltanto il modo di raggiungere le reti ad esso direttamente connesse, e costruisce una propria tabella di instradamento sulla base di questa sola informazione;

2. ciascun router elabora la tabella di instradamento ad esso nota (*RIB, Routing Information Base*) per produrne una versione opportunamente filtrata (*FIB, Forwarding Information Base*), successivamente convertita in un formato esportabile verso altri router;

3. la nuova tabella così calcolata viene comunicata ai router adiacenti;

4. a loro volta, questi ultimi ricalcolano le proprie tabelle di instradamento sulla base delle informazioni appena ricevute;

5. ogni router che abbia ricalcolato la propria tabella di instradamento procede poi come al punto 2.

Questo criterio di propagazione delle informazioni di raggiungibilità è presente anche in BGP, anche se con alcuni elementi di differenza.

Innanzitutto, i router si scambiano informazioni di raggiungibilità soltanto se effettivamente sono configurati per farlo; nel caso di BGP, questo significa che ogni router deve essere esplicitamente impostato per riconoscere come propri vicini i router con i quali deve scambiarsi tali informazioni. Il comando `neighbor` serve proprio a stabilire quali siano questi router. Per ciascun vicino il router apre una così detta *sessione di peering*, che altro non è che una connessione TCP sulla quale viaggiano le informazioni di routing. Proprio per la natura di questa connessione, è importante tenere presente che

> il fatto che un router abbia iniziato una sessione di peering con un altro non comporta necessariamente che i due router siano fisicamente connessi in modo diretto: la sessione può anche essere instaurata passando attraverso router intermedi.

Questa considerazione sottolinea la necessità di tenere presente la distinzione tra connettività fisica e connettività logica. In [7] si afferma che il protocollo BGP può addiruttura non assicurare la comunicazione tra ogni coppia di sistemi, anche nel caso in cui la topologia che li collega fosse completamente connessa.

Altra differenza con i protocolli distance vector "standard" è costituita dalla natura delle informazioni scambiate tra partecipanti ad un peering. Infatti, piuttosto che inviare ai vicini un'intera tabella di instradamento (la FIB), ciascun router comunica degli *annunci*, nei quali dichiara la propria capacità di raggiungere certe destinazioni. Un annuncio tipicamente contiene diverse informazioni, di cui, comunque, le più importanti ai fini del problema in esame sono:

- un blocco di indirizzi IP contigui, specificato sotto forma di prefisso; un prefisso altro non è che una coppia {indirizzo IP, netmask}, che individua un insieme di indirizzi appartenenti ad un'unica rete; questa informazione viene chiamata *NLRI* (*Network Layer Reachability Information*), e costituisce, di fatto, l'oggetto dell'annuncio; talvolta, al prefisso annunciato si dà anche il nome di *rotta*;
- l'indirizzo IP del router al quale inoltrare i pacchetti destinati al prefisso annunciato; tale indirizzo prende anche il nome di *next hop*;
- il percorso finora seguito dall'annuncio, in termini di Autonomous Systems attraversati; questo campo viene in genere denominato *AS path*; ciascun router che riceva l'annuncio e scelga di ritrasmetterlo antepone il numero dell'Autonomous System in cui si trova al contenuto corrente del campo AS path;

Se ad un router arrivano più annunci relativi ad uno stesso prefisso, quest'ultimo attiva un *processo decisionale* che lo porta a preferire solo una delle alternative che ha a disposizione per raggiungerlo. In seguito, ogni router annuncia, per ogni prefisso, soltanto l'alternativa che ha ritenuto migliore.

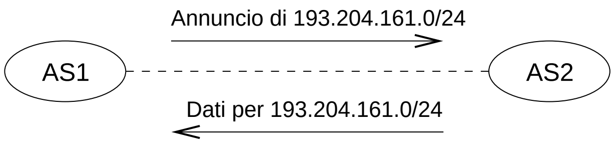

Figura 1.1: Esempio di annuncio BGP

Nella figura 1.1 viene rappresentata una situazione in cui un Autonomous System (AS1) comunichi ad un altro (AS2) la propria disponibilità a recapitare pacchetti verso gli indirizzi IP compresi nel prefisso 193.204.161.0/24; come risultato, quando necessario, l'Autonomous System AS2 inoltrerà i dati da recapitare a tali indirizzi proprio ad AS1, il quale si occuperà poi a sua volta di farli giungere alla destinazione richiesta (eventualmente attraversando altri Autonomous Systems). La linea tratteggiata che collega i due Autonomous Systems rappresenta la sessione di peering (tra due border routers dei due Autonomous Systems) sulla quale è stato trasferito l'annuncio. Si noti come i dati per una certa destinazione viaggino sempre in verso opposto all'annuncio di tale destinazione.

Oltre agli annunci che comunicano disponibilità di nuove rotte, BGP prevede anche messaggi di *ritiro* (*withdrawal*), utilizzati nel momento in cui un Autonomous System non sia più disponibile (per esempio, per via della caduta di un link) a recapitare dati verso una certa destinazione.

1.2.3 Politiche di instradamento

Una delle caratteristiche peculiari del protocollo BGP è costituita dalla possibilità di vincolare i percorsi seguiti dai pacchetti sulla base di necessità di carattere economico, sociale o politico. Esistono, infatti, opportuni comandi di configurazione che consentono di limitare la diffusione degli annunci BGP in modo arbitrario. Le regole che è possibile imporre a tal fine prendono il nome di *politiche*; ciascun router può essere configurato per adottare le proprie politiche, e può utilizzare strategie diverse per gli annunci ricevuti (*politiche di importazione*) e per quelli in fase di uscita (*politiche di esportazione*).

L'introduzione del concetto di politiche consente di definire in modo più preciso quali siano i contenuti della RIB, classificandoli in tre categorie che corrispondono alle tre fasi fondamentali delle elaborazioni effettuate dai router che utilizzano il protocollo BGP:

- `Adj-RIBs-In`: si tratta delle informazioni acquisite da annunci provenienti da altri Autonomous Systems; queste informazioni vengono utilizzate come input del processo decisionale, finalizzato a selezionare, tra le varie alternative disponibili per raggiungere ogni rotta, quella ritenuta migliore;

- `Loc-RIB`: questa porzione della RIB contiene i risultati del processo decisionale illustrato sopra, a loro volta filtrati in accordo con le politiche di importazione; questo insieme costituisce, dunque, la tabella di instradamento che il router utilizza per inviare effettivamente i dati;

- `Adj-RIBs-Out`: qui vengono immagazzinate le informazioni che il router ha filtrato sulla base delle politiche di esportazione, e che, quindi, sono pronte per essere trasmesse sotto forma di annunci ai propri peers.

Il processo decisionale di cui si è parlato sopra prende in considerazione molti altri attributi e metriche che qui non vengono riportati. A puro titolo di esempio basti sapere che, se un prefisso può essere raggiunto utilizzando più di un'alternativa, il router sceglierà quella

- alla quale ha attribuito il maggior grado di preferenza (dipendente dalle politiche con cui è configurato), oppure, a parità di preferenza, quella

- che abbia l'AS path più corto, oppure, a parità di lunghezza di AS path, quella

- che sia stata originata dal router con identificatore più basso.

L'introduzione delle politiche fa sì che BGP non si comporti (come avrebbe fatto in loro assenza) come un algoritmo *shortest path* puro, poiché le scelte fatte per l'instradamento sono influenzate da criteri che sono del tutto arbitrari ed ostacolano il conseguimento dell'ottimalità dei cammini.

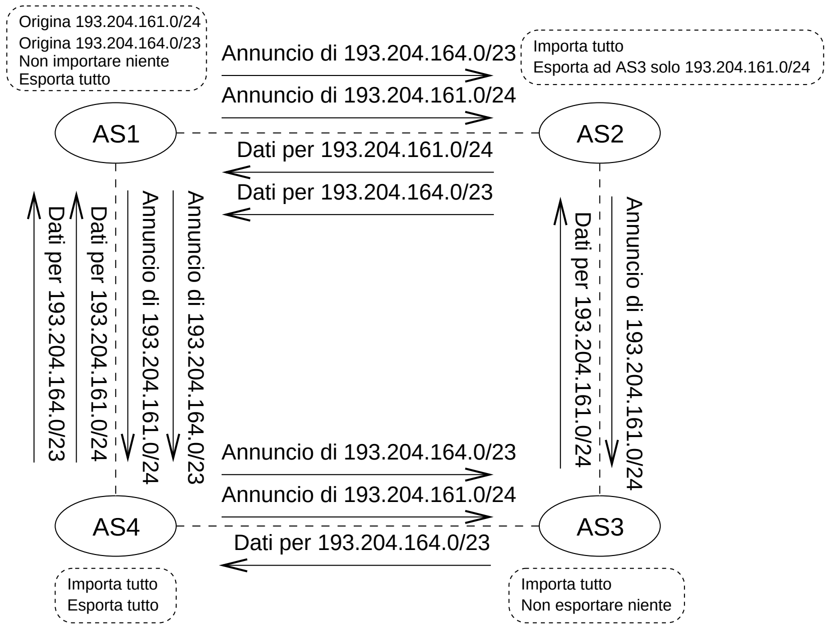

Figura 1.2: Esempio di utilizzo delle politiche

Le politiche che è possibile definire sono di complessità del tutto arbitraria. La struttura più semplice consiste nel limitarsi a bloccare del tutto degli annunci in ingresso od in uscita, come nell'esempio mostrato nella figura 1.2. In essa si può osservare quale sia il percorso degli annunci delle rotte originate dall'AS1. Quest'ultimo comunica ad entrambi gli Autonomous Systems adiacenti la propria capacità di recapitare pacchetti verso queste destinazioni. A loro volta, l'AS2 e l'AS4 propagano questa informazione secondo le politiche con cui sono stati configurati (riportate nei rettangoli con bordo tratteggiato): l'AS2 annuncia soltanto 193.204.161.0/24, mentre l'AS4 annuncia entrambe le rotte. Il risultato è che l'AS3, che si limita ad importare rotte senza effettuare annunci a sua volta, deve necessariamente utilizzare il link verso AS4 per recapitare dati verso indirizzi in 193.204.164.0/23, mentre, in base ai criteri del proprio processo decisionale, può, ad esempio, scegliere di preferire il link verso AS2 per l'invio dei dati verso 193.204.161.0/24. La stessa scelta sarebbe potuta scaturire qualora AS3 fosse stato configurato per importare da AS4 soltanto 193.204.164.0/23: in questo caso, il processo decisionale non avrebbe avuto alcuna influenza, per via della mancanza di alternative per raggiungere le varie destinazioni, ma i dati sarebbero ugualmente stati inoltrati secondo le direzioni indicate in figura.

Altri tipi di politiche consistono nell'alterare alcuni attributi che viaggiano insieme all'annuncio della rotta, dei quali, tuttavia, qui non si è parlato. Per esempio, è possibile decidere di associare un certo grado di preferenza a certi insiemi di rotte, in modo del tutto arbitrario. Anche questo tipo di politiche influenza i percorsi utilizzati per l'invio dei dati: associare diversi gradi di preferenza (così come alterare altri attributi) ha, infatti, impatto sulle scelte operate dal processo decisionale.

1.2.4 Tabelle BGP

I router che utilizzano il protocollo BGP mantengono continuamente aggiornata, al loro interno, una tabella di instradamento che contiene informazioni specifiche del protocollo stesso. In alcuni casi viene messo a disposizione degli utenti un servizio di consultazione di queste tabelle per mezzo di interfacce web o di server telnet. Questo consente di accedere ad una panoramica più o meno integrale della visione che un router ha dell'intera rete Internet, e da qui nasce il nome di *looking glass server*[3], attribuito a router con questa funzionalità.

In realtà, la visione offerta da un looking glass non è necessariamente completa, poiché a sua volta viene influenzata dalla configurazione del router in esame o di quelli con cui fa peering. L'esempio più tipico al riguardo è costituito da un looking glass che abbia instaurato una sessione di peering con un altro router al solo fine di pubblicare i dati di instradamento noti a quest'ultimo: poiché il funzionamento di BGP prevede che vengano annunciate soltanto le alternative ritenute migliori per raggiungere ogni destinazione, il looking glass conoscerà soltanto queste ultime, perdendo una considerevole parte del contenuto informativo della RIB del router che sta monitorando.

Nonostante l'esistenza di numerosi sistemi operativi diversi e, corrispondentemente, di informazioni sull'instradamento organizzate in modo leggermente diverso, le tabelle BGP sono in genere costruite più o meno sempre rispettando uno stesso formato.

In figura 1.3 viene mostrato un esempio di connessione ad un looking glass di tipo telnet e la richiesta della relativa tabella BGP. È possibile osservare che, all'apertura della connessione, molti looking glass forniscono informazioni sui router con i quali hanno sessioni di peering attive. Immediatamente dopo, seguono i comandi `show clock` e `show ip bgp`, dei quali il primo mostra l'orario così come correntemente impostato sul router, ed il secondo richiede al router di mostrare l'intera RIB che ha immagazzinata al suo interno. Molti archivi di tabelle BGP sono costituiti proprio da file testuali generati impartendo al router il comando `show ip bgp`.

Nelle ultime righe della figura viene mostrato l'inizio della tabella BGP, ed è possibile distinguere i vari campi che ne compongono ogni riga:

- `Status code`: un insieme di attributi associati a ciascuna rotta; si noti come, tra di essi, figuri anche l'attributo *best*, con il quale vengono marcate le alternative ritenute migliori; per esempio, la destinazione 3.0.0.0 può essere raggiunta utilizzando lo stesso AS path ma passando da router (next hop) diversi, e la seconda delle due alternative è stata selezionata come migliore; se il router dovesse inoltrare un annuncio relativo a tale destinazione, comunicherebbe soltanto la seconda alternativa;

- `Network`: il prefisso del quale il record corrente mostra le caratteristiche; l'esistenza di un determinato prefisso in una tabella BGP indica che il router ha ricevuto un annuncio ad esso relativo, e che, quindi, sa come raggiungere gli indirizzi IP in esso compresi;

- `Next Hop`: l'indirizzo IP del router al quale inviare pacchetti destinati a macchine appartenenti alla Network del record corrente;

[3]Per un elenco abbastanza completo dei looking glass server esistenti, è possibile visitare i siti `http://www.caida.org/analysis/routing/reversetrace` oppure `http://www.traceroute.org`.

```
telnet> open route-server.west.attcanada.com
Trying 216.191.65.126...

Connected to route-server.west.attcanada.com.

Escape character is '^]'.

CC
########## route-server.west.attcanada.com ############
#######  AT&T Canada IP Services Route Monitor  #######

This router maintains peerings with the following
key routers of AT&T Canada IP Services Backbone:

 216.191.64.1   CORE-1,    Vancouver, BC, Canada
 216.191.64.2   CORE-2,    Vancouver, BC, Canada
 216.191.64.3   GATEWAY-1, Vancouver, BC, Canada
 216.191.64.4   GATEWAY-2, Vancouver, BC, Canada
 216.191.64.5   CORE-1,    Calgary,   AB, Canada
 216.191.64.6   CORE-2,    Calgary,   AB, Canada
 216.191.64.7   CORE-1,    Toronto,   ON, Canada
 216.191.64.8   CORE-2,    Toronto,   ON, Canada
 216.191.64.9   GATEWAY-1, Toronto,   ON, Canada
 216.191.64.10  GATEWAY-2, Toronto,   ON, Canada
 216.191.64.11  CORE-1,    Ottawa,    ON, Canada
 216.191.64.12  CORE-2,    Ottawa,    ON, Canada
 216.191.64.13  CORE-1,    Montreal,  QC, Canada
 216.191.64.14  CORE-2,    Montreal,  QC, Canada
 216.191.64.15  GATEWAY-1, Montreal,  QC, Canada
 216.191.64.16  GATEWAY-2, Montreal,  QC, Canada
 216.191.64.253 GATEWAY-1, Chicago,   IL, USA
 216.191.64.254 GATEWAY-1, Seattle,   WA, USA

This router supports the cisco "| [begin,exclude,include]" command format.
This is an unsupported service and may go away at any time

For questions about this route-server, send email to: routing@corp.attcanada.ca

#################### route-server.west.attcanada.com ####################

route-server.west>show clock
*09:59:07.182 UTC Fri Jan 10 2003
route-server.west>show ip bgp
BGP table version is 12574592, local router ID is 216.191.65.126
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
* i3.0.0.0          216.191.64.10                  60      0 701 7018 80 i
*>i                 216.191.64.4                   60      0 701 7018 80 i
* i4.0.0.0          216.191.64.10                  60      0 701 1 i
* i                 216.191.64.4                   60      0 701 1 i
* i                 216.191.64.9                   60      0 7018 1 i
*>i                 216.191.64.3                   60      0 7018 1 i
*>i5.0.0.0          216.191.66.18                 100      0 6401 i
* i                 216.191.66.18                 100      0 6401 i
*>i6.1.0.0/16       216.191.64.253                 80      0 7170 1455 i
*>i6.2.0.0/22       216.191.64.253                 80      0 7170 1455 i
*>i6.3.0.0/18       216.191.64.253                 80      0 7170 1455 i

... ... ... ...
```

Figura 1.3: Dialogo con un telnet looking glass server e frammento di tabella BGP

- `Metric`, `LocPrf`, `Weight`: queste metriche influenzano il modo in cui il router sceglie l'alternativa migliore per raggiungere una determinata rotta;
- `Path`: il percorso seguito dall'annuncio della Network corrente, in termini di Autonomous Systems attraversati; questo campo è costituito da una lista di numeri identificativi di Autonomous Systems, ordinati a partire da quello da cui è stato ricevuto l'annuncio per finire con quello che ha originato la rotta; se il router annuncia la stessa rotta all'esterno, prima di farlo inserisce il numero dell'Autonomous System in cui si trova in testa a questa lista; alcuni router funzionano in modo tale da fornire la tabella BGP già con il numero del proprio Autonomous System inserito in testa a tutti gli AS path.

Le tabelle BGP contengono molte indicazioni su come si stia comportando il protocollo in un dato istante, e costituiscono per questo la sorgente informativa primaria per analizzare i comportamenti di un Autonomous System nei confronti di un altro. La loro utilità diventa ancora più evidente se si considera che ciascun Autonomous System può essere sia fornitore che fruitore di servizi di connettività: in questo caso, i contenuti di una tabella BGP consentono di trarre delle conclusioni molto interessanti sul ruolo giocato da ciascuno di essi.

1.3 Rapporti tra fornitori e fruitori di servizi di connettività in Internet: relazioni tra Autonomous Systems

I fornitori/fruitori di servizi di connettività in Internet altro non sono che grosse organizzazioni che utilizzano od offrono la possibilità di accedere alla rete. I rapporti tra di essi sono generalmente rappresentati da accordi commerciali che vengono stipulati per amministrare opportunamente la gestione di questi accessi. Poiché le dimensioni delle organizzazioni coinvolte in questi accordi sono considerevoli, l'entità che meglio si identifica con un fornitore/fruitore di servizi di connettività è proprio l'Autonomous System. La sua stessa definizione (enunciata nella definizione 1.1) lo descrive come controllato da un unico organismo amministrativo, ed il rapporto che esso può avere con altri Autonomous Systems può essere proprio di utilizzo od offerta di un accesso ad Internet.

Per queste ragioni, è del tutto equivalente parlare di "rapporti tra fornitori e fruitori di servizi di connettività" o di "relazioni tra Autonomous Systems".

1.3.1 Aspetti economici e commerciali

Gli accordi commerciali che caratterizzano le relazioni tra Autonomous Systems possono essere motivati da diverse ragioni, e per questo assumere molte forme diverse. Inoltre, non sempre sono i soli aspetti economici a governare questi rapporti: esistono situazioni in cui le esigenze tecniche influenzano pesantemente le scelte effettuate (si pensi alla necessità di utilizzare i collegamenti ad alcuni Autonomous Systems soltanto in caso di guasto –backup–). Occorre poi considerare che l'esistenza di molte sfumature contrattuali può ostacolare una classificazione rigida delle relazioni, e che lo sviluppo più naturale di queste ultime consiste nell'evolversi nel tempo verso altre forme di relazione.

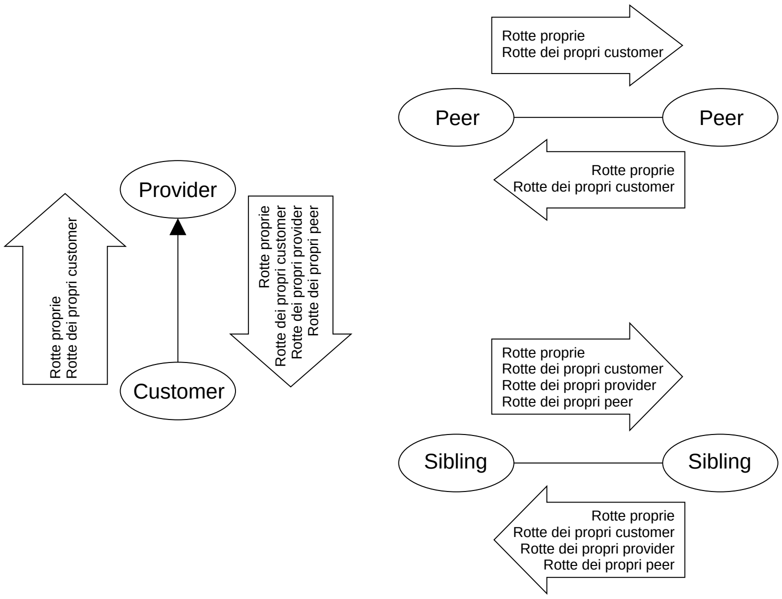

Figura 1.4: Politiche associate ai principali tipi di relazione tra Autonomous Systems

Nonostante la presenza di questi elementi di aleatorietà, la conoscenza, seppur incompleta, delle relazioni commerciali tra Autonomous Systems è un dato di grande interesse. Per esempio, in [8] si evidenzia come un nuovo fornitore di servizi di connettività (*ISP, Internet Service Provider*) a livello regionale che si volesse collegare con un ristretto numero di Autonomous Systems possa trarre vantaggio dal sapere quali tra di essi gli fornirebbero un servizio migliore. Sempre in [8] si cita l'esempio di un'azienda che distribuisce contenuti informativi per conto terzi, la quale trarrebbe giovamento da queste informazioni ai fini della collocazione dei propri server web.

1.3.2 Tipi di relazione

È decisamente improbabile che si possa predisporre uno schema preciso nel quale inquadrare tutti i tipi di relazione esistenti. Tuttavia, per poter usufruire di un riferimento e comprendere meglio quali tipi di rapporti interessa individuare, vengono qui presentati gli accordi commerciali di maggior rilievo. Nelle situazioni reali si incontrano, generalmente, delle combinazioni degli accordi qui elencati, sebbene una delle forme descritte risulti in genere prevalente.

A ciascun tipo di relazione corrisponde in genere una opportuna configurazione dei border router degli Autonomous Systems coinvolti, che fa sì che il traffico si sviluppi soltanto secondo i percorsi consentiti dal reciproco accordo. Per ogni relazione si cerca, dunque, di mostrare quali potrebbero (o dovrebbero) essere le politiche adottate dagli Autonomous Systems in gioco.

Customer-provider Si trovano in questa situazione due Autonomous Systems dei quali uno stia vendendo il servizio di accesso ad Internet ad un altro. Quest'ultimo a sua volta paga per ottenere tale servizio. Questo tipo di relazione viene anche chiamato in [10, 12] di *transito*, poiché effettivamente il provider accetta, dietro opportuno compenso, di far transitare traffico proveniente dai propri customer; il viceversa non è ammesso.

Le politiche adottate in un siffatto rapporto prevedono, in linea di massima, i seguenti comportamenti (mostrati anche in figura 1.4):

- provider: esporta ai propri customer tutte le rotte che conosce, poiché deve fornire loro connettività a tutto il resto di Internet; quindi, annuncia le proprie rotte e quelle che gli sono state annunciate dai propri altri customer, provider e peer;
- customer: esporta al provider soltanto lo stretto indispensabile per rendere visibile agli altri sé stesso ed i propri customer senza dover inoltrare traffico per altri; quindi, annuncia le proprie rotte e quelle dei[4] propri customer.

In [12], ma anche in altri articoli in cui si cerca di individuare una gerarchia di Autonomous Systems ([8, 13]), si sottolinea il ruolo dei così detti ISP di livello 1, ovvero dei provider che risultano essere al livello più alto nella gerarchia: secondo [12], un ISP di livello 1, pur avendo accesso a tutte le destinazioni nella rete Internet, non acquista il servizio di connettività da nessuno, ma lo offre soltanto. Questo tende a scoraggiare questi provider dall'instaurare rapporti di peering, poiché non ne trarrebbero alcun vantaggio economico.

Peering Un accordo di peering[5] ha luogo quando due Autonomous Systems accettano di scambiarsi il traffico dei propri rispettivi customer senza alcuna forma di ritorno economico.

Evidentemente, una simile relazione può essere instaurata solo se i due Autonomous Systems in gioco non sono in alcun modo in competizione, e le loro importanze (misurabili, ad esempio, in termini di traffico generato) sono paragonabili. Se così non fosse, uno di essi trarrebbe molto più vantaggio dall'accordo rispetto all'altro partecipante, il quale potrebbe addirittura ottenerne delle perdite. A fronte dei vantaggi portati da una relazione di peering in termini di banda disponibile e di riduzione delle spese, in [12] si mostrano anche alcuni aspetti che tipicamente ostacolano il raggiungimento di un accordo del genere: il caso di due provider che si scambiano traffico in misura fortemente asimmetrica, il fatto che un peering rende il proprio interlocutore più competitivo (e quindi più temibile) e l'impiego di risorse che in rapporti customer-provider avrebbe dato luogo ad un riscontro economico. Per questo esistono anche delle forme "ibride" di peering, nelle quali le parti in gioco non si aspettano un tornaconto solo finché lo sbilanciamento nel traffico da esse generato non superi una certa soglia.

Ad un accordo di peering corrispondono generalmente le seguenti politiche:

- peer 1: esporta ai propri peer le rotte necessarie per far dialogare i propri customer con quelli dell'altro peer; quindi, esporta le proprie rotte e quelle annunciategli dai propri customer;

[4] Nel seguito, dato un Autonomous System, si indicheranno come rotte *dei* suoi customer, provider, peer e sibling le rotte che da essi gli sono state annunciate.

[5] Il nome di *peering* associato a questo rapporto non ha nulla a che fare con il concetto di sessione di peering BGP, e per questo non va con esso confuso. A seconda del contesto, dovrebbe risultare chiaro se si stia facendo riferimento al rapporto commerciale o alla sessione TCP.

- peer 2: si comporta in modo identico al peer 1.

In [14] si caratterizza la relazione di peering in base alle aspettative delle parti in gioco: se entrambi non considerano come atto competitivo la cessazione del rapporto da parte di uno dei partecipanti, allora la relazione può effettivamente considerarsi di peering. Inoltre, si sottolinea anche come uno stesso Autonomous System possa instaurare più relazioni diverse con altri Autonomous Systems, sebbene il rapporto esistente nell'ambito di ciascuna coppia possa quasi sempre considerarsi di un unico tipo.

Infine, in [15] si descrive la tendenza spontanea dei customer ad evolvere verso la condizione di peer (per svincolare il proprio traffico da un'interconnessione a pagamento) e, successivamente, verso quella di provider (qualora il rapporto tra le parti non si potesse più considerare paritario). Questo sottolinea l'importanza del problema di tracciare l'evoluzione delle relazioni tra Autonomous Systems.

Sibling Sul rapporto di sibling esiste una quantità estremamente ridotta di documentazione, probabilmente per via del suo carattere prevalentemente tecnico piuttosto che commerciale.

In genere, si classifica come sibling una configurazione in cui due Autonomous Systems inoltrino traffico l'uno per l'altro. A differenza del rapporto di peering, in questo caso i due partecipanti si scambiano dati verso tutte le destinazioni, e non soltanto verso i rispettivi customer. In [10, 16] la relazione viene anche chiamata di *transito reciproco*, proprio perché gli Autonomous Systems si forniscono reciprocamente connettività a tutto il resto di Internet.

Dalla stessa definizione di sibling dovrebbe risultare abbastanza chiaro che le politiche adottate siano le seguenti:

- sibling 1: esporta ai propri sibling tutte le rotte, per garantire ad essi lo stesso livello di connettività di cui al momento usufruisce; quindi, esporta le proprie rotte e quelle apprese dai propri customer, provider e peer;

- sibling 2: si comporta esattamente come il sibling 1.

Tipicamente, questo tipo di rapporto si stabilisce quando due organizzazioni con competenze estremamente simili hanno bisogno del supporto l'una dell'altra prima di poter gestire una rete propria. Per esempio, questo può succedere in ambito accademico. La necessità che i partecipanti abbiano delle competenze simili traspare anche da [16], in cui si propone un metodo per la verifica dei rapporti di sibling basato sulla prossimità geografica degli Autonomous Systems coinvolti.

Backup Questo tipo di relazione esula quasi del tutto dalle esigenze di carattere commerciale che governano gli altri rapporti. Tuttavia, è opportuno includerlo in questo elenco perché di fatto presente nella rete e caratterizzato anch'esso da particolari politiche.

È possibile fornire diversi esempi di configurazione che possa considerarsi come contenente elementi "di backup". In genere, comunque, si considera in tale rapporto una coppia di Autonomous Systems collegati da un link che venga utilizzato solo (o molto più pesantemente del solito) quando si verifichi un guasto di uno o più altri link utilizzati abitualmente.

Le politiche associate a questo tipo di relazione variano a seconda dei criteri desiderati per il funzionamento delle strutture di backup, nonché della stessa definizione di rapporto di backup; di conseguenza, non è possibile fornirne una caratterizzazione universale. L'individuazione di link di backup si può, comunque, basare sulla ricerca di configurazioni particolari, dalle quali appaia in modo piuttosto evidente che tali link vengono utilizzati solo secondariamente per il transito di pacchetti.

1.3.3 Il problema dell'inferenza delle relazioni

Lo studio dell'evoluzione delle relazioni tra Autonomous Systems sarebbe decisamente semplice se ci fosse la possibilità di conoscere la cronologia dei rapporti commerciali esistiti tra ciascuna coppia di Autonomous Systems. Naturalmente, questo non è possibile, poiché i contratti stipulati non sono disponibili pubblicamente e, d'altra parte, la loro consultazione richiederebbe un tempo eccessivo; è necessario quindi ricorrere alle notizie disponibili sulla rete per ricostruire i vari accordi esistenti in ogni istante. Alcuni registri (*IRR, Internet Routing Registries*) consultabili liberamente su Internet consentono di esaminare le politiche di importazione ed esportazione degli annunci adottate da alcuni Autonomous Systems. Un buon numero tra questi ultimi, tuttavia, non pubblica informazioni sulle proprie politiche, per lo stesso motivo per cui mantiene riservati i propri contratti.

Di fatto, l'unica strada che sembra essere percorribile per studiare le relazioni tra Autonomous Systems e, quindi, la loro evoluzione, è quella di ricavarle mediante opportuni algoritmi di inferenza. I dati presenti nelle tabelle di routing BGP costituiscono a tal fine un'ottima fonte informativa, poiché da essi si può dedurre in modo abbastanza fedele alla realtà quali siano le politiche adottate dagli Autonomous Systems, e, quindi, anche il tipo di relazione che essi hanno instaurato.

1.3.4 Considerazioni sull'affidabilità e la stabilità del protocollo BGP

Il ruolo giocato dai link tra border router di Autonomous Systems diversi è evidentemente fondamentale per consentire la comunicazione tra di essi. Per questo motivo, non è assolutamente pensabile che le configurazioni reali prevedano di connettere un Autonomous System agli altri utilizzando un solo link (*stub networks*).

Molto spesso, invece, un Autonomous System adotta politiche che sfruttano più link per comunicare con i suoi vicini. Queste politiche possono essere volte ad ottenere il bilanciamento del carico sui vari canali trasmissivi, per realizzare meccanismi di *load sharing*, oppure, altrettanto frequentemente, ad utilizzare determinati link come primari, riservandone altri all'utilizzo in caso di guasto.

In particolare, la gestione di quest'ultimo tipo di comportamento solleva dei problemi relativi all'efficacia del funzionamento del protocollo BGP. In [7, 17, 18, 19] si introduce la definizione di istanza di *Stable Paths Problem*, un problema di carattere statico volto a catturare le proprietà di convergenza del protocollo senza la necessità di conoscerne i comportamenti specifici. Più precisamente, lo *Stable Paths Problem* è un problema su grafi, che analizza le scelte fatte da ciascun nodo (vincolate in un insieme predefinito di cammini e guidate da una funzione di preferenza) per raggiungere un particolare nodo di destinazione. Si può mostrare ([7]) che esistono istanze di questo problema in cui alcuni nodi oscillano permanentemente tra più scelte; una di queste situazioni è rappresentata in figura 1.5.

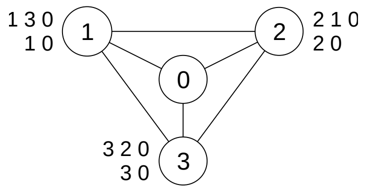

Figura 1.5: Istanza non risolvibile di Stable Paths Problem

In essa viene mostrata una configurazione in cui i nodi 1, 2 e 3 vogliano trovare un percorso per raggiungere il nodo 0, scegliendo esclusivamente tra i percorsi per essi ammissibili; questi ultimi sono elencati a lato di ogni nodo in ordine di preferenza. Ogni nodo è vincolato ad operare scelte coerenti con quelle fatte dal suo successore. Inizialmente, il nodo 1 (oppure un altro qualsiasi dei nodi) seleziona il percorso 1 3 0; di conseguenza, il nodo 2 deve rinunciare al cammino che ritiene migliore (2 1 0) e scegliere invece 2 0. A questo punto, il nodo 3 può optare per il cammino preferito 3 2 0, ma in questo modo costringe il nodo 1 a rivedere la propria scelta (perché non più coerente con quella operata dal nodo 3), selezionando 1 0. Questo consente al nodo 2 di optare per il cammino con più alto grado di preferenza (2 1 0), il che, però, forza il nodo 3 a cambiare a sua volta la selezione effettuata.

Risulta dunque evidente come in questa istanza non vi sia modo, per alcun nodo, di mantenere stabilmente una scelta. Come mostrato in [19], una istanza di configurazione BGP può essere tradotta in una corrispondente istanza di Stable Paths Problem, il che consente di verificare se l'introduzione di determinate politiche porti o meno a situazioni di instabilità come quella descritta. Purtroppo, in [17] si dimostra che il problema di stabilire se un'istanza di Stable Paths Problem sia o meno risolvibile è NP-completo. Tuttavia, [19] propone un algoritmo (*Safe Path Vector Protocol*) che elimina progressivamente dei cammini ammissibili fino a che non si possano più manifestare oscillazioni, mentre in [17] si dimostra una condizione sufficiente perché un'istanza sia risolvibile. Infine, sono molto interessanti le considerazioni fatte in [7, 18], in cui si dimostra che se le politiche BGP obbediscono ad un certo insieme di linee guida, la configurazione che ne risulta è *Inherently Safe*, cioè non può mai divergere, qualunque siano le scelte fatte inizialmente da ciascun nodo e qualunque combinazione di guasti (modellati con la cancellazione di nodi od archi) si verifichi.

1.4 Evoluzione delle relazioni tra Autonomous Systems

1.4.1 Obiettivi dell'analisi

Le finalità dello studio dell'evoluzione delle relazioni non consistono nel tracciamento di un profilo del comportamento assunto da ciascun Autonomous

System, bensì nell'esame di aspetti più tecnici riguardanti il protocollo BGP ed il funzionamento dell'algoritmo di inferenza.

In particolare, si cercano di inquadrare due aspetti fondamentali:

- il grado di coerenza con cui le relazioni inferite seguono le variazioni che si sono verificate negli effettivi accordi commerciali;
- la qualità delle relazioni inferite, in termini di stabilità dei risultati prodotti dall'algoritmo su periodi nei quali non si sono verificati grossi cambiamenti dal punto di vista commerciale.

Una volta valutati questi elementi, si può attribuire all'algoritmo di calcolo delle relazioni un livello più o meno alto di validità anche nella dimensione temporale, il che conferisce ai risultati da esso prodotti un valore aggiunto notevole per chi abbia bisogno di basare le proprie strategie commerciali sulla conoscenza dell'evoluzione delle relazioni.

1.4.2 Requisiti preliminari alla definizione di una metodologia

Dalle considerazioni fatte in precedenza emerge la necessità di calcolare le relazioni tra Autonomous Systems mediante opportuni algoritmi di inferenza.

Affinché si possa analizzare l'evoluzione delle relazioni, poi, diventa estremamente importante che tutti gli strumenti a disposizione siano strutturati in modo il più possibile sistematico, rispettando almeno le seguenti caratteristiche:

- l'algoritmo di inferenza delle relazioni deve essere efficace e veloce; in altre parole, deve consentire il calcolo sulla base di informazioni disponibili su ampi intervalli di tempo (altrimenti non sarebbe possibile parlare di evoluzione), la cui organizzazione si mantenga sempre coerente e stabile (non avrebbe senso effettuare calcoli su dati che forniscono informazioni diverse da periodo a periodo); inoltre, deve permettere la determinazione di un numero elevato di relazioni in un tempo il più possibile ridotto: lo studio dell'evoluzione richiede, infatti, l'elaborazione di notevoli quantità di dati, il che può comportare periodi di attesa molto prolungati; la disponibilità di un algoritmo veloce consente, invece, di calcolare lo stesso numero di relazioni in minor tempo (potendole quindi poi analizzare più a fondo) o un numero più elevato di relazioni nello stesso tempo (potendo quindi prendere in esame l'evoluzione in intervalli più ampi);
- gli strumenti utilizzati per analizzare le relazioni calcolate devono presentare delle caratteristiche di funzionamento chiare e ben definite; questo significa che per ciascun passo dell'analisi deve essere possibile delineare in modo preciso quali siano i dati in gioco e le elaborazioni su di essi effettuate; diversamente, si rischia di produrre dei risultati poco coerenti e di scarso interesse.

Le operazioni necessarie per garantire queste proprietà si possono suddividere in due fasi fondamentali: la fase di messa a punto di un ambiente per l'analisi dell'evoluzione delle relazioni (presentata nel capitolo 3) e quella di ottimizzazione di un algoritmo esistente, utilizzato per inferirle (descritta nel capitolo 4).

Capitolo 2

Inferenza delle relazioni tra Autonomous Systems: stato dell'arte

Il problema di trovare un algoritmo per inferire in modo automatico le relazioni tra Autonomous Systems è stato recentemente studiato utilizzando più approcci diversi.

Il primo tentativo vero e proprio di formalizzazione e di ricerca di una soluzione risale al novembre del 2000, ed è descritto in [10]. Lo studio è basato su di un'euristica che stabilisce che un Autonomous System sta fornendo un servizio di transito ad un altro sulla base del loro grado, e successivamente attribuisce le varie relazioni a seconda dello sbilanciamento tra fruizione ed offerta di tale servizio.

Quasi un anno dopo (aprile 2001) vengono proposte in [8] delle regole di inferenza di carattere statistico/probabilistico, che prendono in considerazione le informazioni provenienti da diversi looking glass. L'idea è quella di incrociare i dati di punti di osservazione diversi per ottenere dei risultati più attendibili.

Gli sviluppi più recenti sono stati presentati quasi simultaneamente nel 2002 in [20] e [21]. Nel primo si opera una riduzione dal problema di associare delle relazioni alle coppie di Autonomous Systems a quello di assegnare dei valori di verità a variabili che appaiono in formule proposizionali; l'obiettivo è quello di verificare se è possibile assegnare le relazioni limitando (o addirittura evitando) l'introduzione di situazioni "strane" (per esempio, un customer che consenta il transito di traffico tra due suoi provider). Nel secondo articolo si utilizza lo stesso procedimento, ma si pone più enfasi sulla qualità delle soluzioni che è possibile ottenere in tempo polinomiale.

Nei paragrafi che seguono, le varie soluzioni vengono presentate coerentemente con l'ordine temporale con cui sono state elaborate.

2.1 Formalizzazione del problema dell'individuazione delle relazioni tra Autonomous Systems

Ciascuno degli approcci proposti per risolvere il problema in esame ha seguito strade piuttosto differenti dagli altri, talvolta operando assunzioni particolari od utilizzando dati di provenienza diversa. Tutti però sono stati più o meno concordi nell'inquadrare il problema in un'unica formulazione, sempre basata su dati ottenuti da tabelle BGP. Per poter enunciare tale formulazione, è necessario prima introdurre alcuni concetti preliminari.

2.1.1 Grafo delle relazioni e validità di un cammino

Innanzitutto, è opportuno precisare che i cammini cui si fa riferimento in questa sede altro non sono che gli AS path estrapolati dalle tabelle BGP. Ciò premesso, l'altro elemento fondamentale che compare nella formulazione di questo problema è costituito da un grafo non orientato. Tale grafo rappresenta la connettività logica esistente tra gli Autonomous Systems (nel seguito AS), di cui si è parlato nella sezione 1.2.2, ed è fatto nel seguente modo:

Definizione 2.1 (Grafo degli AS) *Sia $G(N, E)$ un grafo non orientato. G è un* grafo degli AS *se è così costituito:*

- $N = \{n \mid 1 \leq n \leq 65535\}$ *è un insieme di numeri identificativi di Autonomous Systems;*
- $E = \{(v_1, v_2) \mid v_1, v_2 \in N \text{ e } (v_1, v_2) \in E \Rightarrow v_1 \leftrightarrows v_2\}$*, dove* $\leftrightarrows$ *vuol dire che* v_1 *e* v_2 *hanno una sessione di peering attiva.*

Si noti come avere una sessione di peering attiva sia solo una condizione necessaria, e non sufficiente, perché ci sia un arco nel grafo degli AS. Questo è dovuto al fatto che un grafo degli AS viene generalmente costruito sulla base delle adiacenze trovate negli AS path estratti da una tabella BGP; in quanto tale, la presenza di un arco indica che c'è stato l'annuncio di almeno una rotta tra i due AS alle estremità, mentre è possibile che l'esistenza di alcuni annunci non sia visibile dalla tabella BGP che si sta usando, e dunque che alcuni archi non vengano inseriti pur esistendo le corrispondenti sessioni di peering.

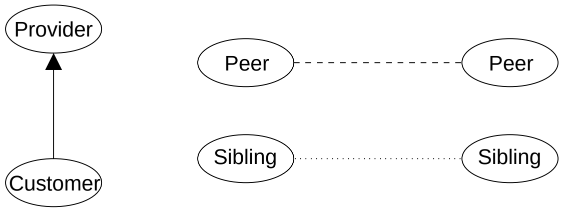

Figura 2.1: Convenzioni adottate per rappresentare le relazioni tra AS

Dal grafo degli AS è poi possibile, orientandone opportunamente gli archi, passare al *grafo delle relazioni tra AS*. In particolare, si suppone di adottare la convenzione riportata in figura 2.1, secondo cui in una coppia customer-provider l'arco si suppone orientato verso il provider. Per rappresentare relazioni di peering o sibling, si suppone di "colorare" gli archi con un opportuno tratteggio, anch'esso riportato nella figura.

Per poter parlare di validità di un cammino bisogna supporre che le relazioni tra Autonomous Systems siano già state fissate. Il concetto di validità si riferisce, infatti, alla correttezza o meno di una catena di relazioni osservate lungo un AS path.

Definizione 2.2 (Cammino valido) *Si consideri un cammino p costituito da una sequenza di vertici $(v_1, v_2, \ldots, v_n)$; si dice che p è un* cammino valido *se rientra in una delle seguenti due categorie:*

> Tipo 1*: p è costituito da una sequenza (eventualmente vuota) di archi customer→provider seguita da una sequenza (eventualmente vuota) di archi provider←customer; in altre parole,*
>
> $$\exists v_i \in p \mid \begin{cases} (v_j \rightarrow v_{j+1}) \in E, & \forall j \in \{1, \ldots, i-1\} \\ (v_j \leftarrow v_{j+1}) \in E, & \forall j \in \{i, \ldots, n-1\} \end{cases}$$
>
> Tipo 2*: p è costituito da una sequenza (eventualmente vuota) di archi customer→provider, seguita da un arco di peering, seguito da una sequenza (eventualmente vuota) di archi provider←customer; in altre parole,*
>
> $$\exists v_i \in p \mid \begin{cases} (v_j \rightarrow v_{j+1}) \in E, \ \forall j \in \{1, \ldots, i-1\} \\ \quad (v_i \text{---} v_{i+1}) \in E \\ (v_j \leftarrow v_{j+1}) \in E, \ \forall j \in \{i+1, \ldots, n-1\} \end{cases}$$

Qualsiasi cammino che non rientri nelle due categorie descritte sopra si dice *non valido*. Esempi di cammini non validi sono riportati nella figura 2.4. Si noti come affermare che un cammino sia valido corrisponda a presupporre che gli annunci che lo hanno seguito abbiano incontrato router configurati secondo le politiche "standard" descritte nella sezione 1.3.2. Di converso, se ogni AS rispetta tali politiche, tutti gli AS path sono di tipo 1 oppure di tipo 2. In effetti, tutti i cammini nella figura 2.4 violano, almeno una volta, queste politiche standard.

2.1.2 Formulazione del problema

Sebbene [10] non enunci in modo rigoroso il problema, anche la trattazione in esso riportata si può ricondurre alle formulazioni proposte in [8] e [20]. Si può dunque assumere che la formulazione comunemente adottata in letteratura sia la seguente:

Problema 2.1 (ToR[1]) *Dato un grafo non orientato G ed un insieme di cammini P, orientare alcuni degli archi di G in modo tale da minimizzare il numero di cammini non validi contenuti in P.*

[1] *Type Of Relationship*

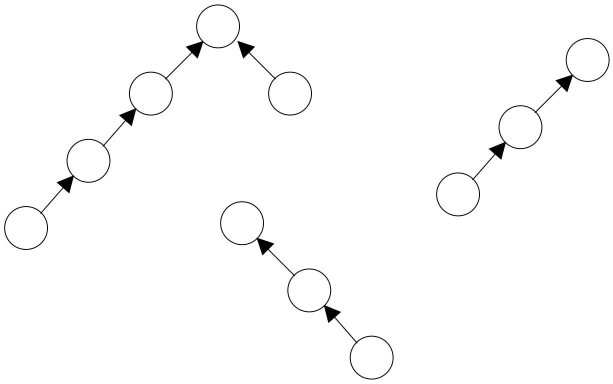

Figura 2.2: Esempi di cammini di tipo 1

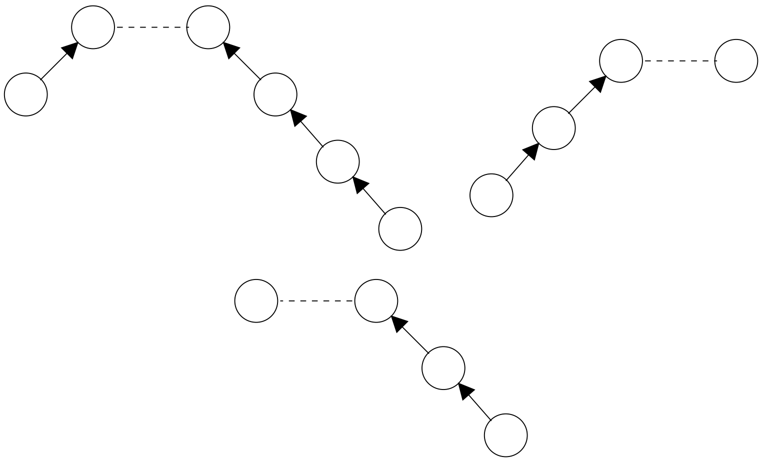

Figura 2.3: Esempi di cammini di tipo 2

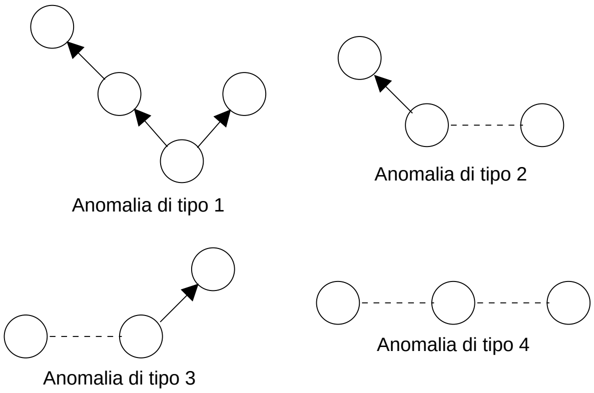

Figura 2.4: Esempi di cammini non validi

L'attribuzione di un'orientazione ad un arco corrisponde, ovviamente, all'individuazione della relazione che lega gli Autonomous Systems alle sue estremità.

A partire da questo enunciato, i vari autori hanno poi proposto diversi metodi per risolvere il problema, nell'ambito dei quali, pur mantenendo abbastanza coerenti le assunzioni di base, hanno seguito approcci piuttosto marcatamente differenziati.

2.2 Approccio basato sul grado degli Autonomous Systems

Il primo studio rigoroso sul problema è stato presentato nel novembre del 2000 in [10]. In esso si propone un algoritmo che implementa un'euristica basata su due fasi fondamentali: l'individuazione delle relazioni di transito e l'inferenza delle relazioni vere e proprie. Il tutto, basato sul grado di ogni Autonomous System.

Nell'ambito della teoria dei grafi si definisce *grado* di un nodo il numero di archi su di esso incidenti. In questo contesto tale definizione viene conservata appieno, sicché il grado di un Autonomous System è il numero di archi che incidono su di esso nel grafo degli AS.

2.2.1 Relazioni considerate e politiche

Nell'intento di fornire una panoramica completa delle relazioni esistenti tra Autonomous Systems, la soluzione proposta prende in considerazione tutti i rapporti descritti nella sezione 1.3.2, ad eccezione di quello di backup, che viene un anno dopo trattato separatamente dallo stesso autore in [7]. Per quello che riguarda le politiche adottate per realizzare questi rapporti, in [10] si tende a sottolineare che i vari comportamenti descritti sono quelli idealmente corretti, ma non si può essere certi che siano effettivamente adottati nelle configurazioni dei router. Le politiche supposte utilizzate sono comunque le seguenti:

- *Esportazione ad un provider*: un AS esporta ai suoi provider le proprie rotte e quelle dei propri customer, ma solitamente non quelle dei propri provider e peer;
- *Esportazione ad un customer*: un AS esporta ai suoi customer le proprie rotte e quelle dei propri customer, provider e peer;
- *Esportazione ad un peer*: un AS esporta ai suoi peer le proprie rotte e quelle dei propri customer, ma solitamente non quelle dei propri provider e peer;
- *Esportazione ad un sibling*: un AS esporta ai suoi sibling le proprie rotte e quelle dei propri customer, provider e peer.

Accanto alla definizione di queste politiche, in [10] si associano ad ogni relazione anche dei comportamenti in termini di servizi di *transito* delle informazioni, nel modo che segue:

- *peering*: due AS u e v sono in una relazione di peering[2] se e solo se u non inoltra traffico per v né v inoltra traffico per u;

- *customer-provider*: un AS u è un provider di un AS v se e solo se u inoltra traffico per v e v non inoltra traffico per u;

- *sibling*: due AS u e v si comportano come sibling se e solo se u inoltra traffico per v e v inoltra traffico per u.

2.2.2 Algoritmo utilizzato

Riprendendo in considerazione la definizione di cammino valido, in [10] si introduce la nozione supplementare di *cammino valley-free*:

Definizione 2.3 (Cammino valley-free) *Un AS path $(u_1, u_2, \ldots, u_n)$ si dice* valley-free *se e solo se sono vere le seguenti condizioni:*

- *se (u_i, u_{i+1}) è un arco provider-customer, allora (u_j, u_{j+1}) deve essere un arco provider-customer oppure sibling-sibling, per ogni $i < j < n$;*

- *se (u_i, u_{i+1}) è un arco peer-peer, allora (u_j, u_{j+1}) deve essere un arco provider-customer oppure sibling-sibling, per ogni $i < j < n$.*

Si noti come queste condizioni corrispondano ad evitare l'insorgere di condizioni di invalidità dei cammini come quelle rappresentate nella figura 2.4.

L'algoritmo proposto per l'individuazione delle relazioni cerca dunque di assegnare queste ultime garantendo la condizione valley-free per ogni cammino. Per fare questo si basa su diverse intuizioni: la prima è che, generalmente, la dimensione di un Autonomous System è proporzionale al suo grado nel grafo degli AS, e per questo l'approccio qui presentato si considera basato sul grado degli AS; la seconda è che le dimensioni dei provider siano superiori rispetto a quelle dei loro customer. In particolare, quest'ultima supposizione porta ad individuare, per ogni cammino, un Autonomous System speciale, chiamato *top provider*, che si ritiene al vertice di una catena di archi customer$\rightarrow$provider seguiti da archi provider$\leftarrow$customer. Questo Autonomous System viene individuato proprio sulla base del grado, e l'algoritmo di inferenza delle relazioni fa in modo che in ogni cammino si trovi collocato come rappresentato in figura 2.5.

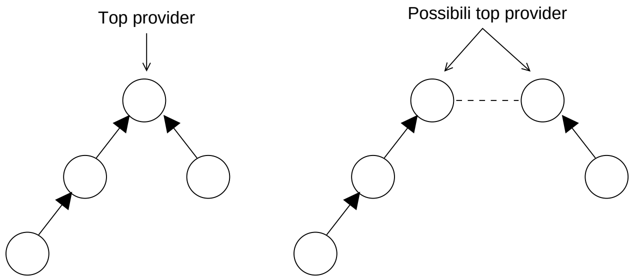

Figura 2.5: Individuazione del *top provider*

L'idea alla base dell'algoritmo (di cui viene riportata una versione completa in figura 2.6) è quella di determinare innanzitutto la posizione del top provider,

[2] In senso commerciale.

Fase 1: calcolo del grado degli AS

Per ogni AS path $p = (u_1, u_2, \ldots, u_n)$
  Per ogni i in $\{1, \ldots, n-1\}$
    $vicini\,[u_i] = vicini\,[u_i] \cup \{u_{i+1}\}$
    $vicini\,[u_{i+1}] = vicini\,[u_{i+1}] \cup \{u_i\}$
Per ogni AS u
  $grado\,[u] = |vicini\,[u]|$

Fase 2: inferenza delle relazioni di transito

Per ogni AS path $p = (u_1, u_2, \ldots, u_n)$
  trovare il più piccolo j tale che $grado\,[u_j] = \max_{1 \leq i \leq n}\,(grado\,[u_i])$
  Per ogni i in $\{1, \ldots, j-1\}$
    $transito\,[u_i, u_{i+1}] = transito\,[u_i, u_{i+1}] + 1$
  Per ogni i in $\{j, \ldots, n-1\}$
    $transito\,[u_{i+1}, u_i] = transito\,[u_{i+1}, u_i] + 1$

Fase 3: assegnazione delle relazioni alle coppie di AS

Per ogni AS path $p = (u_1, u_2, \ldots, u_n)$
  Per ogni i in $\{1, \ldots, n-1\}$
    se $(transito\,[u_{i+1}, u_i] > L$ e $transito\,[u_i, u_{i+1}] > L)$
    oppure $(0 < transito\,[u_i, u_{i+1}] \leq L$ e $0 < transito\,[u_{i+1}, u_i] \leq L)$
      $relazione\,[u_i, u_{i+1}] =$ sibling-sibling
    altrimenti se $transito\,[u_{i+1}, u_i] > L$ o $transito\,[u_i, u_{i+1}] = 0$
      $relazione\,[u_i, u_{i+1}] =$ provider-customer
    altrimenti se $transito\,[u_i, u_{i+1}] > L$ o $transito\,[u_{i+1}, u_i] = 0$
      $relazione\,[u_i, u_{i+1}] =$ customer-provider

Figura 2.6: Algoritmo che implementa l'euristica proposta in [10]

per poi assegnare alle coppie che lo precedono e lo seguono in ogni AS path le relazioni provider-customer o sibling-sibling. In particolare, queste operazioni si svolgono in 3 fasi fondamentali:

1. per prima cosa, viene calcolato il grado degli AS; per fare questo, si costruisce, per ogni AS, l'insieme di quelli che si trovano adiacenti ad esso in almeno un AS path; la cardinalità di questo insieme fornisce, evidentemente, il numero di AS vicini a quello considerato, ovvero il suo grado nel grafo degli AS;

2. successivamente, si marca come top provider di ogni AS path il primo AS che abbia grado pari al grado massimo tra quelli degli AS che compaiono nel path; a questo punto, si suppone che tutti gli AS che precedono e seguono il top provider stiano gerarchicamente usufruendo dei suoi servizi; in altre parole, in ciascuna coppia di AS si assume che il più vicino al top provider fornisca servizio di transito al più lontano;

3. infine, si assegnano le relazioni alle varie coppie di AS, sulla base di quale sia lo squilibrio nel transito offerto nei due versi; in particolare, si

inferisce la presenza di una coppia di sibling nel caso in cui (passo 3) i due AS considerati risultino offrirsi reciprocamente servizio di transito in modo evidente ($transito[\ldots] > L$) o in almeno uno dei cammini analizzati ($0 < transito[\ldots] \leq L$); si inferisce invece l'esistenza di una relazione provider-customer quando (passi 5, 7) un solo AS stia evidentemente offrendo servizio di transito ad un altro ($transito[\ldots] > L$) oppure quest'ultimo non risulti offrirlo al primo ($transito[\ldots] = 0$).[3]

Questa procedura consente l'assegnazione delle sole relazioni customer-provider e sibling-sibling. Una parte successiva dell'algoritmo (descritta nella figura 2.7) si basa sulle informazioni calcolate fino a questo punto per arricchire il grafo degli AS con le relazioni peer-peer.

Questa seconda parte procede escludendo preliminarmente le coppie di AS per le quali è improbabile la presenza di un rapporto di peering. L'assunzione che viene fatta è che questa esclusione abbia senso quando vi sia offerta di transito in uno (o entrambi) dei due versi (e ciò, per come vengono ricavate le relazioni di transito, esclude anche che vi possa essere un peering che non coinvolga il top provider), oppure quando le dimensioni degli AS coinvolti differiscano di molto (una relazione di peering in genere ha luogo tra due AS di dimensioni paragonabili). Queste ipotesi garantiscono ancora che tutti i cammini analizzati si mantengano valley-free.

I rapporti di peering vengono inferiti nel seguente modo:

1. prima di tutto, viene eseguita la parte di algoritmo finalizzata a calcolare le relazioni provider-customer e sibling-sibling;

2. in secondo luogo, si localizza ancora una volta in ogni AS path la posizione del top provider, procedendo esattamente come descritto in precedenza, e si marcano tutte le coppie di AS come non candidate ad avere un rapporto di peering, ad eccezione delle due coppie formate con il top provider stesso ($[u_{j-1}, u_j]$ e $[u_j, u_{j+1}]$); i due AS ad esso adiacenti sono a questo punto gli unici candidati rimasti; se nessuno dei due ha instaurato un sibling con il top provider, allora le loro dimensioni vengono comparate, e si esclude che quest'ultimo possa avere un peering con il più piccolo tra i due; questo comportamento rispecchia l'assunzione che un top provider faccia più probabilmente peering con il suo vicino di grado più alto;

3. come ultima operazione, si analizzano i link che ancora sono potenzialmente accettabili come peer-peer, e si attribuisce loro tale relazione nel caso in cui il grado degli AS alle loro estremità non differisca troppo.

L'intero algoritmo risulta essere parametrico rispetto a due valori di soglia che ne possono influenzare il comportamento. Tali valori possono essere introdotti grazie al fatto che a ciascuna relazione di transito viene attribuito un peso, rappresentato dal numero di AS path nei quali si è manifestata.

Il primo parametro è il valore L, che rappresenta la quantità oltre la quale una relazione di transito solamente supposta si assume effettivamente esistente;

[3] Si osservi come, a causa delle operazioni effettuate nella fase 2, non possa risultare contemporaneamente $transito[u_i, u_{i+1}] = 0$ e $transito[u_{i+1}, u_i] = 0$, ma debba per forza valere $transito[u_i, u_{i+1}] = 0 \iff transito[u_{i+1}, u_i] \neq 0$; alla luce di questo, se risulta che un AS non sta offrendo servizio di transito ad un altro, necessariamente quest'ultimo lo sta offrendo al primo e viceversa.

Fase 1: utilizzo dell'algoritmo in figura 2.6 per individuare i rapporti provider-customer e sibling-sibling

Fase 2: individuazione delle coppie di AS che non possono avere una relazione di peering

1. Per ogni AS path $p = (u_1, u_2, \ldots, u_n)$
2. trovare l'AS u_j tale che $grado\,[u_j] = \max_{1 \leq i \leq n} (grado\,[u_i])$
3. Per ogni i in $\{1, \ldots, j-2\}$
4. $nopeering\,[u_i, u_{i+1}] = 1$
5. Per ogni i in $\{j+1, \ldots, n-1\}$
6. $nopeering\,[u_i, u_{i+1}] = 1$
7. se $relazione\,[u_{j-1}, u_j] \neq$ sibling-sibling e $relazione\,[u_j, u_{j+1}] \neq$ sibling-sibling
8. se $grado\,[u_{j-1}] > grado\,[u_{j+1}]$
9. $nopeering\,[u_j, u_{j+1}] = 1$
10. altrimenti
11. $nopeering\,[u_{j-1}, u_j] = 1$

Fase 3: assegnazione delle relazioni di peering

1. Per ogni AS path $p = (u_1, u_2, \ldots, u_n)$
2. Per ogni i in $\{1, \ldots, n-1\}$
3. se $nopeering\,[u_i, u_{i+1}] \neq 1$ e $nopeering\,[u_{i+1}, u_i] \neq 1$ e $\frac{1}{R} < \frac{grado[u_i]}{grado[u_{i+1}]} < R$
4. $relazione\,[u_i, u_{i+1}] =$ peer-peer

Figura 2.7: Euristica proposta da [10] per la ricerca dei rapporti di peering

questo valore pilota l'inferenza delle relazioni provider-customer e sibling-sibling, ma non di quelle peer-peer. La sua presenza è giustificata dalla necessità di ridurre l'influenza di dati di routing BGP derivanti da configurazioni errate. Poiché, in genere, si assume che l'impatto di siffatte informazioni sia abbastanza limitato, si cerca di abbatterlo del tutto ignorando fenomeni che si presentano in modo relativamente sporadico.

Il secondo parametro, costituito dal valore R, influenza il calcolo della relazione peer-peer, e rappresenta un confine entro il quale devono mantenersi i gradi di due AS affinché tra di essi possa considerarsi esistente tale rapporto.

Le sperimentazioni effettuate in [10] sono basate sui dati di Oregon Route Views ([22]), e terminano con una convalida delle relazioni inferite fondata su informazioni interne della AT&T. L'algoritmo è stato utilizzato ponendo sempre $L = 1$, dal momento che il contenuto informativo di una quantità (relativamente) ridotta di dati di routing a disposizione sarebbe stato ulteriormente penalizzato dall'esclusione di elementi ritenuti di scarsa rilevanza. I risultati manifestano una percentuale di relazioni confermate pari al 96.5% nel caso in cui $R = \infty$, ed un risultato ancora migliore (99.1%) quando si è reso l'algoritmo più selettivo nei confronti delle relazioni di peering, ponendo $R = 60$.

2.3 Approccio basato su più punti di vista

Il secondo approccio sistematico al problema ad essere presentato è stato, nell'aprile del 2001, quello proposto in [8]. In esso compare per la prima volta una formulazione rigorosa del problema ToR, e si cerca di colmare le lacune derivanti dalle visioni inevitabilmente parziali fornite da un singolo looking glass prendendo in considerazione le informazioni derivanti da più *vantage points*. I dati di queste sorgenti vengono poi fusi ed elaborati, e l'inferenza delle relazioni avviene dando priorità ai rapporti confermati dal maggior numero di vantage points.

In [8] si propone anche una possibile classificazione gerarchica degli Autonomous Systems, finalizzata all'identificazione del così detto *dense core*, ovvero un insieme di AS di grandi dimensioni che costituiscono probabilmente il nucleo trasmissivo principale di Internet. In questa sede tale aspetto non viene comunque preso in considerazione.

2.3.1 Relazioni considerate e politiche

La soluzione qui descritta è stata elaborata con la consapevolezza dell'esistenza di relazioni di sibling e di backup, e tuttavia non prevedendo la possibilità di inferirle. Gli autori si sono invece soffermati maggiormente sui rapporti customer-provider e di peering.

Anche in questo caso (come nella soluzione proposta nella sezione 2.2) ad ogni rapporto commerciale si suppone associata una serie di politiche con le quali i router sono configurati. Tali politiche sono assunte essere le seguenti:

- *Esportazione ad un provider*: un AS esporta ai suoi provider le proprie rotte e quelle dei propri customer, ma non può esportare quelle apprese dai propri provider e peer;
- *Esportazione ad un peer*: un AS esporta ai suoi peer le proprie rotte e quelle dei propri customer, ma non può esportare quelle annunciategli dai propri provider e peer;
- *Esportazione ad un customer*: un AS esporta ai suoi customer le proprie rotte e quelle apprese dai propri customer, provider e peer.

In una situazione di rispetto di queste politiche, tutti i cammini estrapolati dalle informazioni di routing risultano classificabili nelle sole categorie *tipo 1* e *tipo 2*, secondo le modalità descritte nella definizione 2.2.

2.3.2 Algoritmo utilizzato

L'idea alla base della procedura illustrata in [8] è quella di sfruttare le singole informazioni fornite da più looking glass (chiamati nell'articolo *vantage points*) per inferire le relazioni tra AS, evitando invece di limitarsi a fondere tali informazioni in un singolo grafo.

La prima operazione che viene effettuata è quella di analizzare i dati di routing di ogni singolo vantage point in modo tale da costruire un grafo dei nodi che da esso si dipartono. Per poter fare questo, vengono analizzati gli AS

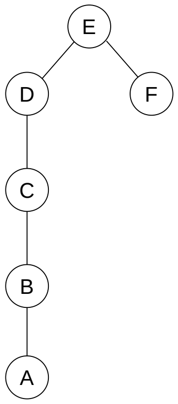

Figura 2.8: Albero dei cammini visibili presso un vantage point

path prelevati presso il vantage point X considerato; ciascuno di essi, in quanto prelevato presso X, avrà come primo AS X stesso[4].

Di conseguenza, è possibile tracciare un albero che abbia come radice X e rappresenti il modo in cui gli AS path si sviluppano intorno ad esso. Si denoti con G_X tale albero. Ad esempio, per 3 cammini (E, D, C, B, A), (E, F) e (E, D, C) si ottiene la struttura in figura 2.8.

A questo punto si associa a ciascun Autonomous System nell'albero costruito un particolare *rank*. Il criterio con cui viene assegnato questo valore è fondato su alcune ipotesi relative alla struttura dell'intero grafo di Internet ed al modo in cui esso appare presso i vari looking glass. In particolare, Internet viene pensata come composta in modo predominante da AS di piccole dimensioni; la visione che essi hanno della rete, in termini di AS path associati alle varie rotte, sarà generalmente caratterizzata da un tratto in cui si incontrano link customer$\rightarrow$provider (tratto ascendente), seguito da un tratto costituito da link provider$\leftarrow$customer (tratto discendente) di lunghezza paragonabile al precedente. Viceversa, il punto di vista di un grande AS è tale da far prevalere in molti cammini il tratto discendente.

Secondo questa visione, una gran parte degli archi del grafo G_X, orientato secondo le relazioni tra AS, cade in una grossa porzione aciclica, mentre i restanti costituiscono una componente fortemente connessa che circonda l'AS in esame.

L'attribuzione dei valori di ranking procede operando una potatura inversa (a partire dalle foglie) del grafo G_X. Ad ogni passo, si considerano tutte le foglie del grafo, e si associa ad esse il valore corrente del rank. Il grafo viene poi privato di tali foglie e si incrementa il valore del rank, dopodiché si esaminano ancora le nuove foglie e così via. L'algoritmo si arresta nel momento in cui il grafo non ha più foglie, nel qual caso si può constatare il raggiungimento della componente connessa intorno all'AS X e si può attribuire a tutti i nodi rimanenti lo stesso rank. Più formalmente, la procedura di potatura inversa

[4]Si ricorda che, per via dell'ordine con cui sono inseriti, il primo AS di un AS path è l'ultimo AS cui è giunto l'annuncio della rotta; in realtà, un AS aggiunge il proprio numero in testa ad un AS path soltanto nel momento in cui esporta il corrispondente prefisso mediante un annuncio; in questa sede, tuttavia, si supporrà che gli AS path noti ad un AS contengano già il suo numero identificativo in testa.

procede come schematizzato in figura 2.9 (l'insieme $v(G)$ è l'insieme dei vertici del grafo G).

```
G = G_X
r = 1
finché (foglie(G) ≠ ∅)
    per ogni u ∈ foglie(G)
        rank(u) = r
    v(G) = v(G) − foglie(G)
    r = r + 1
per ogni u ∈ v(G)
    rank(u) = r
```

Figura 2.9: Algoritmo di potatura inversa

Mentre gli archi rimasti all'interno della componente connessa intorno ad X non possono ancora essere classificati in base alle relazioni che su di essi hanno luogo, il fatto di aver assegnato un rank diverso ai nodi "lontani" da X consente già di cominciare a formulare supposizioni sui loro ruoli. Per esempio, se tra due AS A e B c'è un link, e secondo le informazioni prelevate al vantage point X risulta $rank(A) > rank(B)$, allora A è probabilmente un provider di B, almeno dal punto di vista di X.

L'ultima fase dell'algoritmo di inferenza consiste nell'analizzare i nodi non distinguibili in base al rank e nel confermare o rivedere le relazioni per le quali sia già stata formulata un'ipotesi. Tutto ciò avviene mettendo a confronto le informazioni elaborate per gli N vantage point a disposizione. Per fare questo, ad ogni nodo viene associato un vettore N-dimensionale $(r_{i,1}, \ldots, r_{i,N})$, dove $r_{i,j}$ è il rank associato all'AS i dal vantage point j. Successivamente si definiscono le seguenti quantità:

$$l\,(i,j) = |\{(r_{i,k}, r_{j,k}) \ \mid k \in \{1,2,\ldots,N\} \ \text{ e } r_{i,k} > r_{j,k}\}|$$

$$e\,(i,j) = |\{(r_{i,k}, r_{j,k}) \ \mid k \in \{1,2,\ldots,N\} \ \text{ e } r_{i,k} = r_{j,k}\}|$$

La funzione l fornisce il numero di vantage point secondo cui i risulta avere rank maggiore di j, mentre e rappresenta il numero di vantage point da cui i e j vengono visti come aventi lo stesso rank.

Ciò premesso, valgono le regole di inferenza che seguono.

Relazioni peer-peer Alcuni archi di peering[5] si manifestano in molti punti di vista: per essi è possibile utilizzare semplicemente la regola di Equivalenza; nel caso in cui, invece, la relazione tra AS sia rivelata soltanto da un numero limitato di vantage point si utilizza quella di Equivalenza Probabilistica:

- *Equivalenza*: due AS i e j si dicono *equivalenti* se $e\,(i,j) > N/2$; se i e j sono collegati da un arco, ciò vuol dire che a tale arco viene attribuita una relazione di peering[5];
- *Equivalenza Probabilistica*: due AS sono *probabilisticamente equivalenti* se $\frac{1}{\delta_1} \leq \frac{l(i,j)}{l(j,i)} \leq \delta_1$, con $\delta_1 \simeq 1$.

[5] In senso commerciale.

Relazioni provider-customer Poiché può succedere che un vantage point j associ a sé stesso ed ai propri customer un rank non coerente con la relazione che ha con essi (per esempio, può essere $rank(j) > rank(i)$ anche se i è un provider di j), in alcune situazioni è preferibile utilizzare la regola di Dominanza Probabilistica in luogo di quella di Dominanza pura:

- *Dominanza*: si dice che un AS i *domina* un altro AS j se $l\,(i,j) \geq N/2$ e $l\,(j,i) = 0$; se i e j sono collegati da un arco, ciò corrisponde ad inferire che i è un provider di j;
- *Dominanza Probabilistica*: si dice che un AS i *domina probabilisticamente* un altro AS j se $\frac{l(i,j)}{l(j,i)} > \delta_0$, con δ_0 sufficientemente grande.

Le prove effettuate in [8] mostrano come sia possibile, con questo algoritmo, raggiungere una percentuale di archi orientati pari al 99.26% (per gli altri non è possibile completare l'inferenza), utilizzando come parametri del calcolo i valori $\delta_1 = 2$ e $\delta_0 = 3$. Le informazioni utilizzate sono state prelevate dai 10 vantage points elencati nella tabella 2.1, e sono attualmente disponibili all'indirizzo [9], sotto la data *18 aprile 2001*.

# AS	Nome
1	Genuity
1740	CERFnet
3549	Globalcrossing
3582	University of Oregon
3967	Exodus Comm.
4197	Global Online Japan
5388	Energis Squared
7018	AT&T
8220	COLT Internet
8709	Exodus Europe

Tabella 2.1: Vantage points utilizzati

Nel lavoro [8] si opera anche una classificazione dei cammini invalidi sulla base del tipo di anomalia che su di essi si manifesta. La classificazione viene effettuata individuando, come in figura 2.4, le 4 situazioni base a causa della cui presenza decade la condizione di validità. Secondo i risultati calcolati, la maggior parte dei cammini invalidi presenta un'anomalia di tipo 4. Le anomalie di tipo 1, presenti in minor misura, vengono considerate un possibile indice della presenza di relazioni di sibling, poiché tale ruolo si addice molto di più ad un AS che stia transitando traffico verso due suoi supposti provider.

2.4 Approccio basato sulla soddisfacibilità di formule logiche

La soluzione presentata più recentemente per l'inferenza delle relazioni tra Autonomous Systems è del 2002, ed è stata elaborata quasi simultaneamente in [20] e [21].

L'approccio proposto prevede innanzitutto di riformulare il problema ToR per verificare l'esistenza di un modo di assegnare le relazioni tra AS tale da non introdurre cammini invalidi. Viene poi proposto un algoritmo per il calcolo di tale assegnazione, basato su una riduzione del problema ad un'istanza di 2–SAT.

Infine, per quello che riguarda il problema ToR nella sua versione originale, in [20] viene presentata un'euristica finalizzata a ricercare un insieme massimale di cammini validi, mentre in [21] si dimostrano dei limiti superiori alla qualità delle soluzioni che è possibile calcolare in tempo polinomiale.

Nel seguito, per garantire la coerenza dell'esposizione ed evitare considerazioni eccessivamente complesse sulla complessità computazionale, si utilizza come riferimento il lavoro [20].

2.4.1 Relazioni considerate e politiche

Come nei lavori precedenti, anche qui viene mantenuta l'assunzione che le politiche utilizzate per implementare ciascun tipo di relazione obbediscano alle seguenti regole:

- *Esportazione ad un provider*: un customer esporta ad un provider le proprie rotte e quelle apprese dai propri customer, ma non quelle annunciategli dai propri provider e peer,
- *Esportazione ad un customer*: un provider esporta ad un customer le proprie rotte e quelle apprese dai propri customer, provider e peer;
- *Esportazione ad un peer*: due peer si esportano reciprocamente le proprie rotte e quelle dei rispettivi customer, ma non quelle dei propri provider e peer.

Anche in questo caso, il rispetto di queste politiche comporta che tutti i cammini su cui si sono propagati gli annunci siano di tipo 1 oppure di tipo 2.

2.4.2 Algoritmo utilizzato

L'obiettivo primario nei lavori [20] e [21] è quello di catturare nel modo migliore possibile le relazioni customer-provider. A tal fine, il primo passo che viene compiuto è la riformulazione del problema ToR in termini della sua corrispondente versione decisionale:

Problema 2.2 (ToR–D) *Dato un grafo non orientato G, un insieme di cammini P ed un intero k, determinare se è possibile attribuire un'orientazione ad alcuni archi di G in modo tale che il numero di cammini invalidi in P sia al più k.*

Senza perdere di generalità, è possibile basarsi su una formulazione ulteriormente differente, in cui anziché richiedere un'orientazione parziale (di alcuni degli archi di G), se ne ricerca una totale:

Problema 2.3 (ToR–D–simple) *Dato un grafo non orientato G, un insieme di cammini P ed un intero k, determinare se è possibile attribuire un'orientazione a tutti gli archi di G in modo tale che il numero di cammini invalidi in P sia al più k.*

L'intercambiabilità delle due formulazioni è presto dimostrata: se esiste un'orientazione corretta di tutti gli archi di G (cioè se ToR–D–simple ammette soluzione), allora tale orientazione costituisce una soluzione anche per il problema ToR–D; viceversa, data una soluzione per quest'ultimo, è possibile trovarne una per ToR–D–simple soltanto fornendo un'orientazione arbitraria agli archi che ancora non ne hanno. Infatti, l'impatto di quest'ultima operazione su ogni cammino può essere soltanto di due tipi: se il cammino è invalido, può rimanere tale o diventare valido con la nuova orientazione; se è valido, non può che essere di tipo 2 (vista la presenza di un arco non orientato), e dunque la nuova orientazione lo trasforma in uno di tipo 1. In entrambi i casi non si ha mai un aumento nel numero dei cammini invalidi, e quindi la soluzione così modificata risulta corretta anche per ToR–D–simple.

La strategia utilizzata per risolvere ToR–D–simple consiste nell'operare una sua riduzione verso il problema 2–SAT. A tal fine occorre compiere i seguenti passi:

1. si attribuisce un'orientazione arbitraria a tutti gli archi del grafo G;

2. per ogni arco (v_i, v_j) di G si introduce una variabile $x_{i,j}$ nell'istanza di problema 2–SAT; supponendo che l'arco (v_i, v_j) sia stato orientato al passo 1 come $v_i \to v_j$, il valore di verità di $x_{i,j}$ deve essere interpretato in questo modo: se $x_{i,j}$ è vera, l'orientazione finale dell'arco (v_i, v_j) cui è associata è $v_i \to v_j$ (quella assegnatagli inizialmente); se è falsa, l'orientazione di tale arco è $v_j \to v_i$; questa convenzione è descritta in figura 2.10; si noti come debba risultare $x_{i,j} = \overline{x}_{j,i}$;

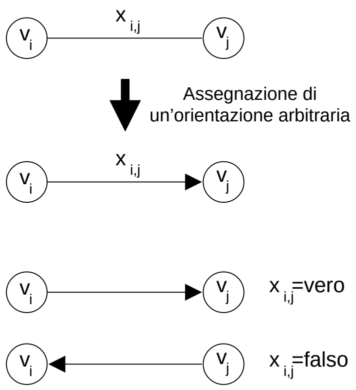

Figura 2.10: Significato dei valori di verità delle variabili del problema 2–SAT

3. per ogni cammino costituito da almeno 2 archi, si considerano tutte le possibili triple di vertici consecutivi v_{i-1}, v_i, v_{i+1}; a seconda delle orientazioni attribuite ai vari archi, si aggiungono al problema 2–SATdelle clausole, in accordo con quanto schematizzato nella figura 2.11 ed illustrato di seguito:

   – se gli archi sono orientati $v_{i-1} \to v_i$ e $v_i \leftarrow v_{i+1}$, si introduce

$$x_{i-1,i} \vee x_{i+1,i}$$

– se gli archi sono orientati $v_{i-1} \leftarrow v_i$ e $v_i \rightarrow v_{i+1}$, si introduce

$$\overline{x}_{i,i-1} \vee \overline{x}_{i,i+1}$$

– se gli archi sono orientati $v_{i-1} \rightarrow v_i$ e $v_i \rightarrow v_{i+1}$, si introduce

$$x_{i-1,i} \vee \overline{x}_{i,i+1}$$

– se gli archi sono orientati $v_{i-1} \leftarrow v_i$ e $v_i \leftarrow v_{i+1}$, si introduce

$$\overline{x}_{i,i-1} \vee x_{i+1,i}$$

Si noti come, per tutte le clausole introdotte, valga la seguente catena di equivalenze:

$$\left.\begin{array}{l} x_{i-1,i} \vee x_{i+1,i} \\ \overline{x}_{i,i-1} \vee \overline{x}_{i,i+1} \\ x_{i-1,i} \vee \overline{x}_{i,i+1} \\ \overline{x}_{i,i-1} \vee x_{i+1,i} \end{array}\right\} \equiv (x_{i-1,i} \vee \overline{x}_{i,i+1}) \equiv (x_{i,i+1} \Longrightarrow x_{i-1,i}) \equiv (x_{i,i-1} \Longrightarrow x_{i+1,i})$$

Questo equivale ad affermare che, se un arco è orientato $v_i \rightarrow v_j$, allora tutti gli archi che precedono v_i in ogni AS path devono essere anch'essi orientati verso "destra" ($v_{i-k} \rightarrow v_{i-k+1} \rightarrow \ldots \rightarrow v_{i-1} \rightarrow v_i \rightarrow v_j$), e viceversa nel caso in cui l'arco sia orientato in modo opposto. La catena di implicazioni così costituita è finalizzata ad evitare l'insorgere di situazioni di invalidità, come si avrebbero se fosse $v_{i-1} \leftarrow v_i \rightarrow v_{i+1}$ (anomalia di tipo 1) per qualche i.

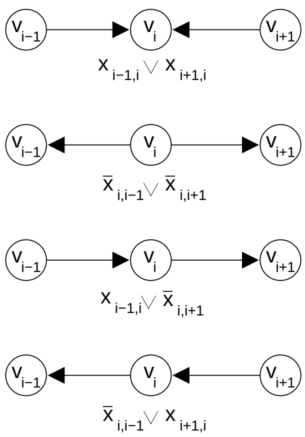

Figura 2.11: Clausole del problema 2–SAT

Il problema 2–SAT, nella sua formulazione classica, mira alla ricerca di un'assegnazione di valori di verità alle variabili $x_{i,j}$ così introdotte tale da rendere vere tutte le clausole. Questo fa già intuire che, a causa di aspetti relativi alla complessità computazionale, il problema che può essere risolto più efficientemente è quello di orientare gli archi del grafo in modo che non vi siano affatto ($k = 0$) cammini invalidi.

Tuttavia, prima di prendere in esame il procedimento utilizzato per ottenere questa orientazione, occorre stabilire se, di fatto, una tale assegnazione esista oppure no. Un approccio tipicamente utilizzato nella letteratura consiste nel mappare l'istanza di problema 2–SAT in un apposito grafo $G_{2-\mathrm{SAT}}$, costruito nel modo che segue:

- per ogni variabile $x_{i,j}$ si introducono due nodi, corrispondenti ai suoi due letterali $x_{i,j}$ e $\overline{x}_{i,j}$;
- per ogni clausola $l_1 \vee l_2$ (dove l_1 e l_2 sono letterali) si introduce la coppia di archi orientati $\overline{l}_1 \rightarrow l_2$ e $\overline{l}_2 \rightarrow l_1$, che rappresentano le rispettive implicazioni logiche $\overline{l}_1 \Longrightarrow l_2$ e $\overline{l}_2 \Longrightarrow l_1$.

Il problema 2–SAT ammette soluzione se e solo se nel grafo $G_{2-\mathrm{SAT}}$ non ci sono cicli che contengano entrambi i letterali $x_{i,j}$ e $\overline{x}_{i,j}$ di qualche variabile $x_{i,j}$; una simile struttura rappresenterebbe, infatti, la contraddizione logica $x_{i,j} \iff \overline{x}_{i,j}$. La verifica dell'esistenza di cicli di questo tipo può essere effettuata calcolando le componenti fortemente connesse del grafo $G_{2-\mathrm{SAT}}$, e verificando per ciascuna di esse se contenga o meno entrambi i letterali di una variabile.

Una volta descritta la riduzione da ToR–D–simple a 2–SAT, ed individuato il modo di risolvere quest'ultimo problema, ci si accorge che il caso in cui sia $k = 0$ (cioè in cui si ricerchi un'orientazione che non introduca nessun cammino invalido) consente di calcolare le relazioni tra AS in tempo $O(n + m + q)$, essendo n il numero di nodi del grafo degli AS, m il numero dei suoi archi e q la somma delle lunghezze di tutti gli AS path. Di converso, è possibile dimostrare (mediante una riduzione dal problema MAX–2–SAT–D a ToR–D–simple) che il problema ToR–D–simple (e quindi ToR) nella sua formulazione generale è NP-completo.

Una volta stabilito che l'istanza di 2–SAT ammette soluzione, è possibile calcolare i valori di verità da assegnare alle variabili (e quindi l'orientazione degli archi del grafo G) analizzando la posizione dei vari nodi del grafo $G_{2-\mathrm{SAT}}$. Tenendo presente che $(x_{i,j} \Longrightarrow \overline{x}_{i,j}) = vero \iff x_{i,j} = falso$ e $(\overline{x}_{i,j} \Longrightarrow x_{i,j}) = vero \iff x_{i,j} = vero$, questo vuol dire attribuire i valori di verità sulla base dell'ordine in cui si susseguono, in un cammino orientato[6], i due letterali di ciascuna variabile.

A tal fine è possibile definire una funzione f su ogni vertice del grafo $G_{2-\mathrm{SAT}}$ tale che, se esiste un cammino orientato da un vertice u ad un vertice v, allora $f(u) \leq f(v)$. In questo modo, se $f(x_{i,j}) > f(\overline{x}_{i,j})$, allora si pone $x_{i,j} = vero$, altrimenti si pone $x_{i,j} = falso$. Si osservi come, nel caso in cui l'istanza di problema 2–SAT fosse soddisfacibile (unica situazione in cui ha senso calcolare i valori di verità), non potrebbe mai risultare $f(x) = f(\overline{x})$. La definizione della funzione f può sfruttare algoritmi esistenti per il calcolo del ***topological sort*** dei nodi di un grafo, purché quest'ultimo venga eseguito su una versione di $G_{2-\mathrm{SAT}}$ in cui ciascuna componente fortemente connessa sia già stata contratta in un unico nodo.

Si osservi, infine, come sia possibile imporre dei vincoli nell'orientazione calcolata semplicemente introducendo delle clausole fittizie del tipo $(x_{i,j} \vee x_{i,j})$ nell'istanza di 2–SAT.

[6]che corrisponde ad una sequenza di implicazioni logiche

Per quanto riguarda le sperimentazioni effettuate, è emerso un comportamento interessante, che consiste nella possibilità di trovare orientazioni senza cammini invalidi solo quando i dati utilizzati si riferiscano a visioni parziali della rete. Infatti, in tutti i casi analizzati, non è stato possibile fare altrettanto sui dati del server Oregon Route Views ([22]), che integra le informazioni di decine di altri router con cui ha sessioni di peering attive.

2.4.3 Euristiche

Una volta appurata la difficoltà computazionale insita nella minimizzazione del numero di cammini invalidi, è stata messa a punto un'euristica che mira all'ottenimento di un insieme massimale di cammini che rendano il problema ToR–D–simple soddisfacibile. L'euristica funziona in questo modo:

1. si considerano tutti i dati di routing a disposizione;

2. si verifica se i dati correntemente in esame sono soddisfacibili (cioè si verifica se, in base ad essi, è possibile ottenere un'orientazione senza cammini invalidi): se è così, si salta al passo 3; altrimenti, si ricerca l'arco che compare nel minor numero di AS path e si cancellano tutti gli AS path in cui è presente; si ripete poi daccapo questo passo;

3. per garantire che l'insieme sia davvero massimale, si analizzano nuovamente i cammini scartati al passo 2, e si prova a riconsiderarli uno alla volta: se un cammino non compromette la soddisfacibilità del problema, lo si reintroduce nell'insieme degli AS path; altrimenti, lo si scarta definitivamente.

Il calcolo di un'orientazione sulla base di un insieme di cammini così costruito consente di ottenere risultati in cui la percentuale di cammini invalidi (rispetto a tutti gli AS path da cui si è partiti) oscilli tra lo 0.13% e lo 0.57%, a seconda del punto di vista considerato.

Capitolo 3

Un ambiente per l'analisi e lo studio dell'evoluzione delle relazioni tra Autonomous Systems

Il primo requisito da soddisfare quando si deve effettuare un'analisi sistematica è costituito dalla necessità di definire in modo il più possibile preciso i criteri in essa seguiti e gli strumenti con i quali la si effettua. Per quanto riguarda questi ultimi, si può decidere di utilizzarne di preesistenti oppure di progettarne e realizzarne di nuovi, più orientati all'utilizzo nel contesto di cui ci si occupa.

In questo capitolo viene innanzitutto descritta la situazione attuale sullo studio dell'evoluzione delle relazioni; successivamente, si giustifica la scelta di ricorrere alla progettazione e realizzazione di un nuovo ambiente finalizzato a supportare tale studio; infine, si mostrano le caratteristiche architetturali ed implementative dell'ambiente nel suo insieme e dei suoi singoli componenti.

3.1 Analisi dei requisiti

Il processo di analisi dei requisiti, effettuato alla luce degli strumenti esistenti per il trattamento di dati relativi agli Autonomous Systems, ha messo in evidenza l'esigenza di creare un ambiente specifico per effettuare lo studio che qui viene presentato. Poiché nessuno degli ambienti esistenti sembra soddisfare adeguatamente le caratteristiche qui analizzate, è stata la stessa analisi dei requisiti a far sì che si traesse questa conclusione.

3.1.1 Requisiti e motivazioni per l'introduzione di un nuovo ambiente

Il problema dell'inferenza delle relazioni tra Autonomous Systems è, come si è già enfatizzato più volte, di formulazione abbastanza recente. In quanto tale, la maggior parte dei lavori che ne parlano si pone come obiettivo l'introduzione di soluzioni il più possibile valide ed efficienti. Non sono ancora stati diffusi, invece, risultati di sperimentazioni massicce, finalizzati a studiare sia il comportamento delle soluzioni trovate, sia (se la qualità del sistema lo consente) quello delle relazioni da esse rivelate. La ragione per cui questo non è ancora accaduto è, probabilmente, proprio da ricercarsi nell'estrema attualità del problema, che spinge a posticipare questo tipo di atteggiamenti verso un momento in cui le scelte operate negli algoritmi di inferenza siano state ulteriormente consolidate.

In realtà, oltre alla necessità di inquadrare rigorosamente queste scelte, un altro aspetto che può aver dissuaso dall'adottare un approccio pragmatico è costituito dalla quasi totale assenza di metodi e strumenti che consentano di affrontare l'analisi in modo altrettanto rigoroso e sistematico.

Studiare l'evoluzione delle relazioni tra AS significa, d'altra parte, avere a disposizione quantomeno due strumenti fondamentali:

- un algoritmo efficace ed efficiente per l'inferenza delle relazioni;
- un ambiente che consenta di effettuare diversi tipi di analisi sulle relazioni inferite, in modo il più possibile flessibile ed, al tempo stesso, chiaro ed affidabile.

Mentre il primo di questi due aspetti è stato considerato con un notevole livello di approfondimento, sul secondo non è stata ancora rivolta particolare attenzione. Più precisamente, anche sul primo aspetto esistono ancora delle ambiguità: mentre per alcune soluzioni sono, infatti, disponibili anche delle implementazioni liberamente utilizzabili (all'indirizzo [11] per l'algoritmo proposto in [10] ed all'indirizzo [23] per quello proposto in [20]), per altre l'unica fonte informativa su cui basarsi è costituita dalla descrizione teorica, talvolta priva di specifiche relative ad alcuni dei comportamenti adottati in fase di sperimentazione (per esempio, per [8] non traspare chiaramente dall'articolo se i risultati ottenuti tengano o meno conto delle anomalie di tipo 4 –che in alcuni punti vengono invece considerati indizi utili per l'individuazione di sibling–).

Il secondo aspetto, invece, tralasciando qualche script di elaborazione dei risultati ottenuti (peraltro non disponibile pubblicamente), non è mai stato particolarmente approfondito, specialmente per quello che riguarda la possibilità di incrociare molti risultati di inferenze per valutarne i comportamenti lungo la dimensione temporale.

Per questo motivo si è deciso di finalizzare l'analisi dei requisiti qui presentata alla definizione di un *banco da lavoro* messo a punto appositamente per lo scopo.

3.1.2 Caratteristiche dell'ambiente

L'individuazione delle caratteristiche e dei requisiti necessari perché l'ambiente possa essere utilizzato proficuamente ha richiesto di porsi di fronte a diversi problemi.

- *Affidabilità.* Prima di tutto, poiché deve elaborare quantitativi di informazioni notevoli, l'ambiente deve garantire un funzionamento il più possibile costante ed affidabile. Questo requisito, in realtà, deve valere anche per le implementazioni degli algoritmi utilizzati per inferire le relazioni; i vari esperimenti effettuati, infatti, hanno mostrato come esse possano, per esempio, produrre informazioni errate in presenza di anomalie nei dati in input. Questo suggerisce la necessità di introdurre anche uno strumento che "protegga" le implementazioni esistenti da eventuali impurità dei dati in ingresso.

- *Flessibilità.* L'analisi dell'evoluzione delle relazioni tra AS può richiedere il calcolo di diversi tipi di statistiche sulle informazioni in esame. L'ambiente deve, dunque, assicurare un notevole livello di flessibilità, senza, tuttavia, pregiudicarne le altre qualità, ivi inclusa la semplicità d'uso.

- *Chiarezza.* Uno degli elementi che spesso viene maggiormente trascurato nella creazione di un nuovo sistema è la definizione precisa dei comportamenti di ogni suo singolo componente. L'ambiente di cui ci si vuole avvalere deve invece consentire il tracciamento di un profilo esatto delle trasformazioni subite dai dati elaborati, per evitare che i risultati finali perdano la propria significatività o addirittura manifestino delle incoerenze dovute a passaggi intermedi poco motivati.

- *Efficacia.* Lo scopo primario per cui viene introdotto questo ambiente è la generazione di rapporti sui cambiamenti nelle relazioni attribuite alle coppie di AS. Pur conservando la sua caratteristica di flessibilità, quindi, esso deve permettere primariamente il rapido conseguimento di questo tipo di risultati, senza costringere l'utente ad effettuare operazioni aggiuntive poco attinenti.

- *Modificabilità.* L'ambiente in esame potrebbe essere utilizzato in altri contesti: per esempio per verificare le caratteristiche dell'evoluzione sulla base di nuovi algoritmi di inferenza, oppure semplicemente per lavorare su altre sorgenti informative utilizzando algoritmi esistenti. Quindi, deve consentire, in modo relativamente semplice, l'introduzione di nuovi componenti o la modifica di quelli esistenti.

- *Portabilità.* Se il sistema soddisfa adeguatamente le proprietà elencate, è ragionevole pensare che possa essere utilizzato anche al di fuori degli scopi delineati in questa trattazione. In particolare, è possibile che sia necessario eseguirlo anche su piattaforme diverse da quella su cui viene concepito. Per questo, è determinante che abbia anche un buon grado di portabilità.

- *Leggibilità.* È importante che le scelte operate in fase di realizzazione siano in qualche modo documentate e giustificate. Molte di esse, infatti, sono motivate dalla necessità di definire convenzioni o predisporre particolari risultati intermedi da sottoporre poi ad elaborazioni successive. I dettagli sugli algoritmi e sulla codifica dei vari componenti dell'ambiente devono quindi essere in qualche modo descritti, anche per garantire la caratteristica di modificabilità.

3.2 Progettazione dell'ambiente

Una volta terminata la fase di analisi dei requisiti, nella quale è stato possibile confermare in modo definitivo l'esigenza di creare un nuovo ambiente, è stata fatta una serie di scelte progettuali volte a garantire le proprietà in essa individuate.

L'ambiente che è stato elaborato prende il nome di TORQUE, acronimo di *Type Of Relationship QUality Evaluation toolkit.*

Al fine di conseguire il miglior grado di flessibilità, mantenendo al tempo stesso il sistema affidabile e chiaro, si è scelto di suddividerne le funzionalità fondamentali in più moduli, ciascuno dei quali utilizzabile in modo indipendente. I moduli possono interagire semplicemente facendoli operare su file comuni.

3.2.1 Un ambiente modulare: descrizione dei singoli componenti

La caratteristica fondamentale dell'ambiente TORQUE è che esso deve operare su grafi di AS, ottenuti sulla base di un'orientazione assegnata da un algoritmo di inferenza di relazioni.

Generatore di grafi di AS Il primo problema da affrontare è relativo alla necessità di consentire la sintesi dei grafi sulla base di dati di provenienza molto varia. Questo aspetto nasce per una serie di motivi: innanzitutto, l'orientazione stessa può essere prodotta utilizzando strumenti diversi, ciascuno dei quali riporta i risultati tipicamente utilizzando un proprio formato di output; inoltre, nel caso dei dati pubblicati in [9], vi sono informazioni diverse distribuite in file diversi (per esempio, gli archi che l'algoritmo non è riuscito ad orientare vengono riportati separatamente).

Vi sono poi alcuni tipi di analisi che richiedono la costruzione di grafi ottenuti mescolando informazioni diverse: per esempio, per conoscere il numero di archi che un algoritmo di inferenza non è riuscito ad orientare, può essere necessario introdurre, oltre agli archi orientati forniti dall'algoritmo stesso, anche tutti gli archi non orientati che esso non riporta ma che appaiono nell'insieme degli AS path considerati.

Infine, alcuni dati non possono, per loro natura, essere prodotti dall'algoritmo di inferenza, ma devono necessariamente essere calcolati "al volo" all'atto della creazione del grafo. Ne costituisce un esempio il così detto *path covering*, ossia il numero di AS path che attraversano ogni arco.

Da questa serie di considerazioni è nata l'esigenza di fornire a questo modulo una notevole flessibilità per quanto riguarda le informazioni in ingresso. A tal fine, sono state distinte due caratteristiche ortogonali dei suoi possibili input:

- il formato del file, ossia il modo in cui i dati sono organizzati al suo interno; i formati che possono essere utilizzati sono:
  - tabella BGP: il risultato dell'esecuzione del comando `show ip bgp` su un looking glass di tipo telnet;
  - BGPPATHSAT: l'output dell'esecuzione dell'algoritmo di inferenza presentato nella sezione 2.4;
  - vantage: il formato utilizzato da [9] per pubblicare i risultati dell'esecuzione dell'algoritmo presentato nella sezione 2.3;
  - path: una lista di AS path;
  - unknown: un insieme di archi per i quali non è stato possibile inferire un'orientazione (file di questo tipo vengono utilizzati in [9]);

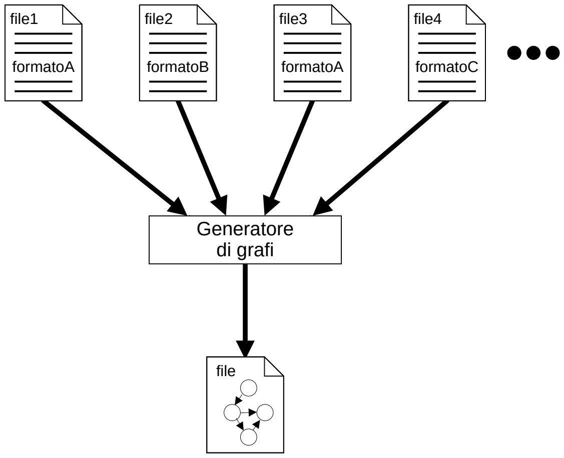

Figura 3.1: Funzionamento del modulo di generazione dei grafi

- il contenuto informativo, ovvero la (o *le*) caratteristica del grafo che deve essere prelevata dal file; può essere scelto nel seguente insieme:
  - nodi del grafo;
  - archi (non orientati) del grafo;
  - orientazione degli archi del grafo;
  - *path covering*, ossia il numero di cammini che attraversano ogni arco;
  - *IP covering*, ossia il numero di indirizzi IP distinti che sono stati annunciati su ogni arco.

La figura 3.1 mostra come sia possibile comporre informazioni e formati di file diversi per ottenere un unico grafo finale. Per esempio, è possibile prelevare nodi ed archi da un file che contiene una lista di AS path, e l'orientazione degli archi da un file prodotto da un software di inferenza. Il risultato è un grafo che ha tutti i nodi e gli archi visibili dalla tabella BGP considerata, ed in cui alcuni archi sono orientati secondo le relazioni inferite. Su un grafo del genere è possibile calcolare immediatamente il numero di situazioni in cui non è stato possibile assegnare un'orientazione.

Generatore di grafi differenziali Il punto chiave del sistema TORQUE è, naturalmente, costituito dalla possibilità di analizzare differenze tra le orientazioni. Un elemento di primaria importanza consiste dunque nella possibilità di valutare la stabilità nell'orientazione assegnata nel tempo ad un arco: questo consente, infatti, di individuare immediatamente quali coppie di AS abbiano alterato la propria relazione e quanto spesso ciò sia accaduto. La disponibilità di un'informazione di questo tipo consente sia di studiare l'evoluzione dei rapporti tra AS, sia di convalidare la qualità di un algoritmo che li inferisce: infatti, in un periodo in cui non vi sono stati grossi cambiamenti dal punto di vista commerciale, è lecito supporre che non ve ne siano stati neanche nell'orientazione calcolata, e questo strumento deve consentire questa verifica.

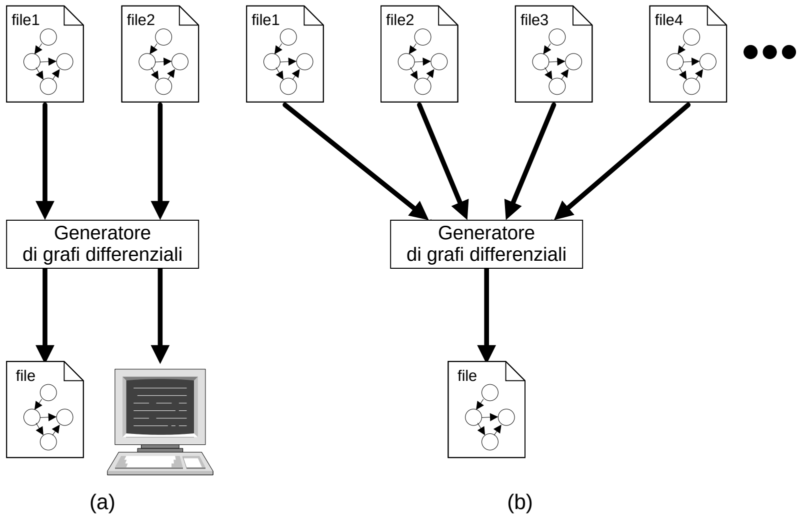

Figura 3.2: Funzionamento del modulo di generazione di grafi differenziali

In secondo luogo, può essere utile anche valutare differenze a granularità più fine, ottenute dal confronto di coppie di grafi. Tali differenze possono essere classificate in due categorie ortogonali:

- differenze in termini di presenza o assenza di archi; in altri termini, rientra in questo gruppo la capacità di isolare la porzione comune (in termini di archi) ai due grafi in esame;
- differenze in termini di orientazione: è di primaria importanza avere la possibilità di localizzare quali archi presentano nei due grafi orientazioni opposte o, comunque, diverse.

Per supportare entrambe queste funzionalità, il modulo di generazione di grafi differenziali è stato progettato in modo da poter essere eseguito in due modalità diverse, rappresentate in figura 3.2.

La prima modalità consiste nell'analisi dettagliata di due grafi (figura 3.2(a)), e prevede che vengano forniti in ingresso esattamente due file generati in precedenza con uno dei moduli dell'ambiente TORQUE (molto probabilmente con il modulo di generazione dei grafi). I dati generati in uscita possono essere di vario genere; nel caso più semplice, si può trattare di valori numerici che rappresentino quantità di nodi ed archi presenti in uno solo dei grafi o in entrambi, eventualmente insieme ad un conteggio degli archi orientati in modo diverso. Nel caso più generale, invece, si può richiedere la creazione di un nuovo grafo (a sua volta manipolabile con tutti gli strumenti di TORQUE), che rappresenti la parte sovrapponibile dei due forniti in ingresso (ossia contenga solo gli archi presenti in entrambi) oppure il complemento dell'uno rispetto all'altro (cioè abbia soltanto archi che compaiono in uno solo dei grafi). La flessibilità dello strumento dovrebbe essere tale da consentire la generazione di tutti i grafi che contengano gli archi appartenenti ai vari insiemi rappresentati in figura 3.3.

La seconda modalità di funzionamento consiste nell'analisi a più ampio raggio effettuata mettendo a confronto un numero qualunque di grafi (figura 3.2(b)).

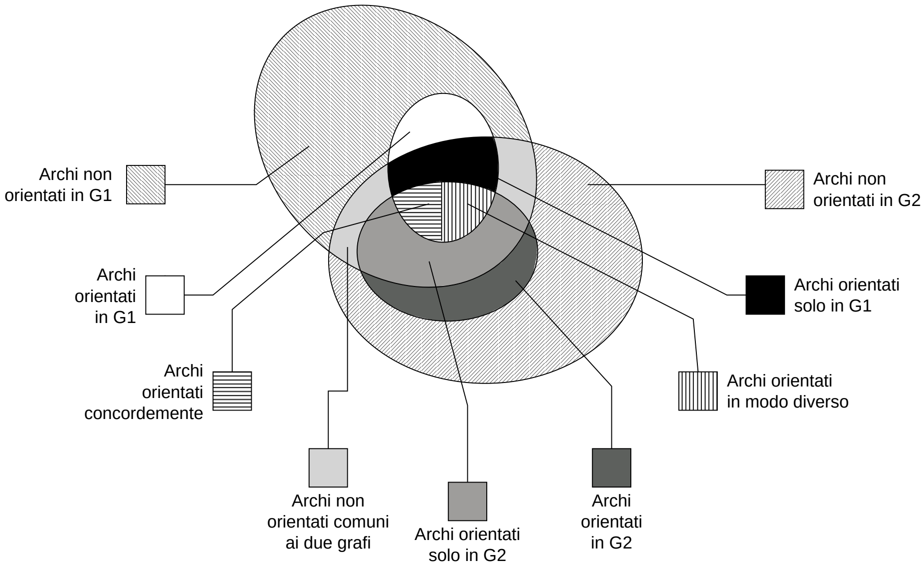

Figura 3.3: Tipi di archi individuati nella sovrapposizione di due grafi G_1 e G_2

Il risultato ottenuto è, in questo caso, una serie di indici che descrivano il comportamento dell'orientazione di ogni singolo arco. Piuttosto che limitarsi ad elencare tali indici e gli archi cui essi sono associati, il generatore di grafi differenziali può elaborare un nuovo grafo (grafo "unione"), con tutti gli archi (ed i relativi nodi terminali) presenti in almeno uno dei grafi in ingresso, in cui ciascun arco sia etichettato con questi valori. Questa modalità è, evidentemente, quella di maggior interesse per lo studio dell'evoluzione delle relazioni tra AS.

Analizzatore di grafi Entrambi i moduli per la generazione di grafi di AS e di grafi differenziali possono produrre dei file che contengano la rappresentazione del grafo di cui si è richiesta la costruzione. Tale rappresentazione utilizza un proprio formato per l'immagazzinamento delle informazioni, sicché per renderla accessibile, ma soprattutto per elaborare automaticamente delle statistiche sul grafo in esame, è necessaria l'introduzione di un modulo di analisi.

Le funzioni di cui deve essere primariamente dotato consistono nella produzione di elenchi di nodi o archi, questi ultimi con tutte le rispettive etichettature (orientazione, path covering, ecc.).

Inoltre, con questo modulo deve essere possibile elaborare delle medie di insieme che siano particolarmente significative. Per esempio, esso deve consentire l'elaborazione di distribuzioni relative alle dimensioni delle componenti fortemente connesse, oppure al numero di oscillazioni nell'orientazione che ogni arco ha subito.

Infine, poiché in questo contesto spesso risulta utile soffermarsi sulla struttura delle componenti fortemente connesse di dimensioni relativamente grandi (per esempio, per studiare il nucleo di Internet), un'altra funzione utile è quella di consentire l'isolamento di una siffatta porzione salvandola su file. Successiva-

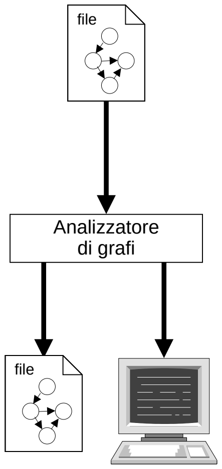

Figura 3.4: Funzionamento del modulo per l'analisi dei grafi

mente, essa risulta accessibile a tutti i moduli dell'ambiente, esattamente come se fosse stata generata *ex novo* dal componente di generazione dei grafi.

Tutti questi comportamenti sono rappresentati in figura 3.4, in cui si evidenzia come sia possibile, con questo modulo, produrre sia risultati statistici (su terminale o su file), sia nuovi grafi.

Estrattore di AS path Uno dei problemi di cui si è parlato in fase di analisi consiste nell'eventualità che un algoritmo di inferenza delle relazioni possa produrre risultati insensati qualora i di dati in ingresso presentino qualche forma di anomalia (per esempio, ritorni a capo in un record di una tabella BGP). Un modo per evitare questo consiste nel modificare tutte le implementazioni esistenti di questi algoritmi per renderle più robuste rispetto a simili eventualità. Tuttavia ciò richiederebbe, oltre ad un impiego non trascurabile di risorse per agire sulle varie implementazioni, anche la disponibilità ad effettuare lo stesso tipo di intervento su eventuali strumenti prodotti in tempi successivi.

Evidentemente, un approccio del genere non è applicabile vantaggiosamente. Di conseguenza si è scelto di dotare l'ambiente TORQUE anche di un modulo di preprocessamento dei dati da elaborare.

Poiché gli algoritmi di inferenza si basano su insiemi di AS path, questo componente, come rappresentato in figura 3.5, ha la funzione fondamentale di estrarre tale informazione da tabelle BGP. Tuttavia, oltre a dover riconoscere le eventuali differenze di formato con cui esse vengono mostrate dai vari router, il processo di estrazione degli AS path deve anche permettere di effettuare su di essi alcune elaborazioni fondamentali:

- inserimento di un numero identificativo di AS in testa ad ogni AS path; generalmente, infatti, i router inseriscono nei cammini il numero dell'AS di cui fanno parte soltanto all'atto di esportare una rotta;
- eliminazione del *prepending*; con questo nome si identifica una pratica abbastanza diffusa, derivante dall'adozione di opportune politiche nella

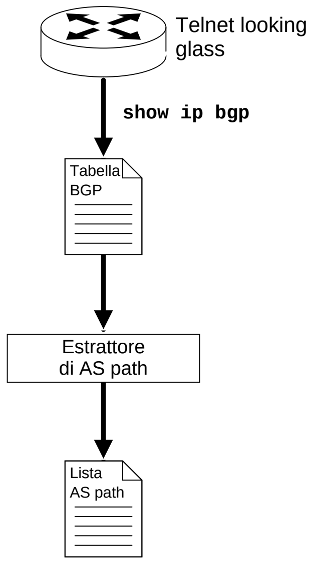

Figura 3.5: Funzionamento del modulo di estrazione degli AS path

configurazione del protocollo BGP, secondo cui un router A inserisce nei cammini il proprio numero di AS più di una volta; questo viene in genere fatto per influenzare il processo decisionale dei router B_i che ricevono l'annuncio, in modo da scoraggiarli dallo scegliere come percorso per recapitare i dati proprio quello passante per A;

- eliminazione degli AS privati; i numeri di AS nell'intervallo $64512 \leq ASn \leq 65534$ sono riservati per uso privato, e, come tali, non sono univoci in tutto il mondo; è opportuno, pertanto, rimuoverli prima di effettuare l'inferenza;
- eliminazione degli *AS set*; un *AS set* è una porzione speciale di un AS path, in cui l'ordine con cui appaiono gli AS non corrisponde al percorso effettivamente seguito dagli annunci; un AS set viene introdotto quando un router scelga di aggregare più rotte in un'unico prefisso: in tal caso, infatti, lo spazio di indirizzamento ottenuto risulta annunciato non da un solo AS, ma da un insieme di AS;
- eliminazione dei cicli; sebbene il funzionamento del protocollo BGP garantisca lo scarto di un annuncio ricevuto qualora venga rilevato che l'informazione da esso veicolata introdurrebbe un ciclo nell'AS path, questa situazione può ugualmente verificarsi, per esempio a causa di errori di battitura nella configurazione di un router; in figura 3.6 viene mostrato l'esempio più frequente di questo tipo di anomalia.

3.2.2 Architettura complessiva

Come già anticipato, i vari moduli dell'ambiente TORQUE possono essere utilizzati in maniera indipendente l'uno dall'altro. Tuttavia, risultano tutti forte-

Prepending

`2497 7518` $\boxed{\texttt{10002 10002 10002 10002}}$

`2497 7518` $\boxed{\texttt{10002 10002 10002 10002}}$

Cicli

`6453 8297 3249` $\boxed{\texttt{3249 3294 3249}}$

`6453 8297 3249` $\boxed{\texttt{3249 3294 3249}}$

`6453 8297 3249` $\boxed{\texttt{3249 3294 3249}}$

`6453 8297 3249` $\boxed{\texttt{3249 3294 3249}}$

AS set

`7018 19244 11664` $\boxed{\{\texttt{20305,20305,20305,20305,17401}\}}$

`7018 19244 11664` $\boxed{\{\texttt{20305,20305,20305,20305,17401}\}}$

AS privati

`7018 568 721 765` $\boxed{\texttt{65010}}$

`7018 568 721 765` $\boxed{\texttt{65010}}$

Figura 3.6: Esempi di anomalie negli AS path

mente integrati grazie all'utilizzo di un formato comune nei file che essi gestiscono. La loro interazione può essere sufficientemente articolata da consentire l'ottenimento di molti tipi di risultati.

In figura 3.7 viene mostrato uno schema dei possibili percorsi che ciascun dato può compiere. L'utilizzo tipico consiste nei seguenti passi:

1. si preleva una tabella BGP da un looking glass di tipo telnet, mediante il comando `show ip bgp`;

2. si estraggono gli AS path utilizzando l'apposito modulo di estrazione, opportunamente configurato secondo le esigenze;

3. si esegue un algoritmo di inferenza sui cammini così ricavati, ottenendo in tal modo un'orientazione per il grafo degli AS;

4. si utilizza il modulo di generazione dei grafi per immagazzinare in un file una struttura dati che contenga anche informazioni sul covering degli archi (a tal fine sono necessarie anche le informazioni prelevate dalla tabella BGP stessa e dalla lista degli AS path);

5. si effettua uno studio dei risultati generati, il quale può seguire una o più delle seguenti strade alternative:

   - elaborazione del grafo ottenuto con il modulo analizzatore, il quale può a sua volta produrre un nuovo grafo (per esempio, la componente fortemente connessa più grande del grafo di origine) che può essere oggetto di un ulteriore esame;
   - generazione di un grafo differenziale su una coppia di risultati ottenuti dall'esecuzione di due algoritmi (o da due istanze dell'esecuzione di

uno stesso algoritmo), il quale può essere poi esaminato con il modulo analizzatore;

- generazione di un grafo differenziale complessivo, che consenta di tracciare un profilo dell'evoluzione delle relazioni assegnate; anch'esso può poi essere studiato con il modulo analizzatore.

Nella figura 3.7 sono racchiusi in rettangoli tratteggiati i componenti dell'ambiente TORQUE; si noti come, nell'architettura complessiva, la loro posizione sia notevolmente spostata verso il basso, e cioè nel punto in cui i dati prodotti dagli algoritmi di inferenza sono pronti per essere analizzati. Si osservi inoltre come il sistema non sia restrittivo nei confronti degli algoritmi utilizzati: in altre parole, è possibile utilizzare algoritmi diversi per il calcolo delle orientazioni, e mettere poi a confronto i risultati ottenuti sempre utilizzando questi stessi strumenti.

3.3 Realizzazione dell'ambiente

In sede di realizzazione si è cercato di tenere il più possibile presenti i requisiti individuati in fase di analisi. In questa sezione, oltre ad illustrare e giustificare le scelte che hanno guidato la programmazione, vengono anche analizzate le implementazioni dei quattro moduli, cercando di fornirne una caratterizzazione comportamentale di tipo ingresso-uscita.

3.3.1 Scelte operate nell'implementazione

La prima scelta che è stato necessario effettuare per procedere con l'implementazione dei componenti è stata relativa al linguaggio da utilizzare. La struttura spiccatamente modulare avrebbe, di per sé, consentito la realizzazione di molte parti utilizzando soltanto dei linguaggi di scripting (ad esempio `python`); tuttavia, procedendo in questo modo si sarebbe inevitabilmente persa gran parte della flessibilità di cui l'ambiente deve essere dotato. Pertanto, si è deciso di utilizzare un vero e proprio linguaggio di programmazione. La scelta si è fondamentalmente ristretta ai due linguaggi attualmente più utilizzati per la scrittura di applicazioni: C/C++ e Java. Lo sviluppo dell'ambiente sarebbe dovuto avvenire su piattaforma Linux, di fatto molto utilizzata nell'ambito del networking per via della notevole ricchezza di componenti di gestione della rete e della sua notevole robustezza. Ciò ha escluso la possibilità di ricorrere ad ambienti quali Visual Basic o Visual C++. Poiché nell'ambito dell'inferenza delle relazioni tra AS molti strumenti sono scritti utilizzando il C++, la scelta è infine ricaduta proprio su quest'ultimo, in modo tale da facilitare l'integrazione dei nuovi strumenti con quelli esistenti.

Molte delle funzioni di cui deve essere dotato TORQUE sono più o meno disponibili in librerie preesistenti, di cui non tutte disponibili gratuitamente od installate con i compilatori più utilizzati. L'esigenza di dotare l'ambiente di un buon grado di portabilità ha dunque portato a scartare l'idea di utilizzare tali librerie. In realtà, alla stessa conclusione si è pervenuti anche (e soprattutto) considerando il requisito di chiarezza: poiché il codice di queste librerie non è sempre disponibile o facilmente accessibile, e nella documentazione alcuni comportamenti risultano talvolta poco trasparenti, il rischio cui si sarebbe andati

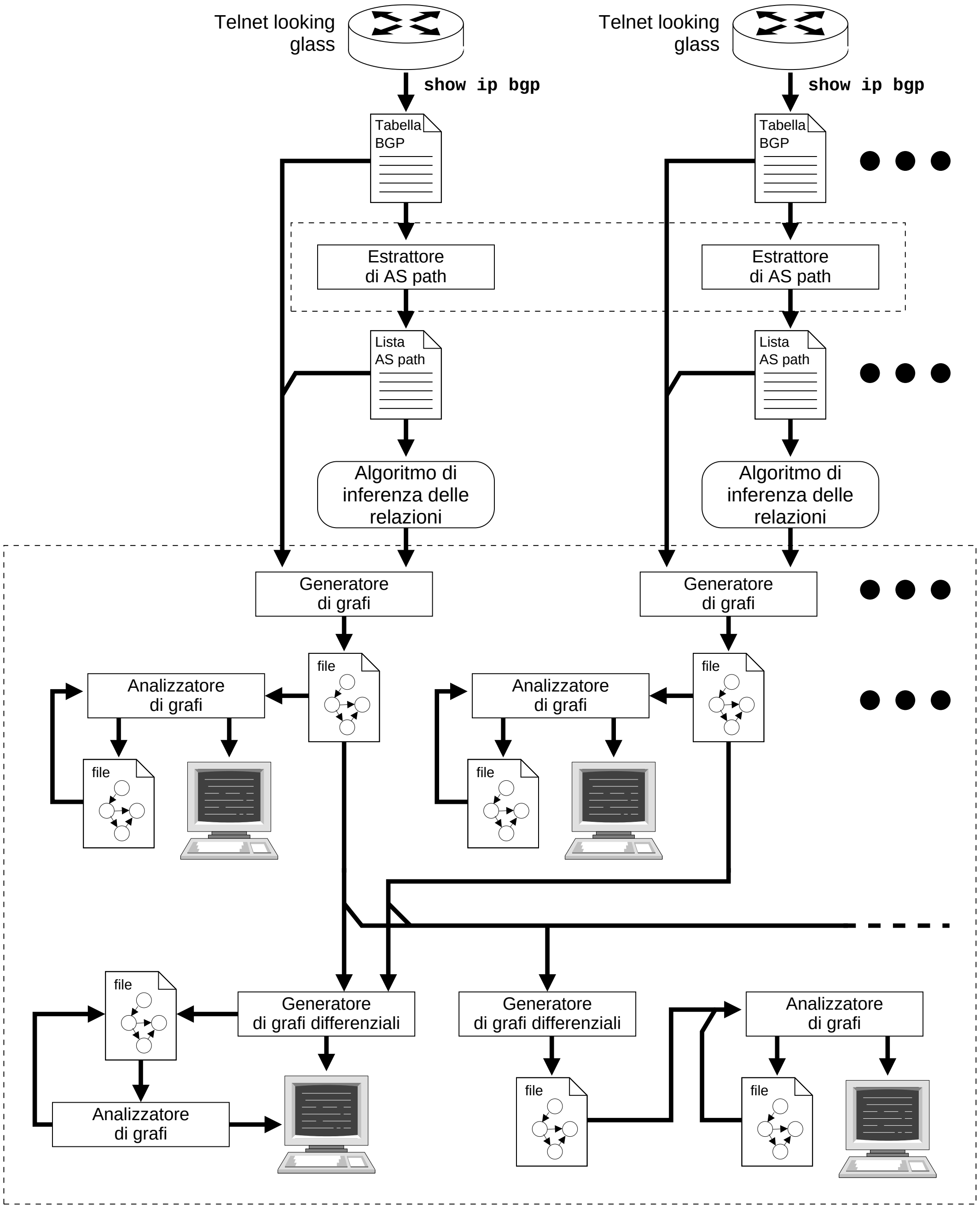

Figura 3.7: Percorso dei flussi informativi nell'ambiente TORQUE

incontro utilizzandole sarebbe stato quello di produrre strumenti di per sé corretti, ma basati su un utilizzo improprio delle loro funzioni: evidentemente, il risultato finale sarebbe stato un ambiente poco affidabile, o, addirittura, inutilizzabile. Per queste ragioni, molte delle funzioni necessarie (specialmente per quello che riguarda i grafi) sono state riscritte, e non sono state utilizzate librerie oltre quelle oramai davvero considerate come standard (`stdio.h`, `string.h`, `math.h`, `stdlib.h`, ecc.).

L'implementazione è stata fortemente orientata ad ottenere un sistema correttamente funzionante, più che estremamente efficiente (per quanto, poi, il prodotto finale sia comunque abbastanza veloce). A tal fine, in molte fasi intermedie sono state operate delle verifiche volte a riscontrare errori di vario genere: di battitura nella scrittura del codice, di gestione delle strutture dati, ecc.

Il codice scritto è fortemente parametrico: molte classi sono, infatti, disponibili sotto forma di *template*, con l'obiettivo di renderle il più possibile riutilizzabili per altri scopi. Le strutture dati utilizzate per rappresentare internamente i grafi si basano su alberi AVL[1] e su liste doppiamente linkate. Molte funzionalità (come il parsing dei file) sono state isolate in file separati, per consentirne una modifica più agevole.

Per soddisfare ulteriormente il requisito di chiarezza, molte porzioni del codice sono ampiamente commentate, cercando, ove possibile, di rendere chiaro l'utilizzo delle variabili, il significato degli argomenti delle funzioni ed i risultati da esse prodotti.

Tutti i messaggi forniti come output ed i commenti all'interno del codice sono scritti in lingua inglese, per ampliare ulteriormente la fruibilità dell'ambiente.

Il software è dotato di un *makefile* che ne facilita la compilazione, consentendo di scegliere di quali moduli generare la versione eseguibile. La compilazione stessa non richiede accorgimenti particolari, ed è stata effettuata con successo utilizzando l'ambiente `g++` fornito con le distribuzioni di Linux Red Hat 7.2 e 7.3, quest'ultima dotata della versione 2.96 20000731 (Red Hat Linux 7.3 2.96-110) di `g++`.

3.3.2 Descrizione del funzionamento dei moduli

Le tabelle 3.1, 3.2, 3.3 e 3.4 mostrano gli input di cui ciascun modulo ha bisogno, le elaborazioni che esso effettua e gli output che esso produce. Per maggior completezza, dopo ciascuna tabella viene riportata una copia dell'aiuto in linea di cui ogni modulo è dotato.

Nell'illustrazione delle funzionalità dei moduli si fa spesso riferimento all'orientazione degli archi parlando di "archi non orientati" (*unoriented edges*), "archi orientati" (*oriented edges*), "archi diretti" (*directed edges*), "archi non diretti" (*undirected edges*). Il concetto di orientazione è generalmente riferito alla significatività della direzione assunta dall'arco: se un arco non è orientato, vuol dire che non ha proprio nessuna orientazione (per esempio, non è stato orientato con successo dall'algoritmo di inferenza); il fatto di essere "diretto" è invece indice della presenza di una relazione customer-provider: un arco non diretto, e, tuttavia, orientato, rappresenta un arco di peering.

[1] Dal nome di *Adelson, Velskii, Landis*. Si tratta di particolari alberi binari di ricerca in cui due sottoalberi hanno sempre profondità che differisce al più di un livello.

TmakeGraph

INPUT
Uno o più dei seguenti file: ▶ una tabella BGP nel formato prodotto dall'esecuzione del comando `show ip bgp` su un looking glass di tipo telnet; ▶ il risultato del calcolo di un'orientazione effettuato utilizzando l'implementazione dell'algoritmo presentato nella sezione 2.4 disponibile sotto il nome di bgpPathSat; ▶ il risultato del calcolo di un'orientazione ottenuto utilizzando l'algoritmo presentato nella sezione 2.3, pubblicamente disponibile in [9] (file `as.relation.gz`); ▶ una lista di archi che l'algoritmo di inferenza descritto nella sezione 2.3 non è riuscito ad orientare, disponibile pubblicamente in [9] (file `aspairs.unknown.gz`); ▶ una lista semplice di AS path, in cui ogni riga è un AS path e su di essa compaiono esclusivamente gli AS che fanno parte del cammino; ▶ un grafo già esistente, nel quale inserire nuovi elementi.

ELABORAZIONI EFFETTUATE
Costruzione (o arricchimento) di una struttura dati interna che rappresenti un grafo, effettuata eseguendo i seguenti passi, nell'ordine in cui vengono elencati: ▶ aggiunta al grafo dei nodi prelevati dai file specificati dall'utente; ▶ aggiunta al grafo degli archi (non orientati) prelevati dai file specificati dall'utente; ▶ orientazione dei soli archi già presenti nel grafo secondo quanto indicato nei file specificati dall'utente; ▶ etichettatura dei soli archi già presenti nel grafo con informazioni sul path covering ottenute da file specificati dall'utente; ▶ etichettatura dei soli archi già presenti nel grafo con informazioni sull'IP covering, ottenute da tabelle BGP specificate dall'utente; l'IP covering viene calcolato nel modo seguente: assumendo (come in effetti è) che le tabelle BGP siano ordinate in base al prefisso, si considera quest'ultimo, e si aggiunge il numero di indirizzi IP che esso racchiude a tutti gli archi presenti negli AS path sui quali esso è stato annunciato; contemporaneamente, si marca ognuno di questi archi con tale prefisso; quando si passa al prefisso successivo, per ogni arco presente negli AS path su cui è stato annunciato, si calcola l'intersezione dello spazio da esso rappresentato con quello corrispondente all'ultimo prefisso annunciato su quell'arco; il valore ottenuto è il numero di indirizzi "nuovi" annunciati su quell'arco, e come tale viene aggiunto al totale degli indirizzi annunciati su quest'ultimo; se richiesto, durante l'estrazione dei cammini dalla tabella BGP (necessaria per dedurre quali archi aggiornare), si può anteporre ad essi un numero di AS specificato dall'utente; ▶ marcatura dell'intero grafo con i nomi dei file da cui è stato generato e con un commento specificato dall'utente.

OUTPUT
Un file in formato ASCII proprietario che rappresenta tutte le caratteristiche del grafo elaborato. Il file può essere processato con uno qualsiasi degli strumenti di TORQUE, compreso questo stesso modulo (nel caso in cui si voglia arricchire un grafo esistente).

Tabella 3.1: Caratterizzazione ingresso-uscita del modulo di generazione dei grafi

```
tmakeGraph 1.0
TORQUE toolkit AS graph builder
USAGE: ./tmakeGraph [options] [[-o] <outFile>] {--file:{bBsvpunedci} <filename>...}

Input files must be provided using --file option, in the following way:
  --file:  this preamble is used to indicate that a file name will follow;
           each file must be identified by its format (BGP dump, list of
           AS paths, etc.) and the contents that must be extracted from it
           (nodes, edges, etc.)

  {bBsvpu} each `--file:' option must be followed by exactly one of these
           letters, which identify file format as follows:
             b,B  file is a BGP dump (see below for an explanation of the
                  difference between 'b' and 'B')
             s    file is the output of a bgpPathSat computation
             v    file is a relationship file (i.e., each line looks as
                  follows: `<AS1> <AS2> <flag>'; when flag is 1, AS1 is
                  supposed to be a provider and AS2 a customer; if flag
                  is 0, AS1 and AS2 have a peering relationship)
             p    file is a plain list of AS paths (i.e., paths must
                  neither be groupped by number of occurrences nor tagged)
             u    file is a list of couples of ASs whose corresponding
                  edges could not be assigned a direction

  {nedci}  each `--file:' option must also be followed by at least one of
           these letters, which identify what to get from current file:
             n    get graph nodes from current file; do not use if file
                  format is 'b'
             e    get graph edges (but not their direction) from current
                  file; edges are inserted in the graph only if they do
                  not exist yet; otherwise, they are left unchanged; do
                  not use if file format is 'b'
             d    get direction of already existing edges from current
                  file; do not use if file format is either 'b' or 'u'
             c    get path covering (number of paths that traverse each
                  edge) from current file; do not use if file format is
                  any one of 'b', 'v' or 'u'
             i    get IP address covering (number of IP addresses that are
                  announced over each edge) from current file; only use if
                  file format is 'b'

  <filename>  the name of the current input file


Option --file can be used with more than one content per file; this allows
to optimize file access, by avoiding that each file is read more than
once; anyway, regardless of the order in which files are provided on
command line, they are always read in the following order:
nodes files, edges files, edge directions files, path covering files, IP
address covering files.
Minimal input consists of a file to get nodes from.

When using option --file to feed more than one file with the same
contents, the result is that such contents are merged together (e.g., all
the nodes from all the files are inserted in the graph - of course, with
no duplicates... -, all the IP addresses found in all the BGP dumps are
supposed to be exchanged over the appropriate edges - without considering
addresses more than once -, etc.).

File ordering is significant; in fact, if (for example) two node files A
and B are provided, and A is also an edge file, reading A before B implies
that none of the edges between nodes in B (but not in A) will be inserted.
The only exception are the BGP dumps: since they are ordered according to
the value of the network, the sequence in which they are provided on the
command line can be arbitrary.

If the AS paths inside the BGP dumps do not include receiving vantage
point number, it is possible to use 'B' instead of 'b'; by doing this,
following parameters are expected to be the number of the AS that should
be prepended to each path and the name of the file containing the BGP dump.

BGP dumps are processed as follows: no check is performed for both cycles
and private ASs; AS sets are just skipped; prepending is automatically
skipped because AS graphs (usually...) do not have self-edges; incomplete
paths are considered as they appear.


Options are:
```

```
  --comment <string>  Insert a comment string inside the graph (e.g., to
                      make it easier to understand which data set it comes
                      from); by default, graph is also labelled with the
                      names of the files it was built from
  --enhance <filename>  Read an existing graph file and perform some sort
                        of update over it; when using this option, input
                        files can be of any type (i.e., minimal input is
                        not a nodes file any more); it is also possible to
                        give no input files if, for example, one just
                        wants to update graph comment; this is the only
                        file name that can be replaced with '-' if input
                        graph is provided on standard input
  -f      Don't prompt user (default behaviour) before overwriting
          existing files
  -v      Be verbose: report all errors that occur during file parsing
  --help  Show this help


<outFile> is the name of the output file graph. If none is given, the
graph will be dumped to stdout. Use -o option when ambiguities may arise
between file name and other command line options.
```

TdiffGraph

INPUT
Almeno due file contenenti rappresentazioni di grafi prodotte con altri strumenti di TORQUE.

ELABORAZIONI EFFETTUATE
Dipendenti dal numero di file forniti in ingresso: ▶ se i file sono due, l'utente può scegliere di effettuare una delle seguenti operazioni: • richiedere un confronto quantitativo di carattere generale (numero di nodi/archi solo in un grafo o in entrambi, numero di archi customer-provider, di peering e non orientati, numero di archi orientati concordemente/discordemente); • richiedere la produzione di una lista dei nodi presenti in uno solo dei grafi o in entrambi; • richiedere la costruzione di un grafo *differenziale* o *di sovrapposizione*, ossia un grafo in cui sia inserito un arco non orientato (e, se necessario, i suoi nodi terminali) ogniqualvolta tale arco esista in entrambi i grafi di partenza e siano soddisfatte delle condizioni specificate dall'utente (relative alla sua orientazione ed al suo peso); ogni arco inserito viene inoltre etichettato con gli attributi (orientazione e pesi) che aveva nei due grafi in input; • richiedere la costruzione di un grafo *complementare*, ovvero un grafo che abbia soltanto gli archi (ed i nodi alle loro estremità) presenti in uno solo dei grafi in input e che soddisfano delle condizioni specificate dall'utente (sulla sua esistenza ed orientazione nei due grafi); gli archi in esso inseriti appaiono (come orientazione e come pesi) esattamente come nel grafo da cui sono stati estratti; ▶ se i file sono più di due ha luogo la costruzione di un grafo (detto *multidifferenziale*) avente tutti gli archi presenti in almeno uno dei grafi in ingresso ed i relativi nodi terminali; ciascun arco è etichettato con delle informazioni sulla stabilità dell'orientazione ad esso assegnata (numero di grafi in input in cui l'arco risulta visibile, numero di cambiamenti nell'orientazione –inclusi i passaggi da uno stato orientato ad uno non orientato–, durata media di un'orientazione). In entrambi i casi l'eventuale grafo prodotto viene poi etichettato con i nomi dei file da cui è stato generato e con un commento fornito dall'utente.

OUTPUT
Dipendente dal numero di file forniti in ingresso: ▶ se i file sono due, può trattarsi di una delle seguenti informazioni: • statistiche numeriche sulle differenze tra i due grafi; • un grafo complementare, in formato utilizzabile con tutti gli strumenti di TORQUE; • un grafo differenziale, in formato utilizzabile con tutti gli strumenti di TORQUE; ▶ se i file sono più di due, si tratta di un grafo multidifferenziale, utilizzabile con tutti gli strumenti di TORQUE.

Tabella 3.2: Caratterizzazione ingresso-uscita del modulo di generazione di grafi differenziali

```
tdiffGraph 1.0
TORQUE toolkit AS graph compare tool
USAGE: src/tdiffGraph [options] [mode] [[-o] <outFile>] {[-s] <filename>...}

By default, output shows a series of statistics about the differences
between the two provided graphs. Anyway, by using any of the modes listed
below, output can be changed.

Mode can be at most one of:
  -l       Show a list of the nodes that appear only in the first graph,
           only in the second one and in both of them; the first input
           file name on the command line is assumed to be the 'first'
           graph, while the other file name corresponds to the 'second'
           graph; output will be a list of node numbers, each one preceded
           by a '<', '>' or '|' symbol if, respectively, the node is found
           to be present only in the first, only in the second or in both
           graphs

  --over:<conds>  Build an 'overlap' or 'differential' graph, which is a
                  graph that has an edge (and, of course, its two terminal
                  nodes) whenever the latter is found to exist in both the
                  source graphs and to satisfy a series of conditions that
                  are listed in the <conds> part of the option; <conds> is
                  a list of comma separated combinations of the characters
                  listed below; each item of this list is logically "or"ed
                  with the others, while conditions within a single item
                  must hold simultaneously (logical "and"); in other
                  words, <conds> represents a DNF formula. Available
                  conditions are:
                    a:     insert the edge anyway
                    o(O):  insert the edge if it is found to be oriented
                           in the first (second) graph
                    u(U):  insert the edge if it is found to be unoriented
                           in the first (second) graph
                    x:     insert the edge if it is directed in both
                           graphs and the two directions are opposite
                    y:     insert the edge if it is oriented in both
                           graphs and the two orientations are different
                    p(P):  insert the edge if it is found to be a peering
                           edge in the first (second) graph
                    d(D):  insert the edge if it is found to be a directed
                           edge in the first (second) graph
                    w:     insert the edge if its weight in the two graphs
                           differs
                  Overlap graph is made up of unoriented edges only.

  --comp:<conds>  This option is very similar to the previous one, but
                  this time edges are inserted whenever they only exist
                  in one of the graphs (that is why the generated graph
                  is called 'complementary') and they have the properties
                  listed in <conds>; the format of the <conds> list is
                  identical to that of the previous option, but the
                  conditions themselves are a little different:
                    a(A):  insert the edge if it just exists in the first
                           (second) graph
                    u(U):  insert the edge if it is an unoriented edge in
                           the first (second) graph
                    o(O):  insert the edge if it is an oriented edge in
                           the first (second) graph
                    d(D):  insert the edge if it is a directed edge in the
                           first (second) graph
                    p(P):  insert the edge if it is a peering edge in the
                           first (second) graph
                  Unlike the overlap graph, the complementary graph is
                  made up of edges which appear exactly as they did in
                  the source graph.

  --multi  This option requires at least two input files, but there is no
           upper limit to their number. Files are read one after the
           other, in the order in which they are provided on command
           line. For each of the edges in the input graphs, an unoriented
           edge is inserted in a so called 'multi-differential' graph; if
           such edge is found to already exist in the multi-differential
           graph, its attributes are just updated. Attributes concern
           statistics about the "history" of that edge (e.g., the
           number of graphs in which it was found to be unoriented). The
           returned graph is the so modified multi-differential graph
           (file format is the same as plain differential graphs).
```

```
Options are:
  -n  Negate conditions for --over and --comp options (i.e., insert edges
      if the conditions in the <conds> list do NOT hold)
  -f  Do not ask user confirmation (default behaviour) when overwriting
      existing files
  --comment <string>  Insert into the graph a comment string; by default,
                      graphs are labelled with the name of the files they
                      were generated from
  --help  Show this help


<filename> is the name of an input file (a graph generated with
tmakeGraph); use -s option to force following argument to be a source file
name whenever ambiguities may arise with other options on the command
line. Input files must be exactly 2, unless --multi option is used (see
above). Any of the input file names can be replaced with a dash ('-'), in
which case input for the corresponding graph will be taken from standard
input instead of a file.

By using -o option, any kind of output (but status messages) can be
directed to a file.
```

TviewGraph
INPUT
▶ Un grafo oggetto dell'analisi; ▶ opzionalmente, un file contenente comandi di scripting utilizzati per effettuare le analisi richieste dall'utente.
ELABORAZIONI EFFETTUATE
Dipendenti dai comandi impartiti dall'utente. Può trattarsi di una combinazione qualunque delle seguenti operazioni: ▶ produzione di un elenco dei nodi o degli archi del grafo, eventualmente con alcuni degli attributi (path covering, IP covering, stabilità dell'orientazione degli archi) con cui sono etichettati; ▶ produzione di distribuzioni relative alle dimensioni delle componenti connesse e fortemente connesse, al path covering, all'IP covering o alla stabilità delle orientazioni degli archi; ▶ estrazione di una componente connessa o fortemente connessa. Nel caso in cui il grafo in esame sia differenziale, tutti gli elenchi contengono informazioni doppie, relative ai due grafi dal cui confronto esso è stato generato. Se si tratta invece di un grafo multidifferenziale, è possibile consultare anche la cronologia dell'orientazione di ogni arco, in termini di numero di grafi in cui ha assunto ogni possibile orientazione.
OUTPUT
Dipendente dai comandi impartiti dall'utente. Può trattarsi di statistiche in formato testuale, oppure di un grafo prodotto dall'estrazione di una componente connessa o fortemente connessa (utilizzabile con tutti gli strumenti di TORQUE).

Tabella 3.3: Caratterizzazione ingresso-uscita del modulo analizzatore di grafi

```
tviewGraph 1.0
TORQUE toolkit graph analysis tool
USAGE: ./tviewGraph [options] [[-s] <scriptFile>]

Options can be:
  -f       Do not ask user (default behaviour) when overwriting an existing
           file
  --help   Show this help


<scriptFile> is the name of a file storing a sequence of tviewGraph commands;
omitting it will result in either entering an interactive mode or commands
being taken from standard input (if it has been redirected). More information
about available scripting commands can be obtained by entering interactive
mode (just use no arguments to achieve this). Use -s option when ambiguities
may arise between file name and other command line options.
```

Aiuto sui comandi di scripting

```
help

Available commands are:
exit quit dump show help load clear echo save
Commands can be truncated, if this does not arise ambiguities (e.g.,
‘d g n’ is the same as ‘dump graph nodes’)

Type ‘help <command>’ to get more help about a command.
```

```
help exit

Both ‘exit’ and ‘quit’ are used to clean up memory and abandon the
program. Command syntaxes are:

exit
quit
```

```
help dump

Dump command is used to get a list of items inside a graph or its
portion. Command syntax is:

dump <object> <property>

where

 <object> is the area to be dumped, and can be one of:

  graph   tells dump to work on the entire previously loaded graph
  scc <n> tells dump to work on the strongly connected component of
          index <n>
  cc <n>  tells dump to work on the connected component of index <n>

 <property> is the type of content to be listed, and can be one of:

  nodes         is used to get a list of the nodes; output is a plain
                list of AS numbers
  edges         is used to get a list of the edges; output is in the
                form
                   A ddd B
                where A and B are the two terminals of the edge, and ddd
                can be one of the following:
                   --- edge has not been assigned an orientation
                       (unoriented)
                   <-- edge is directed from B to A
                   --> edge is directed from A to B
                   <-> edge is a (bi-directed) peering edge
  ipcovering    is used to get a more detailed list of the edges,
                where for each however directed edge A B, the number of
                IP addresses that were announced over it is reported;
                output is a list of edges with covering information
                following each edge: the first value is the number of
                addresses that have been announced from A to B, while
```

```
                the second one is the number of addresses that have
                been announced from B to A; for each value, also its
                corresponding "netmask" coverage is reported
  pathcovering  is used to get a list of the edges together with the
                number of paths they appeared in; for each however
                directed edge A B, two values are reported: the first
                one is the number of appearances of the adjacency A B
                among the paths; the second one is the number of
                appearances of the reversed adjacency B A among the
                same paths
  differences   is used to obtain information about edges' orientation
                history; when using this mode, the plain list of the
                edges is enriched with a set of values, which, from
                the leftmost to the rightmost, have the following
                meaning:
                   - number of compared graphs the edge appeared in
                   - number of changes in the orientation of the edge
                     (including transitions from an oriented state to
                     an unoriented one and vice versa)
                   - mean edge orientation firmness (i.e., the mean
                     number of subsequent graphs in which the edge
                     kept its orientation)
                   - the following values represent the number of
                     times the edge appeared with a given orientation
                     (see above for the meaning of the various
                     orientation symbols)

When dealing with differential graphs, most values (including edge
orientations) are reported twice, meaning that the first group of
values represents the first of the compared graphs, while the second
one represents the second of the compared graphs.
```

```
help show

The 'show' command can be used to get cumulative information about
graph properties. Command syntax is:

show <object> <property> [<diffProperty>]

where

 <object> is the area to work on, and can be one of:

  graph          tells show to work on the entire previously loaded
                 graph
  scc <n>        focuses the attention on the strongly connected
                 component of index <n>
  cc <n>         tells show to use the connected component of index
                 <n> as its target
  anyscc         considers every strongly connected component (use it
                 if you wish to print a distribution of the strongly
                 connected components size)
  anycc          works the same as 'anyscc', but instead focuses on
                 connected components

 <property> is the entity whose size (distribution) should be reported,
 and can be one of:

   information   which is the only "strange" kind of property; use
                 this keyword to print general information concerning
                 the graph
   nodes         which just reports the number of nodes in the selected
                 portion of the graph
   edges         which reports the number of edges
   index         which lists the indexes of the (S)CCs, together with
                 their size; this keyword can only be used when target
                 is either 'anyscc' or 'anycc'
   ipcovering    if you want to get a distribution of the number of
                 addresses that have been advertised over each edge
                 (regardless of the direction in which they were announced
                 over it)
   pathcovering  to print a distribution of the number of paths each
                 edge appeared in (regardless of the order in which its
                 end ASs appeared in the paths)
   differences   in order to unlock a new series of keywords which can
                 be used to get information about graph comparisons
```

```
                 results

 <diffProperty> is one of the following differential properties (use
 this argument only when <property> is 'differences'):

   attendances   shows a distribution of the number of compared graphs
                 each edge appeared in
   directed      shows a distribution of the number of times each edge
                 appeared directed with respect to its attendances
   undirected    the same as 'directed', but for undirected instances
   oriented      the same as 'directed', but for oriented instances
   unoriented    the same as 'directed', but for unoriented instances
   changes       shows a distribution of the number of orientation
                 changes each edge has undergone, with respect to its
                 attendances
   duration      shows a distribution of the mean duration of an
                 orientation - number of subsequent graphs in which the
                 edge retained the same orientation - with respect to
                 the edge's attendances
```

```
help load

The 'load' command reads a graph file and stores it in memory after
having computed its connected components. From that moment onwards,
the graph is available for processing by means of the other commands.
Command syntax is:

load <filename>

where <filename> is the name of the graph file to be loaded
```

```
help clear

Use the 'clear' command to release memory by deleting the graph you
are currently working on. Command syntax is:

clear
```

```
help echo

This command can be used both to switch on/off command echoing and to
display a user message. Command syntax is:

echo [-n|-off|-on] <message>

where:
 -n   tells echo not to insert a newline character after displaying
      <message>
 -off tells echo to disable command echoing (no message must be given)
 -on  tells echo to enable command echoing (no message must be given)
```

```
help save

The 'save' command allows to isolate a (strongly) connected component
of the currently processed graph and to save it to a file (which can
then be loaded for processing). Command syntax is:

save {scc <n>|cc <n>} <filename>

where:
  scc and cc are used to specify which (strongly) connected component
             to work on
  <filename> is the name of the file to save the extracted component
             to
```

TpathExtract

INPUT
Una tabella BGP nel formato prodotto dall'esecuzione del comando `show ip bgp` su un looking glass di tipo telnet.

ELABORAZIONI EFFETTUATE
Una tra le seguenti operazioni in corrispondenza di ognuna delle seguenti caratteristiche che si possono manifestare in un AS path. ▶ Operazioni sui cammini: • nessuna operazione: la caratteristica non viene in alcun modo rimossa dall'AS path; • riparazione: per cammini con prepending, corrisponde compressione di più occorrenze consecutive di uno stesso numero di AS in un'unica occorrenza; per cammini con AS privati o AS set corrisponde all'eliminazione di questi ultimi; per cammini con cicli corrisponde all'eliminazione di ogni occorrenza di un numero di AS tranne la prima; • cancellazione: ogni AS path che presenti la caratteristica viene cancellato dall'output principale; • salvataggio: ogni AS path che presenti la caratteristica viene salvato in un file esterno; • marcatura: ogni AS path che presenti la caratteristica viene marcato con una lettera che la rappresenta. ▶ Caratteristiche dei cammini: • completezza: corrisponde all'opportunità che un cammino sia marcato come *incomplete*; • presenza di AS set; • presenza di prepending; • presenza di AS privati; • presenza di cicli. Le caratteristiche vengono analizzate (e comportano l'esecuzione delle corrispondenti operazioni) nell'ordine in cui sono state elencate. Possono inoltre essere effettuate le seguenti operazioni: ▶ inserimento in testa a ciascun cammino di un numero di AS specificato dall'utente; ▶ marcatura di ogni cammino con i prefissi che sono stati annunciati su di esso; ▶ raggruppamento dei cammini in base al loro numero di occorrenze.

OUTPUT
Una lista di AS path ed un rapporto relativo alla loro estrazione. Se richiesto dall'utente, possono essere prodotti dei file esterni in cui vengono isolati i cammini con determinate proprietà.

Tabella 3.4: Caratterizzazione ingresso-uscita del modulo di estrazione degli AS path

```
tpathExtract 1.0
TORQUE toolkit AS path extractor
USAGE: ./tpathExtract [settings] [options] [[-s] <filename>]

Settings are:
  --group
      By default, output is a list of AS paths, including repetitions;
      using --group option reports each AS path once, together with the
      number of its occurrences in the BGP dump. When requested, the path
      is tagged with a merging of the tags associated with each instance
      of its. This means that the number of occurrences is just referred
      to the path itself, while the presence of a tag means that at least
      one of the occurrences had that property. If you wish to exactly
      know how many paths have that tag, either use -v option or save
      paths to external files

  --prefixes={show,allmask,unique,hide}
      show:    turns on prefix logging (i.e., for each path, a list of the
               prefixes that followed that path is reported); if --group is
               not used, a single prefix is reported for each AS path
      allmask: append a netmask to every network (even those without a
               netmask)
      unique:  report every prefix at most one time for each path
               (warning: a prefix might be reported more than one time
               even when using this option if it appears both with and
               without netmask in the source file); this mode should be
               used only in combination with --group
      Don't use:  show    with    hide
                  allmask without show
                  unique  without show

  --vpoint=always|auto|none
      none:   don't change AS paths
      always: prepend a user provided AS number to each AS path before
              processing it
      auto:   only prepend user provided AS if the first AS in the path is
              different
  When using either 'always' or 'auto', the following argument must be a
  valid AS number.

  The following settings all use the same modes:
      keep:   leave path unchanged
      fix:    remove feature from the path, but do not remove path
      delete: remove path from main output
      save:   also save (unchanged) path to file; when using this mode, the
              following parameter must be a file name
      tag:    tag the path using a letter
      Don't use:  keep   with any combination of the other modes
                  fix with delete or tag
                  delete with tag

  --incomplete={keep,delete,save,tag}
      Perform listed operations on paths marked as incomplete; if
      requested, paths are tagged with an 'I'

  --prepending={keep,fix,delete,save,tag}
      Perform listed operations on paths with prepending; using 'fix' also
      deletes "false" prepending from AS sets; if requested, paths are
      tagged with a 'P'

  --privas={keep,fix,delete,save,tag}
      Perform listed operations on paths with private ASs; use of 'fix'
      implies removal of private ASs; if requested, paths are tagged with
      a 'p'

  --sets={fix,delete,save,tag}
      Perform listed operations on paths with AS sets; 'fix' deletes the
      whole set from the path; 'save' cannot be used without either
      combination of the other modes; if requested, paths are tagged
      with a 'S'

  --cycles={keep,fix,delete,save,tag}
      Perform listed operations on paths with cycles; cycle checking is
      applied after all the other changes took place, and completely
      skips ASs in AS sets; using 'fix' deletes every instance of an
      AS in the path but the first; if requested, paths are tagged with
      a 'C'
```

```
Note: modes included in {} can be combined together, using comma as
separator; incompatible combinations are listed above; when using any of
the above options, corresponding built-in defaults are automatically
overridden (e.g., that is why you cannot use 'save' and 'keep' together)


Options are:
  -f      Don't ask user confirmation (default behaviour) when overwriting
          existing files
  -o <filename>  Save main output to file instead of printing it on stdout
  -v      Verbose mode (shows both parsing errors and final statistics);
          default mode is non-verbose
  --help  Show this help


<filename> is the name of a BGP dump file. Omitting it results in input
being taken from stdin.
Using of -s is suggested when filename may be confused with other command
line options.

Default behaviour corresponds to the use of the following settings/options:
--prefixes=hide --incomplete=keep --prepending=fix --privas=fix
--sets=fix --cycles=delete
```

3.4 Un esempio completo di utilizzo

Si supponga di avere a disposizione un'implementazione di un algoritmo per l'inferenza delle relazioni tra AS, e che la duplice esecuzione di tale algoritmo su dati diversi abbia prodotto due orientazioni contenute in due file di nome `finalAssignment.1` e `finalAssignment.2`. Si supponga inoltre che i cammini estratti dalle tabelle BGP utilizzate nelle computazioni siano contenuti in un file chiamato `all.paths`. Per confrontare le orientazioni con l'ambiente TORQUE è possibile eseguire i seguenti passi:

```
[max@gabbiano 2002-04-06]$ ~/tesi/src/tmakeGraph --file:pnec all.paths \
> --file:sd finalAssignment.1 --file:Bi 1 1 --file:Bi 15290 15290 \
> --file:Bi 1838 1838 --file:Bi 3549 3549 --file:Bi 3582 3582 \
> --file:Bi 3967 3967 --file:Bi 4197 4197 --file:Bi 5511 5511 \
> --file:bi 7018 --file:Bi 8220 8220 --file:Bi 8709 8709 -o final1.graph
tmakeGraph 1.0
Reading nodes, edges, path covering file (step 1 of 3)
'all.paths'
[****************************************]  (100.0 %)
Reading edge directions file (step 2 of 3)
'finalAssignment.1'
[****************************************]  (100.0 %)
Reading IP covering file(s) (step 3 of 3)
'1'
'15290'
'1838'
'3549'
'3582'
'3967'
'4197'
'5511'
'7018'
'8220'
'8709'
[****************************************]  (100.0 %)
Saving graph to file...
[max@gabbiano 2002-04-06]$
```

Questa operazione ha come risultato la produzione di un grafo i cui nodi ed archi siano ottenuti dall'analisi delle adiacenze contenute nel file `all.paths`, in cui gli archi siano orientati secondo quanto indicato nel file `finalAssignment.1`, ed in cui ciascun arco sia etichettato con un valore di path covering ricavato dai contenuti del file `all.paths` ed uno di IP covering calcolato dalle tabelle BGP utilizzate nella computazione.

Dopo aver eseguito un'operazione del tutto identica sul file `finalAssignment.2` è possibile, ad esempio, costruire un grafo che abbia un arco in corrispondenza di ogni orientazione trovata opposta nei due grafi:

```
[max@gabbiano tesi]$ ~/tesi/src/tdiffGraph --over:x -o diff.graph \
> final1.graph final2.graph
tdiffGraph 1.0
Reading graph file 'final1.graph'...
Reading graph file 'final2.graph'...
```

```
Computing differences...
Saving graph to file...
[max@gabbiano tesi]$
```

In realtà, ancora più semplicemente, è già possibile ottenere delle statistiche sulle differenze tra i due grafi, operando nel seguente modo:

```
[max@gabbiano tesi]$ ~/tesi/src/tdiffGraph final1.graph final2.graph
tdiffGraph 1.0
Reading graph file 'final1.graph'...
Reading graph file 'final2.graph'...
Computing differences...
Nodes in 1st graph:       10909
- only in 1st graph:       1289
Nodes in 2nd graph:       12708
- only in 2nd graph:       3088
Common nodes:              9620
Edges in 1st graph:       23817
- peering:                    0
- directed:               23776
- unoriented:                41
- only in 1st graph:       7873
  * peering:                  0
  * directed:              7850
  * unoriented:              23
Edges in 2nd graph:       27555
- peering:                    0
- directed:               27530
- unoriented:                25
- only in 2nd graph:      11611
  * peering:                  0
  * directed:             11593
  * unoriented:              18
Common edges:             15944
- oppositely dir.:          801
- differently dir.:         801
- consistently dir.:      15118
[max@gabbiano tesi]$
```

Infine, il grafo differenziale costruito può essere analizzato per mostrare, ad esempio, quali sono gli archi su cui si manifesta la differenza:

```
[max@gabbiano tesi]$ ~/tesi/src/tviewGraph
tviewGraph 1.0
Interactive mode; type 'help' to get a list of available commands
tvg> load diff.graph
tvg> show graph information
Graph file: diff.graph
Graph type: (multi)differential
Graph comment:
Graph has been generated using 2 files:
 final1.graph
 final2.graph
tvg> show graph nodes
```

```
471
tvg> show graph edges
801
tvg> dump graph edges
    1 <-- --> 174
    1 --> <-- 209
    1 --> <-- 2828
    1 <-- --> 3209
    1 --> <-- 3578
    1 <-- --> 3914
    1 <-- --> 3967
    1 --> <-- 4006
    1 <-- --> 4323
    1 --> <-- 4358
    1 <-- --> 4544
    1 --> <-- 5696
    1 <-- --> 5727
.....
```

Per quello che riguarda il modulo di estrazione degli AS path, piuttosto che esaminarne l'utilizzo diretto (che, al di là della scelta delle modalità di estrazione, non presenta aspetti di particolare interesse), si preferisce mostrare ed analizzare un esempio di rapporto da esso generato.

```
AS paths      unchanged     changed     deleted       saved      tagged
incomplete       513323           -           0           0           0
w/ prep.              0      707782           0           0           0
w/ private            0         349           0           0           0
w/ AS sets            -        1202           0           0           0
w/ cycles             0           0         206           0           0
TOTAL            513323      709333         206           0           0

                                          empty       empty
                                         before       after
AS paths      processed    reported  processing  processing
input file      6067634     6067428           0           0
incom.file            -           0           -           0
prep. file            -           0           -           0
priv. file            -           0           -           0
sets file             -           0           -           0
cycle file            -           0           -           0
TOTAL                       6067428

File lines         read      parsed   w/ errors
                6068459     6068428           0
```

Il primo gruppo di valori riporta quantità di AS path, classificate in base alle caratteristiche che essi hanno presentato ed all'operazione che su di essi ha avuto luogo.

Il secondo gruppo riporta informazioni sul numero di AS path estratti e salvati negli eventuali file esterni (uno per ogni caratteristica). In questo insieme viene anche riportato il numero di cammini per i quali la rimozione di una determinata caratteristica (per esempio, degli AS privati) ha avuto come esito la generazione di un cammino vuoto; in questo caso, il cammino viene scartato automaticamente.

L'ultimo gruppo mostra il numero di righe lette dal file contenente la tabella BGP di input, quelle ritenute significative e quelle contenenti errori. Se TPATHEXTRACT viene avviato con l'opzione `-v`, ciascun errore viene stampato su terminale nel momento in cui si verifica, insieme con una breve spiegazione della sua possibile causa.

Capitolo 4

Ottimizzazione dell'efficienza dell'algoritmo per l'inferenza delle relazioni tra Autonomous Systems

Un elemento fondamentale per potersi porre il problema di analizzare l'evoluzione delle relazioni tra Autonomous Systems è costituito dalla disponibilità di un'implementazione di un algoritmo di inferenza di tali relazioni. Tuttavia, questo non basta a rendere proficua l'analisi effettuata, poiché eventuali problemi di efficienza possono limitare fortemente il numero di risultati che è possibile produrre. Evidentemente, questo rende poco interessanti anche le elaborazioni che su di essi possono essere effettuate.

In questo capitolo viene dunque preso in considerazione un aspetto poco inerente l'evoluzione in quanto tale, e che risulta, tuttavia, fortemente propedeutico alla possibilità di studiarla. Oltre a mostrare le ragioni che hanno spinto a preferire per le sperimentazioni un algoritmo invece di un altro, vengono qui illustrati i miglioramenti che è stato necessario apportarvi per consentire il completo svolgimento dello studio qui presentato.

4.1 Le ragioni per la necessità di una notevole efficienza

Lo scopo di questa trattazione è quello di analizzare i cambiamenti nelle relazioni presenti tra coppie di AS con due obiettivi fondamentali: quello di convalidare la qualità dell'algoritmo di inferenza utilizzato per ricavarle e, una volta che la sua validità fosse in questo modo confermata, quello di renderlo utilizzabile ancora più efficacemente per la conoscenza dei rapporti commerciali.

L'approccio seguito in questa sede consiste in un utilizzo massiccio di un algoritmo per studiare il comportamento dei risultati da esso prodotti, eventualmente con riferimento ad alcune informazioni già note circa gli accordi commerciali. Perché le conclusioni tratte possano considerarsi valide, è necessario che lo studio abbia luogo su una quantità abbastanza elevata di dati da coprire un intervallo temporale significativo (una o due settimane), in cui possano aver realisticamente luogo dei cambiamenti di rapporto.

Per le prime sperimentazioni sono stati utilizzati i dati disponibili in [9], poiché sono serviti anche come termine di paragone tra la soluzione di [8] (basata sui vantage point) e quella di [20] (basata sulle formule logiche). Questi dati sono tuttavia disponibili quantità relativamente limitata (a tutt'oggi sono disponibili tabelle BGP per 8 date diverse), ed inizialmente, quindi, non è stato strettamente necessario ottimizzare l'algoritmo utilizzato per l'inferenza.

Successivamente, una volta esauriti i dati di [9], è stata utilizzata come sorgente informativa primaria l'archivio del server Oregon Route Views ([22]), poiché in esso sono disponibili tabelle BGP distribuite su un arco di circa 3 anni e prelevate ad intervalli regolari e molto ravvicinati (ogni 2 ore). In ogni caso, anche avendo scelto una sorgente che campionasse i dati di routing con un ritmo meno serrato, le tabelle BGP con cui si avrebbe avuto a che fare durante una settimana sarebbero state almeno 15. Considerando che su ciascuna di esse va effettuata una serie di elaborazioni, di cui la più pesante in termini di calcolo è la stessa inferenza, risulta chiaro che si manifesta un bisogno più pressante di ottenere la maggior quantità di risultati nel minor tempo possibile.

4.2 Scelta di un algoritmo e criteri in essa utilizzati

Per operare l'inferenza delle relazioni si è scelto di utilizzare l'algoritmo basato sulla soddisfacibilità di formule logiche, illustrato in [20] e riassunto nella sezione 2.4. Questa decisione è stata influenzata principalmente da due fattori. Innanzitutto, la disponibilità immediata di un'implementazione, che porta a scartare la soluzione di [8] (vantage points). In secondo luogo, il fatto che l'algoritmo di [20] (soddisfacibilità di formule logiche) è stato sviluppato presso il Dipartimento di Informatica e Automazione dell'Università degli Studi Roma Tre, nell'ambito del quale nasce anche il lavoro qui presentato: questo significa che, nel corso di tutte le fasi dello sviluppo di questa trattazione, è stato possibile operare interscambi molto proficui con le persone che hanno partecipato a tale progetto, il che ha consentito di operare scelte più consapevoli e fondate e di convalidare ogni modifica con gli autori stessi dell'algoritmo.

4.3 Analisi della versione preesistente dell'algoritmo scelto

In questa sezione vengono presentate le caratteristiche principali dell'implementazione dell'algoritmo di [20], che è disponibile presso [23] sotto il nome di BGPPATHSAT (versione 17/07/2002). Tale versione è stata prodotta in un momento in cui si è resa necessaria la disponibilità il più possibile immediata di uno strumento di questo tipo, per poter comprovare la validità del metodo introdotto; per questo motivo, in essa sono state operate alcune scelte mirate ad

`SatVariable = AsAdjacency = pair<unsigned short, unsigned short>`	Tipo utilizzato per rappresentare le variabili $(x_{i,j})$
`Literal = pair<SatVariable, bool>`	Tipo utilizzato per la rappresentazione dei letterali $(x_{i,j}, \overline{x}_{i,j})$
`Graph <VertexType, EdgeType>`	Classe astratta per la rappresentazione dei grafi (contiene alcuni metodi virtuali)
`GrowGraph <VertexType, EdgeType>`	Classe per la rappresentazione dei grafi
`SatGraphType = GrowGraph <Literal, int>`	Tipo utilizzato per la rappresentazione del grafo dei letterali

Tabella 4.1: Principali tipi di dato utilizzati in bgpPathSat

ottenere un compromesso tra efficienza e rapidità di realizzazione. Alcune di esse vengono in questa sede riviste per rendere l'algoritmo più funzionale ad un'applicazione a grosse quantità di dati.

4.3.1 Operazioni eseguite nell'algoritmo

Il funzionamento generale dell'algoritmo si basa su una struttura dati interna, il *grafo dei letterali*, che rappresenta il grafo $G_{2-\mathrm{SAT}}$ introdotto nella sezione 2.4.2 per verificare se sia possibile o meno orientare gli archi del grafo degli AS senza introdurre cammini invalidi. Questo grafo viene rappresentato utilizzando una classe `Graph` scritta appositamente per lo scopo, a sua volta basata sull'utilizzo di mappe e liste messe a disposizione dalla libreria *STL*[1]. Ciascun nodo del grafo è di tipo `Literal`, ovvero una coppia il cui primo elemento è, a sua volta, una coppia di numeri di AS ed il secondo un valore booleano. Questi tipi di dato sono riassunti nella tabella 4.1.

Il grafo viene costruito sulla base dei cammini letti da un file, inserendo una coppia di nodi $(x_{i,j}, \overline{x}_{i,j})$ per ogni coppia di AS (i, j) ed una coppia di archi $(x_{j,k} \rightarrow x_{i,j}, \overline{x}_{i,j} \rightarrow \overline{x}_{j,k})$[2] per ogni tripla di AS (i, j, k).

Per calcolare le componenti fortemente connesse di questo grafo si ricorre ad una procedura disponibile nella libreria *LEDA*[3]. Questa scelta è motivata dall'estrema efficienza di tale funzione, difficilmente eguagliabile con strumenti scritti appositamente per lo scopo se non impiegando un'eccessiva quantità di

[1] Standard Template Library, una libreria che mette a disposizione molti tipi di dato sotto forma di *template*, per renderne l'utilizzo estremamente generale.

[2] In realtà, poiché vale la proprietà generale $x_{i,j} = \overline{x}_{j,i}$, l'inserimento dei nodi e degli archi procede verificando sempre preliminarmente la possibilità di riutilizzare nodi esistenti. Così, $x_{i,j}$ non viene inserito se esiste già $\overline{x}_{j,i}$, e gli archi inseriti in corrispondenza della tripla (i, j, k) sono del tipo $l_1 \rightarrow l_2$ e $\overline{l}_2 \rightarrow \overline{l}_1$, con

$$l_1 = \left\{ \begin{array}{l} x_{j,k} \text{ oppure} \\ \overline{x}_{k,j} \end{array} \right. \quad \text{e } l_2 = \left\{ \begin{array}{l} x_{i,j} \text{ oppure} \\ \overline{x}_{j,i} \end{array} \right.$$

[3] Library of Efficient Data types and Algorithms, prodotta dalla Algorithmic Solutions Software GmbH.

INPUT: una lista di AS path
OUTPUT: un'orientazione per il grafo degli AS (se esiste)

$V_{G_{lit}} = E_{G_{lit}} = \varnothing$
per ogni cammino $p = (v_1, v_2, \ldots, v_n)$ in ingresso
    per ogni coppia di AS consecutivi (v_i, v_j)
        se $x_{v_i,v_j} \notin V_{G_{lit}}$ e $x_{v_j,v_i} \notin V_{G_{lit}}$
            $V_{G_{lit}} = V_{G_{lit}} \cup \left\{x_{v_i,v_j}, \overline{x}_{v_i,v_j}\right\}$
per ogni cammino $p = (v_1, v_2, \ldots, v_n)$ in ingresso
    per ogni tripla di AS consecutivi (v_i, v_j, v_k)
        se $x_{v_j,v_k} \in V_{G_{lit}}$
            $l_1 = x_{v_j,v_k}$
        altrimenti
            $l_1 = \overline{x}_{v_k,v_j}$
        se $x_{v_i,v_j} \in V_{G_{lit}}$
            $l_2 = x_{v_i,v_j}$
        altrimenti
            $l_2 = \overline{x}_{v_j,v_i}$
        $E_{G_{lit}} = E_{G_{lit}} \cup \left\{l_1 \to l_2, \overline{l}_2 \to \overline{l}_1\right\}$
copia di G_{lit} in $LEDAG_{lit}$
calcolo (con LEDA) delle componenti fortemente connesse di $LEDAG_{lit}$
per ogni variabile $x_{v_i,v_j} \in V_{G_{lit}}$
    se $scc\left[x_{v_i,v_j}\right] = scc\left[\overline{x}_{v_i,v_j}\right]$
        $contraddizioni = contraddizioni \cup \{(i,j)\}$
se $|contraddizioni| = 0$
    copia di G_{lit} in $LEDAG_{lit}$
    calcolo (con LEDA) di un ordinamento topologico dei nodi di $LEDAG_{lit}$
    per ogni variabile $x_{v_i,v_j} \in V_{G_{lit}}$
        se $sort\left[x_{v_i,v_j}\right] > sort\left[\overline{x}_{v_i,v_j}\right]$
            $x_{v_i,v_j} = vero$
        altrimenti
            $x_{v_i,v_j} = falso$

Figura 4.1: Algoritmo utilizzato in BGPPATHSAT, versione base

tempo sul problema (che, evidentemente, in questo contesto non è di primario interesse).

Il funzionamento normale dell'algoritmo prevede l'esecuzione dei passi riportati nella figura 4.1. In essa si è utilizzato il simbolo G_{lit} per indicare il grafo dei letterali (corrispondente a $G_{2-\mathrm{SAT}}$), e $V_{G_{lit}}$ e $E_{G_{lit}}$ per indicare, rispettivamente, i suoi nodi ed i suoi archi. Il simbolo $LEDAG_{lit}$ indica invece il grafo dei letterali rappresentato utilizzando le strutture dati messe a disposizione dalla libreria LEDA, e $V_{LEDAG_{lit}}$ e $E_{LEDAG_{lit}}$ rappresentano i suoi nodi ed i suoi archi. Il vettore $scc[]$ rappresenta un indice associato a ciascun nodo di un grafo, corrispondente al numero della componente fortemente connessa cui esso appartiene. Analogamente, il vettore $sort[]$ costituisce l'indice attribuito ai nodi durante l'ordinamento topologico.

Tutte le operazioni descritte vengono eseguite una sola volta per ognuno dei file in ingresso. Si noti, tuttavia, come risulti già piuttosto pesante dover effettuare una copia del grafo dei letterali in un grafo di LEDA ogni volta che ci sia bisogno di utilizzare una funzione esterna (calcolo delle componenti fortemente connesse, ordinamento topologico).

Per quanto riguarda le due fasi fondamentali dell'euristica descritta nella sezione 2.4.3, esse sono rappresentate nelle figure 4.2 e 4.3. La prima descrive la fase di scarto dei cammini che rendono l'insieme non soddisfacibile, e la seconda l'operazione di reinserimento di questi ultimi. La notazione utilizzata è la stessa della figura 4.1. In più, viene utilizzato il simbolo @, con il quale si indica l'operatore di concatenazione tra due liste: (i_1, i_2) @ $(i_3, i_4, i_5) = (i_1, i_2, i_3, i_4, i_5)$. Gli insiemi *goodPathsSet* e *badPathsSet* rappresentano, rispettivamente, l'insieme dei cammini che rendono il problema soddisfacibile e quelli scartati perché causa di contraddizioni nel grafo G_{lit}. La lista *goodPathsList* viene costruita perché necessaria come input per l'algoritmo in figura 4.1. *sccCount* è il numero di componenti fortemente connesse individuate da LEDA. $sccNodes[]$ è un vettore che contiene i nodi associati ad ogni componente fortemente connessa. *metric* è un vettore che contiene il *path covering* associato ad ogni coppia di AS (e, quindi, ad ogni variabile). La copia al passo 20 dell'algoritmo in figura 4.3 viene chiamata "veloce" perché effettuata mediante l'utilizzo di puntatori: al contrario delle altre copie, che percorrono l'intera struttura dati in esame per ricostruirla altrove, questo tipo di copia è, dunque, pressoché istantanea. La notazione $goodPathsSet[i]$ indica l'i-esimo elemento dell'insieme *goodPathsSet*, assumendo di averne ordinato strettamente il contenuto. Il valore *chunk* viene utilizzato per tentare di reinserire i cammini più velocemente: se ad una iterazione il reinserimento ha successo, a quella successiva si prova a considerare un numero sempre maggiore di cammini, fino a che, in caso di fallimento, si riparte da 1. Si noti come la fase di reinserimento riconsideri i cammini di lunghezza 2 senza curarsi della soddisfacibilità: questo è possibile perché, riconsiderandoli, ci si limita ad inserire (eventualmente) nuovi nodi nel grafo dei letterali, il che non può generare dei cicli (e, quindi, delle nuove contraddizioni).

Entrambe le fasi possono considerarsi completate dall'assegnazione di valori di verità alle variabili contenute nel grafo G_{lit} costruito sulla base dei soli cammini contenuti in *goodPathsSet*. Per fare questo, si possono eseguire i passi 1-16 e 23-29 dell'algoritmo in figura 4.1.

4.3.2 Individuazione di operazioni critiche dal punto di vista dell'efficienza

La ricerca di parti migliorabili ha avuto come oggetto esclusivamente gli algoritmi rappresentati nelle figure 4.2 e 4.3: infatti, al contrario della versione priva di euristiche, essi presentano dei passi che vengono ripetuti rispettivamente centinaia (per l'euristica) e decine di migliaia (per il reinserimento dei cammini) di volte, il che rende cruciale lo sfruttamento ottimale del tempo ad ogni passo.

Accanto alle operazioni ritenute più pesanti in termini di calcolo sono stati riportati dei tempi di esecuzione (naturalmente, molto approssimativi), che vengono riassunti nella tabella 4.2. Nella tabella sono state omesse le operazioni (per esempio, la costruzione del grafo G_{lit} nella fase di reinserimento) che non vengono eseguite all'interno di iterazioni, poiché il loro impatto è decisamente irrilevante. Le piccole oscillazioni nei tempi riportati sono dovute a piccole variazioni nelle attività del calcolatore durante la valutazione ed al fatto che

INPUT: una lista di AS path
OUTPUT: un insieme di AS path che rendono la lista di partenza insoddisfacibile

1. $sat = falso$
2. $goodPathsSet = \varnothing$
3. $badPathsSet = \varnothing$
4. per ogni cammino $p = (v_1, v_2, \ldots, v_n)$ in ingresso
5. $\quad\quad goodPathsSet = goodPathsSet \cup \{p\}$
6. finché $sat = falso$
7. $\quad goodPathsList = ()$
8. $\quad$ per ogni cammino $p \in goodPathsSet$
9. $\quad\quad goodPathsList = goodPathsList \ @ \ (p)$
10. $\quad$ costruzione del grafo G_{lit} (passi 1-16 dell'algoritmo in figura 4.1) — 20.0 s
11. $\quad$ copia di G_{lit} in $LEDAG_{lit}$ — 2.6 s
12. $\quad$ calcolo (con LEDA) delle componenti fortemente connesse di $LEDAG_{lit}$ — 0.2 s
13. $\quad$ se $sccCount = |V_{G_{lit}}|$
14. $\quad\quad sat = vero$
15. $\quad\quad$ salta all'iterazione successiva
16. $\quad$ per ogni $x_{v_i,v_j} \in V_{G_{lit}}$
17. $\quad\quad sccNodes\left[scc\left[x_{v_i,v_j}\right]\right] = sccNodes\left[scc\left[x_{v_i,v_j}\right]\right] \cup \left\{x_{v_i,v_j}\right\}$
18. $\quad minMetric = \min_{x_{v_i,v_j} \in sccNodes[i],\ i \ | \ |sccNodes[i]|>1} \left(metric\left[x_{v_i,v_j}\right] + metric\left[\overline{x}_{v_i,v_j}\right]\right)$
19. $\quad minAdj = (v_i, v_j) \ | \ metric\left[x_{v_i,v_j}\right] + metric\left[\overline{x}_{v_i,v_j}\right] = minMetric$
20. $\quad$ per ogni $p \in goodPathsSet$
21. $\quad\quad$ se $\exists p_1, p_2 \ | \ p = p_1 \ @ \ minAdj \ @ \ p_2$
22. $\quad\quad\quad goodPathsSet = goodPathsSet - \{p\}$
23. $\quad\quad\quad badPathsSet = badPathsSet \cup \{p\}$

Figura 4.2: Euristica utilizzata in bgpPathSat per lo scarto dei cammini che rendono l'insieme degli AS path insoddisfacibile

INPUT: un insieme soddisfacibile di AS path; un insieme di AS path scartati dall'euristica in figura 4.2
OUTPUT: un insieme soddisfacibile e massimale di AS path

$veryBad = \varnothing$
per ogni cammino $p \in goodPathsSet$
    $goodPathsList = goodPathsList \ @ \ (p)$
costruzione del grafo G_{lit} (passi 1-16 dell'algoritmo in figura 4.1) — 18.0 s
per ogni cammino $p = (v_i, v_j) \in goodPathsSet$
    $goodPathsSet = goodPathsSet \cup \{p\}$
    $badPathsSet = badPathsSet - \{p\}$
$chunk = 16$
finché $badPathsSet \neq \varnothing$
    $reinsertThis = \varnothing$
    per ogni i in $\{1, \ldots, chunk\}$
        se $i < |goodPathsSet|$
            $reinsertThis = reinsertThis \cup goodPathsSet[i]$
    copia G_{lit} in G'_{lit} — 1.0 s
    per ogni cammino $p \in reinsertThis$
        aggiornamento di G'_{lit} con i nuovi archi (passi 7-16 dell'algoritmo in figura 4.1)
    copia di G'_{lit} in $LEDAG_{lit}$ — 2.5 s
    calcolo (con LEDA) delle componenti fortemente connesse di $LEDAG_{lit}$ — 0.2 s
    se $sccCount = |V_{G_{lit}}|$
        copia veloce di G'_{lit} in G_{lit}
        $chunk = chunk * 2$
    altrimenti
        se $chunk = 1$
            $veryBad = veryBad \cup reinsertThis$
    cancellazione di G'_{lit} — 0.7 s
    $chunk = 1$

Figura 4.3: Algoritmo utilizzato in BGPPATHSAT per il reinserimento dei cammini scartati dall'euristica ogni iterazione manifesta valori leggermente diversi, dei quali qui è stata riportata una semplice stima. I totali riportati per ciascuna iterazione includono, chiaramente, anche tutte le altre operazioni di aggiornamento e stampa che la costituiscono, e, dunque, risultano leggermente superiori al valore ottenuto dalla somma delle voci nella tabella.

Scarto dei *bad paths*	
Costruzione G_{lit}	20 s
Copia $G_{lit} \rightarrow LEDAG_{lit}$	2.6 s
Calcolo componenti fortemente connesse	0.2 s
TOTALE ITERAZIONE	25 s
TOTALE SU UNA COMPUTAZIONE COMPLETA	2-4 ore
Reinserimento dei *bad paths*	
Copia $G_{lit} \rightarrow G'_{lit}$	1 s
Copia $G'_{lit} \rightarrow LEDAG_{lit}$	2.5 s
Calcolo compomenti fortemente connesse	0.2 s
Cancellazione di G'_{lit}	0.7 s
TOTALE ITERAZIONE	4.6 s
TOTALE SU UNA COMPUTAZIONE COMPLETA	13-20 ore
TEMPO TOTALE PER UNA COMPUTAZIONE	17-24 ore

Tabella 4.2: Tempi impiegati per l'esecuzione di operazioni critiche (valutati su un Pentium III 866 MHz con 640 Mb di RAM a 133 MHz)

Dalla valutazione dei tempi di calcolo e da un esame delle operazioni eseguite, è possibile osservare come la fase di reinserimento risulti enormemente più pesante di quella di estrazione degli AS path, e, come tale, è stata considerata per prima nel processo di ottimizzazione. Inoltre, alcune operazioni (come la copia di grafi in strutture di LEDA) risultano effettuate anche se non necessario, con un impatto non indifferente su ciascuna iterazione e decisamente enorme sul tempo complessivo di calcolo. In particolare, i passi ritenuti maggiormente critici sono proprio quelli riportati nella tabella 4.2; gli altri sono indispensabili (stampa di informazioni) o di peso decisamente ininfluente (aggiornamento di mappe ed altre strutture dati).

Durante questo processo di *profiling* si è anche rivelato un problema di carattere concettuale. Infatti, ogni volta che viene effettuata la verifica di soddisfacibilità (passo 13 dell'algoritmo in figura 4.2 e passo 19 di quello in figura 4.3), anziché controllare l'esistenza di variabili che inducono contraddizioni, ci si limita a verificare che non esistano componenti fortemente connesse che contengano più di un nodo. Questa è, evidentemente, una condizione necessaria, ma non sufficiente, affinché l'istanza sia soddisfacibile. In altre parole, questo tipo di test è "pessimista", nel senso che non segnala come soddisfacibili istanze che non lo sono (*falsi positivi*), ma può riportare come insoddisfacibili situazioni che avrebbero dovuto causare l'arresto dell'algoritmo (*falsi negativi*), comportando dunque l'esecuzione di più iterazioni del necessario.

4.4 Ottimizzazione

In fase di ottimizzazione sono state eliminate tutte le operazioni ritenute superflue per il funzionamento dell'algoritmo, correggendo opportunamente i passi circostanti per renderli compatibili con la nuova procedura.

È stato inoltre corretto il test di soddisfacibilità, e questo aspetto viene ugualmente considerato come facente parte dell'ottimizzazione, in quanto il vecchio test provocava l'esecuzione di iterazioni inutili.

4.4.1 Interventi effettuati

La ristrutturazione è stata mirata ad eliminare le numerose copie di grafi, utilizzando invece direttamente le strutture di LEDA durante tutto il processo ed eventualmente effettuando un'unica copia al suo termine. Questo ha portato ad alleggerire notevolmente il peso della fase di reinserimento, a tal punto che la fase più lunga (prima di essere essa stessa ottimizzata) è poi diventata quella di scarto dei bad path. Per quest'ultima è stato effettuato lo stesso tipo di ottimizzazione, ma in più è stata eliminata la ricostruzione del grafo G_{lit} ad ogni iterazione, preferendo effettuarne di volta in volta dei rapidi aggiornamenti.

Per quanto riguarda il test di soddisfacibilità, esso è stato corretto sostituendolo con una versione corrispondente alla definizione stessa di soddisfacibilità. Questo ha comportato un leggero decadimento delle prestazioni rispetto alla versione precedente (qualsiasi cosa è più lenta di un semplice test di uguaglianza), tuttavia assolutamente inapprezzabile rispetto al guadagno complessivo che si è avuto in seguito agli altri interventi.

Infine, proprio l'introduzione di un test di soddisfacibilità corretto ha comportato l'insorgere di situazioni in cui si renda necessario calcolare un ordinamento topologico su un grafo con qualche componente fortemente connessa costituita da più di un nodo. Di conseguenza, si è manifestata la necessità di correggere anche la funzione di calcolo di tale ordinamento, poiché in precedenza non prevedeva la contrazione preliminare delle componenti fortemente connesse in un unico nodo. Quest'ultima assunzione era corretta nella versione precedente, in quanto l'orientazione (e quindi l'ordinamento topologico) non veniva mai calcolata prima che fossero scomparse tutte le componenti fortemente connesse di dimensione superiore ad 1, ma produceva inevitabilmente risultati insensati in seguito alle modifiche apportate.

L'algoritmo in figura 4.5 sfrutta la metrica (già calcolata in fase di lettura degli AS path) ed un indice chiamato $usage[]$ per capire quando un nodo o un arco non appaiono più in nessuna implicazione e possono quindi essere effettivamente eliminati.

Si noti inoltre come il nuovo test di soddisfacibilità consenta la localizzazione di almeno una componente fortemente connessa in cui si verifica una contraddizione: nella versione precedente dell'euristica la ricerca di una variabile da eliminare procedeva esplorando tutte le componenti di dimensione superiore ad 1 (passo 18 in figura 4.2), rischiando di cancellare cammini che non avrebbero variato nulla ai fini della soddisfacibilità; in questo caso è invece possibile mirare direttamente ad una componente in cui si sia davvero manifestata una contraddizione (passo 16 in figura 4.4).

Per quanto riguarda l'euristica di scarto dei cammini, il guadagno più considerevole deriva dall'utilizzo esclusivo di grafi di LEDA e dall'aver sostitui-

INPUT: una lista di AS path
OUTPUT: un insieme di AS path che rendono la lista di partenza insoddisfacibile

1. $sat = falso$
2. $goodPathsSet = \varnothing$
3. $badPathsSet = \varnothing$
4. per ogni cammino $p = (v_1, v_2, \ldots, v_n)$ in ingresso
5. $\quad goodPathsSet = goodPathsSet \cup \{p\}$
6. $goodPathsList = ()$
7. per ogni cammino $p \in goodPathsSet$
8. $\quad goodPathsList = goodPathsList \; @ \; (p)$
9. costruzione del grafo $LEDAG_{lit}$ (passi 1-16 dell'algoritmo in figura 4.1, effettuati utilizzando le strutture di LEDA)
10. finché $sat = falso$
11. $\quad sat = satTest\,(LEDAG_{lit}, criticalscc)$
12. $\quad$ se $sat = vero$
13. $\quad\quad$ salta all'iterazione successiva
14. $\quad$ per ogni $x_{v_i,v_j} \in V_{LEDAG_{lit}}$
15. $\quad\quad sccNodes\left[scc\left[x_{v_i,v_j}\right]\right] = sccNodes\left[scc\left[x_{v_i,v_j}\right]\right] \cup \left\{x_{v_i,v_j}\right\}$
16. $\quad minMetric = \min_{x_{v_i,v_j} \in sccNodes[criticalscc]} \left(metric\left[x_{v_i,v_j}\right] + metric\left[\overline{x}_{v_i,v_j}\right]\right)$
17. $\quad minAdj = (v_i, v_j) \mid metric\left[x_{v_i,v_j}\right] + metric\left[\overline{x}_{v_i,v_j}\right] = minMetric$
18. $\quad$ per ogni $p = (v_1, v_2, \ldots, v_n) \in goodPathsSet$
19. $\quad\quad$ se $\exists p_1, p_2 \mid p = p_1 \; @ \; minAdj \; @ \; p_2$
20. $\quad\quad\quad goodPathsSet = goodPathsSet - \{p\}$
21. $\quad\quad\quad badPathsSet = badPathsSet \cup \{p\}$
22. $\quad\quad\quad$ esecuzione dell'algoritmo in figura 4.5 per l'aggiornamento di $LEDAG_{lit}$

Figura 4.4: Versione ottimizzata dell'algoritmo per lo scarto dei cammini che rendono l'insieme degli AS path insoddisfacibile

INPUT: un AS path scartato dall'euristica; un grafo di LEDA che rappresenti il grafo dei letterali corrente
OUTPUT: un grafo di LEDA in cui sono stati cancellati nodi ed archi non più utilizzati da nessuna implicazione logica

$p = (v_1, v_2, \ldots, v_n)$
se $n = 2$
$\quad$ se $x_{v_1,v_2} \in V_{LEDAG_{lit}}$
$\quad\quad$ $l = x_{v_1,v_2}$
$\quad$ altrimenti
$\quad\quad$ $l = x_{v_2,v_1}$
$\quad$ $metric\,[l] = metric\,[l] - 1$
$\quad$ se $metric\,[l] + metric\,[\overline{l}] = 0$
$\quad\quad$ $V_{LEDAG_{lit}} = V_{LEDAG_{lit}} - \{l\}$
se $n \geq 3$
$\quad$ per ogni $i \in \{1, \ldots, n-2\}$
$\quad\quad$ se $x_{v_{i+1},v_{i+2}} \in V_{LEDAG_{lit}}$
$\quad\quad\quad$ $l_1 = x_{v_{i+1},v_{i+2}}$
$\quad\quad$ altrimenti
$\quad\quad\quad$ $l_1 = x_{v_{i+2},v_{i+1}}$
$\quad\quad$ se $x_{v_i,v_{i+1}} \in V_{LEDAG_{lit}}$
$\quad\quad\quad$ $l_2 = x_{v_i,v_{i+1}}$
$\quad\quad$ altrimenti
$\quad\quad\quad$ $l_2 = x_{v_{i+1},v_i}$
$\quad\quad$ $usage\,[l_1 \to l_2] = usage\,[l_1 \to l_2] - 1$
$\quad\quad$ se $usage\,[l_1 \to l_2] = 0$
$\quad\quad\quad$ $E_{LEDAG_{lit}} = E_{LEDAG_{lit}} - \{l_1 \to l_2\}$
$\quad\quad$ $usage\,[\overline{l}_2 \to \overline{l}_1] = usage\,[\overline{l}_2 \to \overline{l}_1] - 1$
$\quad\quad$ se $usage\,[\overline{l}_2 \to \overline{l}_1] = 0$
$\quad\quad\quad$ $E_{LEDAG_{lit}} = E_{LEDAG_{lit}} - \{\overline{l}_2 \to \overline{l}_1\}$
$\quad\quad$ $metric\,[l_2] = metric\,[l_2] - 1$
$\quad\quad$ se $metric\,[l_2] + metric\,[\overline{l}_2] = 0$
$\quad\quad\quad$ $V_{LEDAG_{lit}} = V_{LEDAG_{lit}} - \{l_2, \overline{l}_2\}$
$\quad$ se $x_{v_{n-1},v_n} \in V_{LEDAG_{lit}}$
$\quad\quad$ $l = x_{v_{n-1},v_n}$
$\quad$ altrimenti
$\quad\quad$ $l = x_{v_n,v_{n-1}}$
$\quad$ $metric\,[l] = metric\,[l] - 1$
$\quad$ se $metric\,[l] + metric\,[\overline{l}] = 0$
$\quad\quad$ $V_{LEDAG_{lit}} = V_{LEDAG_{lit}} - \{l, \overline{l}\}$

Figura 4.5: Algoritmo per l'aggiornamento del grafo dei letterali in seguito alla cancellazione di un AS path

INPUT: un grafo dei letterali
OUTPUT: un valore di verità che indica se il grafo rappresenta un'istanza soddisfacibile; un intero che contenga l'indice di una componente fortemente connessa in cui si sia manifestata un'eventuale contraddizione

calcolo (con LEDA) delle componenti fortemente connesse di $LEDAG_{lit}$
per ogni $x_{v_i,v_j} \in V_{LEDAG_{lit}}$
    $sccNodes\left[scc\left[x_{v_i,v_j}\right]\right] = sccNodes\left[scc\left[x_{v_i,v_j}\right]\right] \cup \left\{x_{v_i,v_j}\right\}$
$criticalscc = -1$
$sat = vero$
per ogni $i \in \{1, \ldots, sccCount\}$
    $litset = \varnothing$
    per ogni $x_{v_i,v_j} \in sccNodes[i]$
        $litset = litset \cup \left\{x_{v_i,v_j}\right\}$
        se $\overline{x}_{v_i,v_j} \in litset$
            $criticalscc = i$
            $sat = falso$

Figura 4.6: Algoritmo $satTest$ per controllare la soddisfacibilità di un grafo di letterali indotto da un insieme di AS path

to la ricostruzione di questi ultimi ad ogni iterazione con un loro semplice aggiornamento.

La fase di reinserimento trae anch'essa grosso giovamento dall'utilizzo dei grafi di LEDA e, soprattutto, dall'eliminazione delle copie utilizzate per ripristinare la situazione precedente qualora il reinserimento non avesse successo. Ad ogni iterazione viene mantenuto aggiornato il grafo $LEDAG_{lit}$, e, solo quando necessario, si applicano le modifiche ad esso effettuate anche a G_{lit}. In realtà, quest'ultima operazione avrebbe potuto essere eliminata, ma si rende, di fatto, necessaria perché l'assegnazione di un'orientazione procede poi sfruttando proprio G_{lit}; anziché modificare anche tale procedura, è stato ritenuto più opportuno mantenerla invariata ed adattare opportunamente l'algoritmo di reinserimento. Il valore di *chunk* viene ora aggiornato in modo diverso: questa variazione, in realtà, è frutto di una fase intermedia dell'ottimizzazione, durante la quale sono state ottenute prestazioni migliori (i cammini sono stati reinseriti utilizzando un minor numero di iterazioni) in presenza di alcuni dati di input. Di fatto, questa modifica diventa pressoché irrilevante nelle condizioni di utilizzo generali.

4.4.2 Guadagno in termini di efficienza

Prima di ritenere definitivi i cambiamenti apportati, la nuova versione del software è stata testata confrontandone i risultati prodotti su istanze per le quali fosse già stata calcolata un'orientazione con la vecchia versione, e verificando che non vi fossero differenze. In realtà, mentre il reinserimento produce risultati del tutto identici, la fase di scarto dei bad path è influenzata dall'ordine con cui sono considerati i letterali all'interno del vettore $metric[]$: mentre in precedenza era possibile che, cancellando una variabile $x_{i,j}$, si provocasse l'introduzione di

INPUT: un insieme soddisfacibile di AS path; un insieme di AS path scartati dall'euristica in figura 4.2
OUTPUT: un insieme soddisfacibile e massimale di AS path

1. $veryBad = \varnothing$
2. $V'_{LEDAG_{lit}} = \varnothing$
3. $E'_{LEDAG_{lit}} = \varnothing$
4. per ogni cammino $p \in goodPathsSet$
5. $\quad goodPathsList = goodPathsList \ @ \ (p)$
6. costruzione del grafo G_{lit} (passi 1-16 dell'algoritmo in figura 4.1)
7. per ogni cammino $p = (v_i, v_j) \in goodPathsSet$
8. $\quad goodPathsSet = goodPathsSet \cup \{p\}$
9. $\quad badPathsSet = badPathsSet - \{p\}$
10. $\quad$ aggiornamento di G_{lit} secondo i passi 2-16 dell'algoritmo in figura 4.1
11. copia di G_{lit} in $LEDAG_{lit}$
12. $chunk = 16$
13. finché $badPathsSet \neq \varnothing$
14. $\quad reinsertThis = \varnothing$
15. $\quad$ per ogni i in $\{1, \ldots, chunk\}$
16. $\quad\quad$ se $i < |goodPathsSet|$
17. $\quad\quad\quad reinsertThis = reinsertThis \cup goodPathsSet[i]$
18. $\quad$ per ogni cammino $p \in reinsertThis$
19. $\quad\quad$ aggiornamento di $LEDAG_{lit}$ con i nuovi nodi ed archi (passi 2-16 dell'algoritmo in figura 4.1 eseguiti con le strutture di LEDA); inserimento dei nodi ed archi aggiunti in $V'_{LEDAG_{lit}}$ e $E'_{LEDAG_{lit}}$
20. $\quad sat = satTest\,(LEDAG_{lit}, NULL)$
21. $\quad$ se $sat = vero$
22. $\quad\quad V'_{LEDAG_{lit}} = \varnothing$
23. $\quad\quad E'_{LEDAG_{lit}} = \varnothing$
24. $\quad\quad$ aggiornamento di G_{lit} con i nuovi nodi ed archi (passi 2-16 dell'algoritmo in figura 4.1)
25. $\quad\quad$ se $chunk < 8192$
26. $\quad\quad\quad chunk = chunk * 4$
27. $\quad\quad$ altrimenti se $chunk < 16384$
28. $\quad\quad\quad chunk = chunk * 2$
29. $\quad$ altrimenti
30. $\quad\quad E_{LEDAG_{lit}} = E_{LEDAG_{lit}} - E'_{LEDAG_{lit}}$
31. $\quad\quad V_{LEDAG_{lit}} = V_{LEDAG_{lit}} - V'_{LEDAG_{lit}}$
32. $\quad\quad$ se $chunk = 1$
33. $\quad\quad\quad veryBad = veryBad \cup reinsertThis$
34. $\quad\quad$ se $chunk/4 \geq 1$
35. $\quad\quad\quad chunk = chunk/4$
36. $\quad\quad$ altrimenti
37. $\quad\quad\quad chunk = 1$

Figura 4.7: Versione ottimizzata dell'algoritmo per il reinserimento dei cammini scartati dall'euristica

$x_{j,i}$[4], ora questo non è più possibile, perché il grafo dei letterali viene soltanto aggiornato ad ogni iterazione, e mai ricostruito daccapo. Questo comporta che, se una variabile $x_{i,j}$ è stata introdotta come tale, non verrà più eliminata fino a che non esista più neanche la sua complementare ($x_{j,i}$). Di conseguenza, non potendosi alterare dinamicamente l'ordine delle variabili, ci sono situazioni in cui, a parità di metrica (valore $minMetric$ in figura 4.4), viene scelta una variabile diversa da quella che sarebbe stata scelta nella vecchia versione. Le differenze di orientazione innescate da questo fenomeno sono comunque più che trascurabili.

I risultati dell'ottimizzazione sono mostrati nella tabella 4.3. Nonostante siano stati valutati su insiemi di dati diversi e su un calcolatore diverso, risulta evidente come il guadagno in termini di tempo per una computazione completa oscilli tra le 15 e le 23 ore, ovvero si impiega tra il 4% e il 12% del tempo necessario nella versione precedente di bgpPathSat. La valutazione in un contesto diverso non può che influenzare questi valori solo per piccoli scarti: il risultato ottenuto rimane, comunque, in proporzione, decisamente ottimo.

Scarto dei *bad paths*	
Test di soddisfacibilità	0.3 s
TOTALE ITERAZIONE	0.4 s
TOTALE SU UNA COMPUTAZIONE COMPLETA	2-3 minuti
Reinserimento dei *bad paths*	
Test di soddisfacibilità	0.3 s
TOTALE ITERAZIONE	0.3 s
TOTALE SU UNA COMPUTAZIONE COMPLETA	25-70 minuti
TEMPO TOTALE PER UNA COMPUTAZIONE	1-2 ore

Tabella 4.3: Tempi impiegati nella versione ottimizzata di bgpPathSat (valutati su un AMD 1800 MHz con 1 Gb di RAM)

L'ottimizzazione è stata effettuata anche su un altro fronte: correggendo il test di soddisfacibilità, infatti, è stato possibile operare sui cammini in modo tale da evitare di scartarne alcuni e da reinserirne un numero superiore rispetto a quanto fatto in precedenza. Questo aspetto riguarda, più che il peso computazionale, la correttezza concettuale delle operazioni effettuate dall'algoritmo. Tuttavia, accanto alla lieve penalizzazione prestazionale apportata dall'introduzione del test corretto, è stato proprio grazie a quest'ultimo che il numero di iterazioni compiute complessivamente ha subito una riduzione, il che giustifica il fatto di considerare questo tipo di intervento come mirato anch'esso all'ottimizzazione. La tabella 4.4 mostra un confronto tra il funzionamento del software con il nuovo test di soddisfacibilità e quello con la vecchia versione. I dati della comparazione sono stati prelevati dall'archivio di Oregon Route Views ([22]).

[4] Questo era possibile perché il grafo era ricostruito completamente ad ogni iterazione; visto che le variabili introdotte dipendono dall'ordine con cui gli AS compaiono nella prima istanza di ogni adiacenza (per esempio, incontrando prima l'AS path `...701 7018...` si introduce $x_{701,7018}$, mentre se si presenta per primo l'AS path `...7018 701...` si introduce $x_{7018,701}$), la cancellazione di un cammino può far "emergere" come primo cammino quello in cui compare un'adiacenza già esistente, ma con verso rovesciato.

	Vecchio test	Nuovo test	Differenza
Snapshot 25/03/2003 00:00			
Iterazioni scarto *bad paths*	273	268	5
Cammini scartati	20288	16414	3874
Iterazioni reinserimento *bad paths*	15395	12398	2997
Cammini reinseriti	16775 (83%)	13874 (85%)	2901
Snapshot 25/03/2003 02:00			
Iterazioni scarto *bad paths*	283	280	3
Cammini scartati	19507	16912	2595
Iterazioni reinserimento *bad paths*	14719	12330	2389
Cammini reinseriti	16102 (83%)	14370 (85%)	1732
Snapshot 25/03/2003 04:00			
Iterazioni scarto *bad paths*	276	272	4
Cammini scartati	19505	16343	3162
Iterazioni reinserimento *bad paths*	14440	11938	2502
Cammini reinseriti	16371 (84%)	13985 (86%)	2386

Tabella 4.4: Comparazione delle prestazioni con le due versioni del test di soddisfacibilità

Si ribadisce ulteriormente che le prestazioni della versione preesistente dell'algoritmo sono in gran parte giustificate dalla necessità di implementarne una versione funzionante nel più breve tempo possibile, per poter così ottenere risultati pratici dall'applicazione del metodo illustrato in [20].

4.5 La possibilità di vincolare alcune relazioni

Durante l'analisi della versione esistente dell'algoritmo e la sua ottimizzazione è nata anche l'idea di estendere l'implementazione esistente con una funzione alla quale nella sezione 2.4 è stato fornito soltanto un breve cenno teorico. La caratteristica in questione consiste nella possibilità di alterare il grafo dei letterali G_{lit} ($G_{2\text{–SAT}}$), inserendovi degli archi che forzano l'orientazione finale del grafo degli AS a sottostare a delle condizioni (*vincoli*) imposte dall'esterno.

Imporre l'orientazione $i \to j$ di un arco del grafo degli AS corrisponde ad inserire nel grafo dei letterali G_{lit} un arco $\overline{x}_{v_i,v_j} \to x_{v_i,v_j}$, il che a sua volta significa inserire la clausola fittizia $\left(x_{v_i,v_j} \vee x_{v_i,v_j}\right)$ nell'istanza di problema 2–SAT; tutto ciò ha lo scopo di forzare la variabile x_{v_i,v_j} ad assumere il valore *vero*.

Questa operazione è stata implementata nella versione ottimizzata di bgpPathSat allo scopo di consentire, in futuro, un'analisi relativa ai gradi di libertà presentati da ciascuna soluzione calcolata. L'attuale versione, mediante l'uso dell'opzione `--constraint`, può leggere da un file esterno una lista di coppie di AS, corrispondenti alle orientazioni che devono essere mantenute fisse durante l'elaborazione.

Per implementare questa caratteristica sono state scritte delle apposite funzioni di aggiornamento del grafo G_{lit}, che vengono invocate in modo tale da cancellare gli archi di vincolo quando non abbiano più significato (durante l'estrazione dei bad path), per poi reintrodurli qualora i corrispondenti nodi vengano nuovamente inseriti nel grafo stesso (durante il reinserimento dei cammini).

Capitolo 5

Studio dell'evoluzione su situazioni significative: risultati sperimentali

In questo capitolo vengono presentati i test effettuati sul comportamento dell'algoritmo di inferenza presentato nella sezione 2.4 ed ottimizzato come descritto nel capitolo 4. Per effettuare i test sono stati prelevati dati da due sorgenti diverse e su vari intervalli temporali, allo scopo di ottenere una panoramica opportunamente ampia delle reazioni a vari tipi di situazione.

In particolare, i test sono suddivisi in due categorie principali:

- quelli mirati a verificare la stabilità nel tempo dell'orientazione assegnata dall'algoritmo, utilizzati per verificare la coerenza della soluzione da esso generata;
- quelli mirati a verificare la sensibilità dell'algoritmo nei confronti di cambiamenti di relazione resi noti da altre fonti.

I risultati di tutte le prove sono stati elaborati per mezzo dell'ambiente presentato nel capitolo 3.

5.1 Test basato su dati ottenuti da più punti di vista

Questo test si basa su dati disponibili all'indirizzo [9]. Lo scopo è quello di fornire una prima convalida della bontà del comportamento dell'algoritmo sulla base di dati disponibili su un intervallo temporale relativamente ampio. Il test utilizza l'implementazione bgpPathSat dell'algoritmo di [20] precedente alla fase di ottimizzazione.

5.1.1 Criteri utilizzati per la scelta dei dati

I dati messi a disposizione da [9] si presentano come organizzati in modo abbastanza sistematico. Il loro utilizzo, tuttavia, può risultare proficuo per eseguire istanze isolate di un algoritmo di inferenza, ma diventa particolarmente problematico quando si debba analizzare l'evoluzione dei risultati da esso prodotti. I dati in questione, infatti, manifestano almeno i seguenti tipi di anomalie:

- il ritmo con cui sono resi disponibili non è assolutamente regolare: alcune coppie di snapshot sono distanziate di circa un mese, mentre per altre si arriva fino a 9 mesi; a questo problema, per il momento, non si è posto rimedio;
- i contenuti delle tabelle BGP presentate riportano spesso caratteri "spuri" (#, &, !, >, $, ", ecc.); le righe con simboli invalidi sono state filtrate per mezzo del modulo di estrazione degli AS path di TORQUE;
- le liste di cammini pubblicate spesso non includono tutti i cammini che effettivamente è possibile estrarre dalle corrispondenti tabelle BGP; per ovviare a questo inconveniente, i cammini sono stati ogni volta riestratti dalle tabelle BGP stesse per mezzo del modulo di estrazione degli AS path di TORQUE;
- i looking glass utilizzati in ogni snapshot non sono sempre gli stessi, ma oscillano sia come quantità (da 8 a 14) che come posizione; per ottenere una visione il più possibile coerente sono stati mantenuti soltanto i looking glass illustrati nella tabella 5.1; lo snapshot del 9 luglio 2002 è stato eliminato del tutto perché privo del contributo del server Oregon Route Views.

5.1.2 Procedura seguita

Per inferire le relazioni sono stati eseguiti i passi riportati nella figura 5.1. Il passo numero 6, che, di fatto, influenza soltanto la strada presa dall'euristica durante l'eliminazione dei *bad path*, si è reso necessario, poiché né la versione preesistente, né quella ottimizzata dell'algoritmo utilizzano particolari accorgimenti per velocizzare la gestione di cammini duplicati. D'altra parte, un'ulteriore modifica dell'algoritmo stesso in tal senso avrebbe avuto un impatto troppo marginale rispetto al tempo richiesto per apportarla.

1.	prelievo degli snapshot marcati nella tabella 5.1
2.	per ogni snapshot relativo ad una certa data
3.	per ogni tabella BGP disponibile per lo snapshot
4.	estrazione degli AS path utilizzando l'apposito modulo dell'ambiente TORQUE (con i parametri di default; si veda la sezione 3.3.2 per maggiori dettagli)
5.	concatenazione delle liste di AS path estratti
6.	rimozione dei path duplicati
7.	applicazione dell'euristica per lo scarto dei *bad path*
8.	applicazione del reinserimento dei *bad path*
9.	calcolo di un'orientazione sull'insieme massimale di cammini soddisfacibili

Figura 5.1: Procedura seguita per l'inferenza delle relazioni

Looking glass		Snapshot					
# AS	Nome	18 Apr 2001	29 Gen 2002 – 04 Feb 2002	06 Apr 2002	29 Lug 2002	09 Ago 2002	19 Ott 2002
1	Genuity	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
1740	CERFnet	$\surd$					
1838	CERFnet-2			$\surd$	$\surd$	$\surd$	$\surd$
3257	Tiscali				$\surd$	$\surd$	
3549	Global Crossing	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	
3582	University of Oregon	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
3967	Exodus Communications	$\surd$	$\surd$	$\surd$	$\surd$		$\surd$
4197	Exodus Communications (Global OnLine Japan)	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	
5388	Energis	$\times$	$\times$	$\times$			
5511	OpenTransit France Telecom			$\surd$	$\surd$	$\surd$	$\surd$
6539	GT Group Telecom			$\times$			
7018	AT&T WorldNet	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
8220	COLT Internet	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
8709	Exodus Europe	$\surd$	$\surd$	$\surd$	$\surd$		
9328	GlobalCenter			$\times$			
15290	AT&T Canada			$\surd$	$\surd$	$\surd$	$\surd$

Tabella 5.1: Looking glass utilizzati nel test ($\surd$=utilizzato; $\times$=disponibile ma non utilizzato)

In questa prima computazione è stata utilizzata la versione preesistente di bgpPathSat. La limitata quantità di dati a disposizione, infatti, non aveva ancora reso eccessivamente pressante il problema dell'efficienza.

5.1.3 Risultati ottenuti

I nodi e gli archi riportati in corrispondenza di ogni snapshot costituiscono i nodi e gli archi del grafo degli AS, così come risultano visibili dallo snapshot in questione. Si consideri che l'algoritmo in esame non produce archi di peering, quindi gli archi riportati come "non orientati" sono archi sui quali l'inferenza ha fallito.

Le quantità di cammini riportate si riferiscono a cammini unici (senza duplicati).

Ciascun valore percentuale è riferito al totale gerarchicamente superiore.

Il *grafo unione* è il grafo ottenuto inserendo tutti gli archi che sono visibili in almeno uno snapshot, ed i relativi nodi terminali.

18 Aprile 2001			
AS path	totale		511200
	Bad paths	estratti	47977
		reinseriti	43681 (91 %)
Iterazioni	estrazione *bad paths*		380
	reinserimento *bad paths*		20859
Rapporto iterazioni/path reinseriti			0.48
29 Gennaio 2002 – 04 Febbraio 2002			
AS path	totale		722481
	Bad paths	estratti	25359
		reinseriti	20490 (81 %)
Iterazioni	estrazione *bad paths*		284
	reinserimento *bad paths*		16417
Rapporto iterazioni/path reinseriti			0.8
06 Aprile 2002			
AS path	totale		942382
	Bad paths	estratti	30142
		reinseriti	23881 (79 %)
Iterazioni	estrazione *bad paths*		282
	reinserimento *bad paths*		20390
Rapporto iterazioni/path reinseriti			0.85
29 Luglio 2002			
AS path	totale		948720
	Bad paths	estratti	29479
		reinseriti	24504 (83 %)
Iterazioni	estrazione *bad paths*		266
	reinserimento *bad paths*		18820
Rapporto iterazioni/path reinseriti			0.77
09 Agosto 2002			
AS path	totale		894396
	Bad paths	estratti	26816
		reinseriti	22904 (85 %)
Iterazioni	estrazione *bad paths*		290
	reinserimento *bad paths*		17438
Rapporto iterazioni/path reinseriti			0.76
19 Ottobre 2002			
AS path	totale		881836
	Bad paths	estratti	20029
		reinseriti	15484 (77 %)
Iterazioni	estrazione *bad paths*		257
	reinserimento *bad paths*		15950
Rapporto iterazioni/path reinseriti			1.03

Tabella 5.2: Caratteristiche del processo di inferenza

G_1: orientazione calcolata per il 18 Aprile 2001
G_2: orientazione calcolata per il 29 Gennaio 2002 – 04 Febbraio 2002

Nodi in G_1	totale		13997
	solo in G_1		1289 (9 %)
Nodi in G_2	totale		12708
	solo in G_2		3088 (24 %)
Nodi comuni	totale		9620
Archi in G_1	totale		23817
	orientati		23776 (100 %)
	non orientati		41 (0 %)
	solo in G_1	totale	7873 (33 %)
		orientati	7850 (100 %)
		non orientati	23 (0 %)
Archi in G_2	totale		27555
	orientati		27530 (100 %)
	non orientati		25 (0 %)
	solo in G_2	totale	11611 (42 %)
		orientati	11593 (100 %)
		non orientati	18 (0 %)
Archi comuni	totale		15944
	orientati solo in G_1		7 (0 %)
	orientati solo in G_2		18 (0 %)
	non orientati in entrambi		0 (0 %)
	orientati in entrambi	totale	15919 (100 %)
		concordi	15118 (95 %)
		discordi	801 (5 %)

Tabella 5.3: Comparazione tra due snapshot

G_1: orientazione calcolata per il 29 Gennaio 2002 – 04 Febbraio 2002
G_2: orientazione calcolata per il 06 Aprile 2002

Nodi in G_1	totale		13423
	solo in G_1		344 (3 %)
Nodi in G_2	totale		13079
	solo in G_2		715 (5 %)
Nodi comuni	totale		12364
Archi in G_1	totale		27555
	orientati		27530 (100 %)
	non orientati		25 (0 %)
	solo in G_1	totale	2802 (10 %)
		orientati	2786 (99 %)
		non orientati	16 (1 %)
Archi in G_2	totale		28309
	orientati		28303 (100 %)
	non orientati		6 (0 %)
	solo in G_2	totale	3556 (13 %)
		orientati	3551 (100 %)
		non orientati	5 (0 %)
Archi comuni	totale		24753
	orientati solo in G_1		1 (0 %)
	orientati solo in G_2		9 (0 %)
	non orientati in entrambi		0 (0 %)
	orientati in entrambi	totale	24743 (100 %)
		concordi	24112 (97 %)
		discordi	631 (3 %)

Tabella 5.4: Comparazione tra due snapshot

G_1: orientazione calcolata per il 06 Aprile 2002
G_2: orientazione calcolata per il 29 Luglio 2002

Nodi in G_1	totale		14251
	solo in G_1		545 (4 %)
Nodi in G_2	totale		13706
	solo in G_2		1172 (9 %)
Nodi comuni	totale		12534
Archi in G_1	totale		28309
	orientati		28303 (100 %)
	non orientati		6 (0 %)
	solo in G_1	totale	4668 (16 %)
		orientati	4663 (100 %)
		non orientati	5 (0 %)
Archi in G_2	totale		29074
	orientati		29067 (100 %)
	non orientati		7 (0 %)
	solo in G_2	totale	5433 (19 %)
		orientati	5428 (100 %)
		non orientati	5 (0 %)
Archi comuni	totale		23641
	orientati solo in G_1		2 (0 %)
	orientati solo in G_2		1 (0 %)
	non orientati in entrambi		0 (0 %)
	orientati in entrambi	totale	23638 (100 %)
		concordi	23044 (97 %)
		discordi	594 (3 %)

Tabella 5.5: Comparazione tra due snapshot

G_1: orientazione calcolata per il 29 Luglio 2002
G_2: orientazione calcolata per il 09 Agosto 2002

Nodi in G_1	totale		13848
	solo in G_1		94 (1 %)
Nodi in G_2	totale		13754
	solo in G_2		142 (1 %)
Nodi comuni	totale		13612
Archi in G_1	totale		29074
	orientati		29067 (100 %)
	non orientati		7 (0 %)
	solo in G_1	totale	1025 (4 %)
		orientati	1022 (100 %)
		non orientati	3 (0 %)
Archi in G_2	totale		29009
	orientati		28998 (100 %)
	non orientati		11 (0 %)
	solo in G_2	totale	960 (3 %)
		orientati	957 (100 %)
		non orientati	3 (0 %)
Archi comuni	totale		28049
	orientati solo in G_1		5 (0 %)
	orientati solo in G_2		1 (0 %)
	non orientati in entrambi		3 (0 %)
	orientati in entrambi	totale	28040 (100 %)
		concordi	27720 (99 %)
		discordi	320 (1 %)

Tabella 5.6: Comparazione tra due snapshot

G_1: orientazione calcolata per il 09 Agosto 2002
G_2: orientazione calcolata per il 19 Ottobre 2002

Nodi in G_1	totale		14490
	solo in G_1		377 (3 %)
Nodi in G_2	totale		14113
	solo in G_2		736 (5 %)
Nodi comuni	totale		13377
Archi in G_1	totale		29009
	orientati		28998 (100 %)
	non orientati		11 (0 %)
	solo in G_1	totale	3545 (12 %)
		orientati	3537 (100 %)
		non orientati	8 (0 %)
Archi in G_2	totale		29422
	orientati		29408 (100 %)
	non orientati		14 (0 %)
	solo in G_2	totale	3958 (13 %)
		orientati	3948 (100 %)
		non orientati	10 (0 %)
Archi comuni	totale		25464
	orientati solo in G_1		3 (0 %)
	orientati solo in G_2		2 (0 %)
	non orientati in entrambi		1 (0 %)
	orientati in entrambi	totale	25458 (100 %)
		concordi	24916 (98 %)
		discordi	542 (2 %)

Tabella 5.7: Comparazione tra due snapshot

Numero di nodi	16514
Numero di archi	47527

Tabella 5.8: Caratteristiche del grafo unione

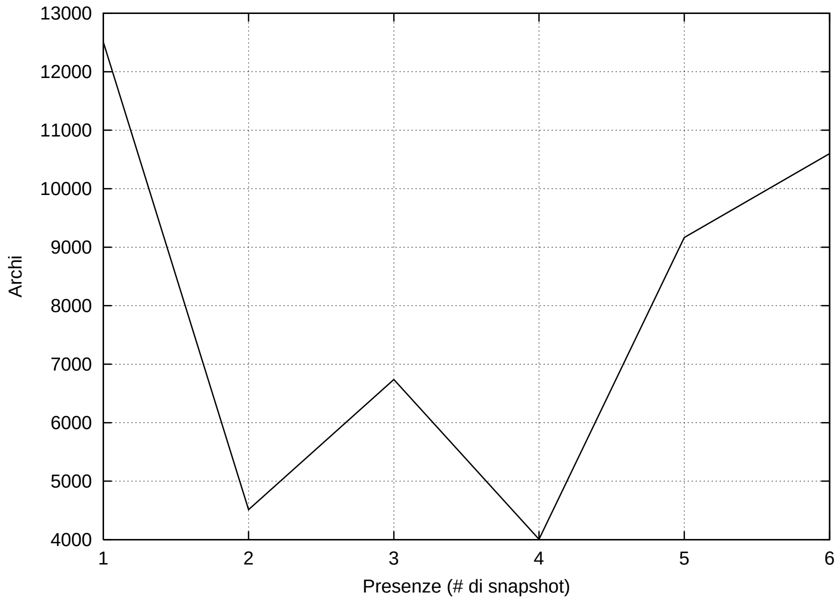

Figura 5.2: Distribuzione del numero di presenze di ciascun arco nei vari snapshot

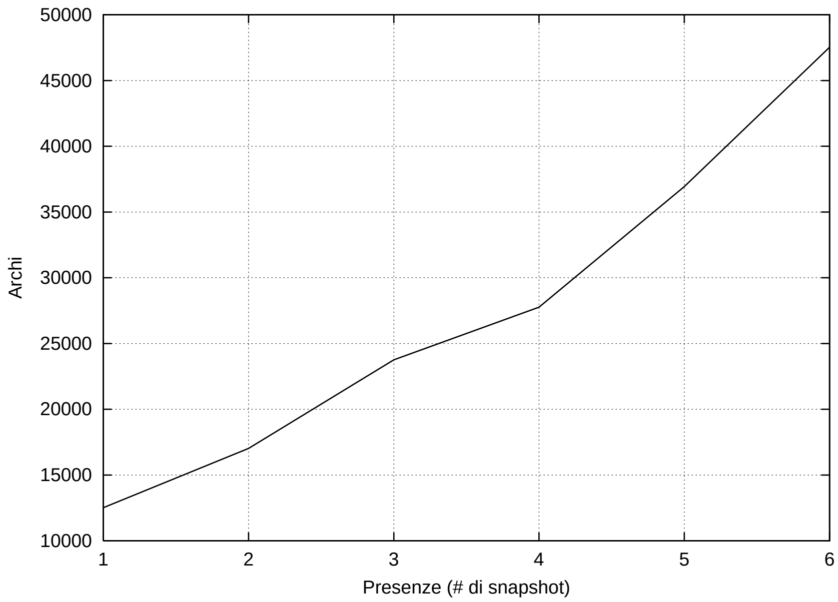

Figura 5.3: Distribuzione cumulativa del numero di presenze di ciascun arco nei vari snapshot (numero y di archi con al più x presenze)

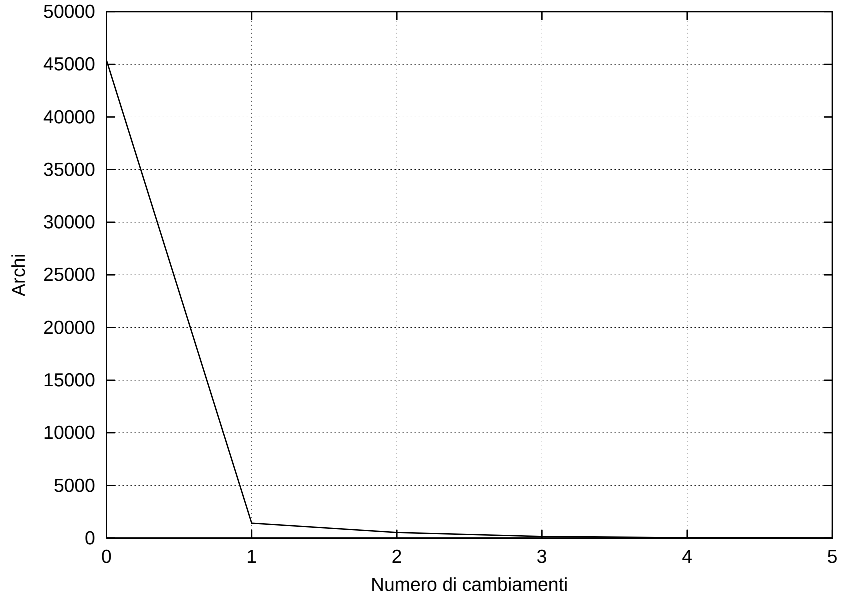

Figura 5.4: Distribuzione del numero di cambiamenti di orientazione che ciascun arco ha subito tra due snapshot consecutivi (inclusi i passaggi dallo stato orientato a quello non orientato e viceversa)

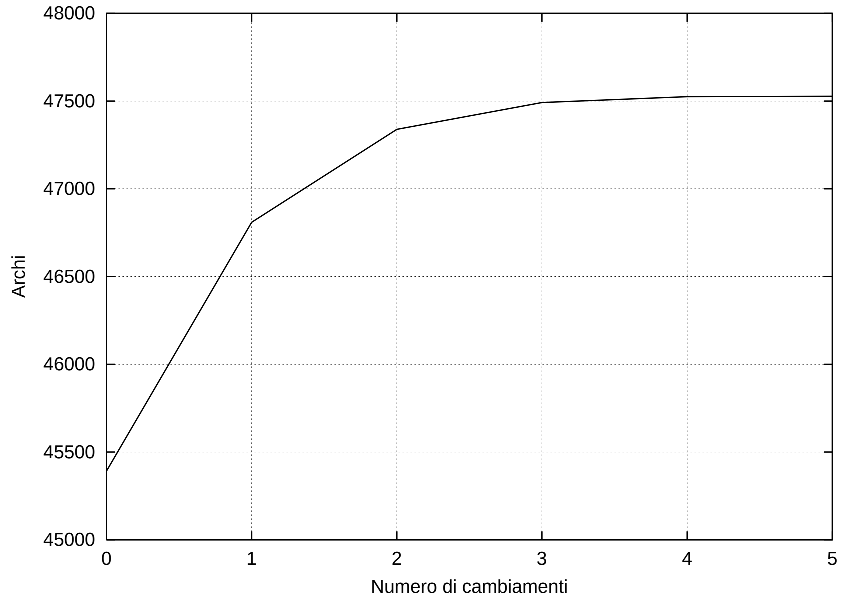

Figura 5.5: Distribuzione cumulativa del numero di cambiamenti di orientazione che ciascun arco ha subito tra due snapshot consecutivi (inclusi i passaggi dallo stato orientato a quello non orientato e viceversa)

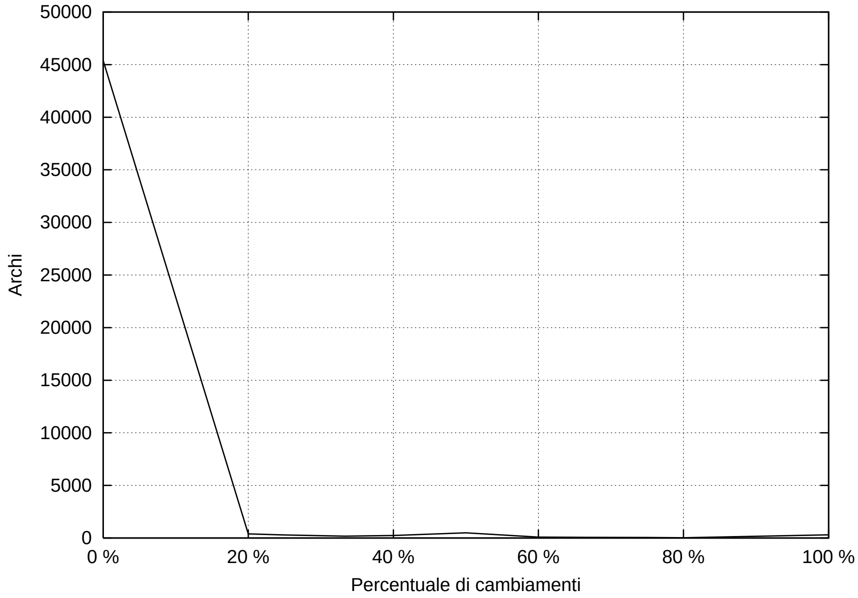

Figura 5.6: Distribuzione del rapporto tra i cambiamenti di orientazione che ciascun arco ha subito ed il numero delle sue presenze (ad esempio, gli archi riportati in corrispondenza del valore 100% hanno cambiato orientazione in ogni snapshot in cui sono stati presenti)

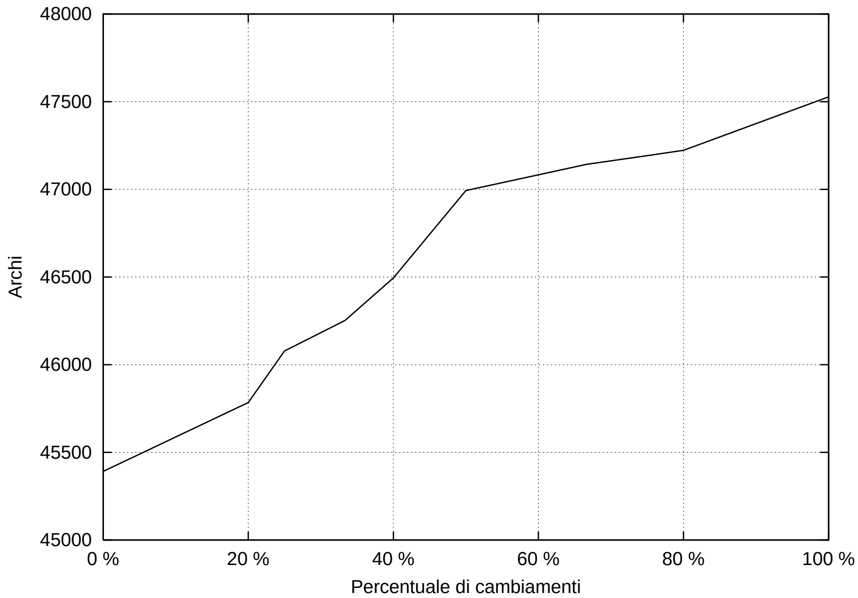

Figura 5.7: Distribuzione cumulativa del rapporto tra i cambiamenti di orientazione che ciascun arco ha subito ed il numero delle sue presenze

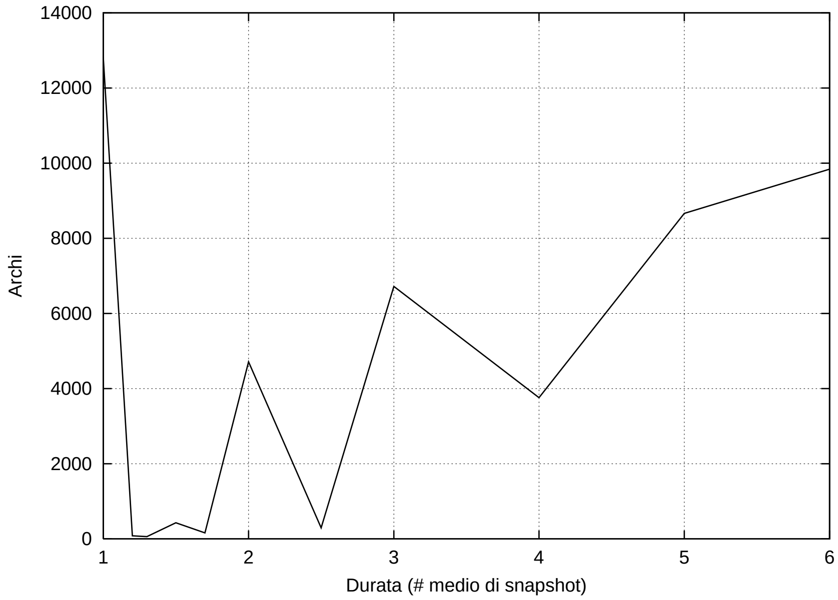

Figura 5.8: Distribuzione della durata media di un'orientazione (in termini di numero di snapshot consecutivi per cui viene mantenuta)

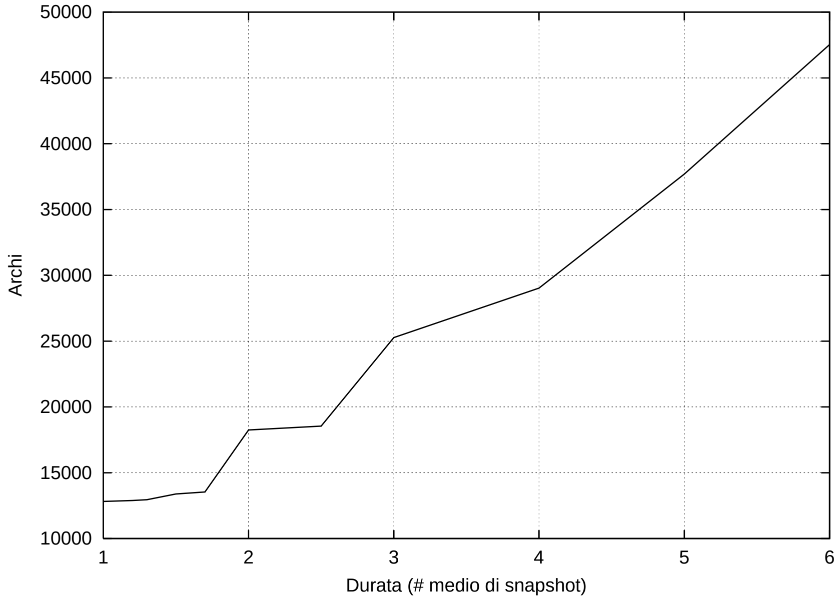

Figura 5.9: Distribuzione cumulativa della durata media di un'orientazione (in termini di numero di snapshot consecutivi per cui viene mantenuta)

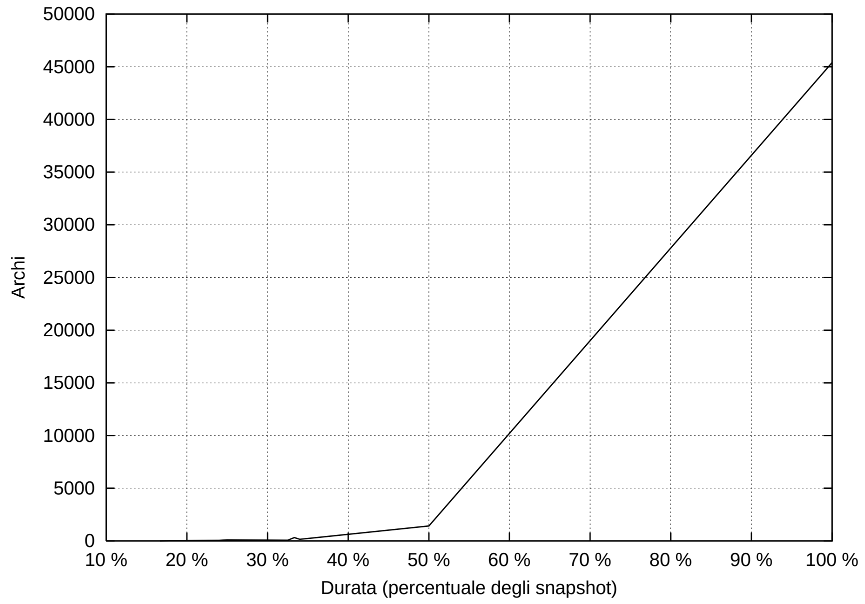

Figura 5.10: Distribuzione della durata media di un'orientazione (in termini di numero di snapshot consecutivi per cui viene mantenuta) rapportata al numero di presenze di ciascun arco (ad esempio, il valore in corrispondenza di 100% è il numero di archi che hanno mantenuto sempre stabile la propria orientazione)

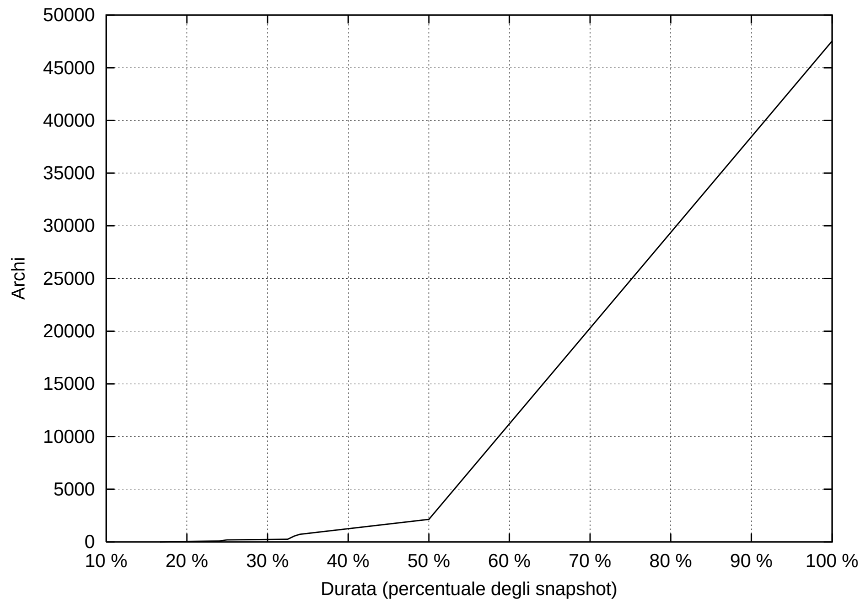

Figura 5.11: Distribuzione cumulativa della durata media di un'orientazione (in termini di numero di snapshot consecutivi per cui viene mantenuta) rapportata al numero di presenze di ciascun arco

5.1.4 Conclusioni

Nonostante le numerose anomalie presentate dai dati in ingresso, l'algoritmo sembra comportarsi molto bene, riuscendo ad orientare più di 45000 archi (95% del totale) in modo stabile. Anche se l'arco temporale considerato è relativamente ampio (più di un anno), è lecito aspettarsi che le variazioni commerciali di rilievo non siano eccessivamente numerose, e dunque questo dato può essere indice di un buon comportamento anche per quanto riguarda la sensibilità dell'algoritmo alla comparsa di nuovi accordi.

Durante le sperimentazioni, è stato interessante notare come le visioni di ampiezza relativamente ridotta (vantage point isolati) diano sempre luogo ad istanze soddisfacibili del problema ToR–D, mentre in presenza di dati di insieme (come quelli forniti da Oregon Route Views, che ha sessioni di peering con molti altri router) si manifesti una situazione di insoddisfacibilità.

5.2 Test su un periodo di relativa stabilità dal punto di vista commerciale

Una volta compiuta la fase di ottimizzazione presentata nel capitolo 4, è stato possibile eseguire un numero elevato di istanze dell'algoritmo di inferenza su dati diversi, per valutarne il comportamento in modo sistematico. Per il contesto della prima di queste prove si è scelto di individuare un periodo in cui non avessero avuto luogo cambiamenti commerciali significativi. In questo modo, è stato possibile verificare quanto ciò si rifletta sulla stabilità dell'orientazione calcolata.

5.2.1 Criteri utilizzati per la scelta dei dati

I dati utilizzati per questa prova sono stati prelevati dall'archivio del server Oregon Route Views ([22]). Al contrario di quelli utilizzati nella prova precedente, essi manifestano molti vantaggi: oltre ad essere privi delle anomalie presentate da questi ultimi, sono disponibili su un intervallo temporale molto ampio ed a cadenza regolare, il che fa sì che riescano a coprire sia periodi stabili, che periodi relativamente turbolenti per quello riguarda le variazioni negli accordi commerciali.

Il periodo temporale scelto è quello della settimana che va dal 25 Marzo 2003 al 31 Marzo 2003. Per individuare tale periodo ci si è serviti delle informazioni pubblicate da [24, 25], due siti Internet finalizzati a divulgare notizie che riguardano gli Internet Service Provider. In accordo con le informazioni da essi pubblicate, nella settimana considerata non ci sono stati eventi di particolare rilievo. La selezione del periodo è stata ulteriormente convalidata in base ai dati pubblicati in [26], in cui è possibile consultare informazioni quantitative sui dati di routing, che mostrano che nel periodo individuato non vi sono particolari episodi di carattere tecnico (guasti, riconfigurazioni, ecc.).

Gli snapshot disponibili per questa settimana sono 84 (uno ogni 2 ore).

5.2.2 Procedura seguita

La procedura utilizzata per l'inferenza è analoga a quella adottata nella prova precedente.

1.	prelievo degli snapshot selezionati
2.	per ogni snapshot
3.	estrazione degli AS path utilizzando l'apposito modulo dell'ambiente TORQUE (con i parametri di default; si veda la sezione 3.3.2 per maggiori dettagli)
4.	rimozione dei path duplicati
5.	applicazione dell'euristica per lo scarto dei *bad path*
6.	applicazione del reinserimento dei *bad path*
7.	calcolo di un'orientazione sull'insieme massimale di cammini soddisfacibili

Figura 5.12: Procedura seguita per l'inferenza delle relazioni

Per la prova è stata utilizzata l'implementazione dell'algoritmo di inferenza di [20] (presentato nella sezione 2.4), ottimizzata secondo quanto descritto nel capitolo 4. È stata proprio l'ottimizzazione stessa a rendere possibile questo tipo di sperimentazione, che, altrimenti, avrebbe richiesto decine di settimane per essere completata. Il guadagno in termini di velocità è stato reso possibile sia dalla ristrutturazione dell'algoritmo, sia dall'utilizzo del test di soddisfacibilità corretto, che ha permesso di terminare ogni computazione con meno iterazioni di quante ne servissero in precedenza.

5.2.3 Risultati ottenuti

L'interpretazione dei risultati è la stessa utilizzata per la prova precedente. In questo caso, al posto dei dati comparativi diretti tra coppie di snapshot, sono stati inseriti dei grafici che forniscono indizi sulla qualità dei dati utilizzati. In quello in figura 5.13 viene mostrato il numero di AS path ed il numero di nodi ed archi del grafo degli AS che sono visibili in ogni snapshot. Si osservi come, visti i valori presenti sulle due scale del grafico, non vi siano oscillazioni abbastanza notevoli da poter perturbare l'inferenza in modo evidente. Nella figura 5.14 è invece possibile osservare l'attività dei router con i quali il server Route Views ha sessioni di peering attive. In particolare, si noti come l'aumento nel numero di AS path estratti abbia luogo proprio in corrispondenza dell'attivazione di un nuovo peer.

Numero di nodi	15150
Numero di archi	32534

Tabella 5.9: Caratteristiche del grafo unione

25/03/2003 00:00			
AS path	totale		789764
	Bad paths	estratti	16414
		reinseriti	13874 (85 %)
Iterazioni	estrazione *bad paths*		268
	reinserimento *bad paths*		12398
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15038
	archi	totale	31205
		orientati	31195 (100 %)
		non orientati	10 (0 %)
25/03/2003 02:00			
AS path	totale		790344
	Bad paths	estratti	16912
		reinseriti	14370 (85 %)
Iterazioni	estrazione *bad paths*		280
	reinserimento *bad paths*		12330
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15037
	archi	totale	31202
		orientati	31188 (100 %)
		non orientati	14 (0 %)
25/03/2003 04:00			
AS path	totale		789930
	Bad paths	estratti	16343
		reinseriti	13985 (86 %)
Iterazioni	estrazione *bad paths*		272
	reinserimento *bad paths*		11938
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15034
	archi	totale	31184
		orientati	31175 (100 %)
		non orientati	9 (0 %)
25/03/2003 06:00			
AS path	totale		790329
	Bad paths	estratti	15855
		reinseriti	13434 (85 %)
Iterazioni	estrazione *bad paths*		272
	reinserimento *bad paths*		12191
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15038
	archi	totale	31184
		orientati	31178 (100 %)
		non orientati	6 (0 %)

Tabella 5.10: Caratteristiche del processo di inferenza

25/03/2003 08:00			
AS path	totale		790690
	Bad paths	estratti	16041
		reinseriti	13675 (85 %)
Iterazioni	estrazione *bad paths*		271
	reinserimento *bad paths*		12068
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15041
	archi	totale	31213
		orientati	31204 (100 %)
		non orientati	9 (0 %)
25/03/2003 10:00			
AS path	totale		790353
	Bad paths	estratti	17096
		reinseriti	14605 (85 %)
Iterazioni	estrazione *bad paths*		272
	reinserimento *bad paths*		12523
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15045
	archi	totale	31203
		orientati	31196 (100 %)
		non orientati	7 (0 %)
25/03/2003 12:00			
AS path	totale		793646
	Bad paths	estratti	15971
		reinseriti	13539 (85 %)
Iterazioni	estrazione *bad paths*		278
	reinserimento *bad paths*		12080
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15043
	archi	totale	31212
		orientati	31200 (100 %)
		non orientati	12 (0 %)
25/03/2003 14:00			
AS path	totale		790744
	Bad paths	estratti	16693
		reinseriti	14266 (85 %)
Iterazioni	estrazione *bad paths*		280
	reinserimento *bad paths*		12495
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15045
	archi	totale	31205
		orientati	31199 (100 %)
		non orientati	6 (0 %)

Tabella 5.11: Caratteristiche del processo di inferenza

25/03/2003 16:00			
AS path	totale		790425
	Bad paths	estratti	17109
		reinseriti	14611 (85 %)
Iterazioni	estrazione *bad paths*		281
	reinserimento *bad paths*		12488
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15044
	archi	totale	31217
		orientati	31210 (100 %)
		non orientati	7 (0 %)

25/03/2003 18:00			
AS path	totale		792248
	Bad paths	estratti	16277
		reinseriti	13791 (85 %)
Iterazioni	estrazione *bad paths*		279
	reinserimento *bad paths*		12130
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15048
	archi	totale	31201
		orientati	31193 (100 %)
		non orientati	8 (0 %)

25/03/2003 20:00			
AS path	totale		792441
	Bad paths	estratti	17225
		reinseriti	14620 (85 %)
Iterazioni	estrazione *bad paths*		276
	reinserimento *bad paths*		12703
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15045
	archi	totale	31202
		orientati	31193 (100 %)
		non orientati	9 (0 %)

25/03/2003 22:00			
AS path	totale		792773
	Bad paths	estratti	16626
		reinseriti	14098 (85 %)
Iterazioni	estrazione *bad paths*		284
	reinserimento *bad paths*		12260
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15046
	archi	totale	31214
		orientati	31208 (100 %)
		non orientati	6 (0 %)

Tabella 5.12: Caratteristiche del processo di inferenza

26/03/2003 00:00			
AS path	totale		792548
	Bad paths	estratti	17615
		reinseriti	15092 (86 %)
Iterazioni	estrazione *bad paths*		281
	reinserimento *bad paths*		12821
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15045
	archi	totale	31203
		orientati	31195 (100 %)
		non orientati	8 (0 %)

26/03/2003 02:00			
AS path	totale		792295
	Bad paths	estratti	16482
		reinseriti	13938 (85 %)
Iterazioni	estrazione *bad paths*		285
	reinserimento *bad paths*		12544
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15043
	archi	totale	31213
		orientati	31205 (100 %)
		non orientati	8 (0 %)

26/03/2003 04:00			
AS path	totale		792285
	Bad paths	estratti	17182
		reinseriti	14568 (85 %)
Iterazioni	estrazione *bad paths*		274
	reinserimento *bad paths*		12695
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15038
	archi	totale	31210
		orientati	31201 (100 %)
		non orientati	9 (0 %)

26/03/2003 06:00			
AS path	totale		792595
	Bad paths	estratti	17959
		reinseriti	15353 (85 %)
Iterazioni	estrazione *bad paths*		291
	reinserimento *bad paths*		13417
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15044
	archi	totale	31240
		orientati	31219 (100 %)
		non orientati	21 (0 %)

Tabella 5.13: Caratteristiche del processo di inferenza

26/03/2003 08:00			
AS path	totale		792903
	Bad paths	estratti	17903
		reinseriti	15351 (86 %)
Iterazioni	estrazione *bad paths*		286
	reinserimento *bad paths*		12807
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15045
	archi	totale	31199
		orientati	31190 (100 %)
		non orientati	9 (0 %)
26/03/2003 10:00			
AS path	totale		792063
	Bad paths	estratti	17789
		reinseriti	15133 (85 %)
Iterazioni	estrazione *bad paths*		290
	reinserimento *bad paths*		13288
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15044
	archi	totale	31208
		orientati	31197 (100 %)
		non orientati	11 (0 %)
26/03/2003 12:00			
AS path	totale		793143
	Bad paths	estratti	16698
		reinseriti	14183 (85 %)
Iterazioni	estrazione *bad paths*		280
	reinserimento *bad paths*		12495
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15046
	archi	totale	31212
		orientati	31202 (100 %)
		non orientati	10 (0 %)
26/03/2003 14:00			
AS path	totale		793230
	Bad paths	estratti	18622
		reinseriti	16006 (86 %)
Iterazioni	estrazione *bad paths*		280
	reinserimento *bad paths*		12884
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15047
	archi	totale	31204
		orientati	31197 (100 %)
		non orientati	7 (0 %)

Tabella 5.14: Caratteristiche del processo di inferenza

26/03/2003 16:00			
AS path	totale		792755
	Bad paths	estratti	18096
		reinseriti	15559 (86 %)
Iterazioni	estrazione *bad paths*		278
	reinserimento *bad paths*		12992
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15050
	archi	totale	31216
		orientati	31209 (100 %)
		non orientati	7 (0 %)
26/03/2003 18:00			
AS path	totale		792951
	Bad paths	estratti	16483
		reinseriti	13938 (85 %)
Iterazioni	estrazione *bad paths*		280
	reinserimento *bad paths*		12894
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15051
	archi	totale	31215
		orientati	31201 (100 %)
		non orientati	14 (0 %)
26/03/2003 20:00			
AS path	totale		793156
	Bad paths	estratti	16880
		reinseriti	14341 (85 %)
Iterazioni	estrazione *bad paths*		278
	reinserimento *bad paths*		12493
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15054
	archi	totale	31228
		orientati	31217 (100 %)
		non orientati	11 (0 %)
26/03/2003 22:00			
AS path	totale		793272
	Bad paths	estratti	16347
		reinseriti	13581 (83 %)
Iterazioni	estrazione *bad paths*		281
	reinserimento *bad paths*		12894
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15049
	archi	totale	31226
		orientati	31216 (100 %)
		non orientati	10 (0 %)

Tabella 5.15: Caratteristiche del processo di inferenza

27/03/2003 00:00			
AS path	totale		792404
	Bad paths	estratti	15605
		reinseriti	13072 (84 %)
Iterazioni	estrazione *bad paths*		271
	reinserimento *bad paths*		12166
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15044
	archi	totale	31216
		orientati	31207 (100 %)
		non orientati	9 (0 %)
27/03/2003 02:00			
AS path	totale		792611
	Bad paths	estratti	15925
		reinseriti	13274 (83 %)
Iterazioni	estrazione *bad paths*		281
	reinserimento *bad paths*		12523
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15046
	archi	totale	31217
		orientati	31207 (100 %)
		non orientati	10 (0 %)
27/03/2003 04:00			
AS path	totale		795311
	Bad paths	estratti	16002
		reinseriti	13444 (84 %)
Iterazioni	estrazione *bad paths*		279
	reinserimento *bad paths*		12506
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15043
	archi	totale	31213
		orientati	31202 (100 %)
		non orientati	11 (0 %)
27/03/2003 06:00			
AS path	totale		793474
	Bad paths	estratti	17236
		reinseriti	14600 (85 %)
Iterazioni	estrazione *bad paths*		289
	reinserimento *bad paths*		12866
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15045
	archi	totale	31227
		orientati	31217 (100 %)
		non orientati	10 (0 %)

Tabella 5.16: Caratteristiche del processo di inferenza

27/03/2003 08:00			
AS path	totale		793405
	Bad paths	estratti	18146
		reinseriti	15330 (84 %)
Iterazioni	estrazione *bad paths*		284
	reinserimento *bad paths*		13421
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15043
	archi	totale	31220
		orientati	31212 (100 %)
		non orientati	8 (0 %)

27/03/2003 10:00			
AS path	totale		792885
	Bad paths	estratti	16370
		reinseriti	13554 (83 %)
Iterazioni	estrazione *bad paths*		287
	reinserimento *bad paths*		13024
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15047
	archi	totale	31235
		orientati	31227 (100 %)
		non orientati	8 (0 %)

27/03/2003 12:00			
AS path	totale		792560
	Bad paths	estratti	18367
		reinseriti	15735 (86 %)
Iterazioni	estrazione *bad paths*		283
	reinserimento *bad paths*		13250
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15041
	archi	totale	31217
		orientati	31207 (100 %)
		non orientati	10 (0 %)

27/03/2003 14:00			
AS path	totale		792645
	Bad paths	estratti	18056
		reinseriti	15444 (86 %)
Iterazioni	estrazione *bad paths*		290
	reinserimento *bad paths*		13332
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15045
	archi	totale	31222
		orientati	31213 (100 %)
		non orientati	9 (0 %)

Tabella 5.17: Caratteristiche del processo di inferenza

27/03/2003 16:00			
AS path	totale		792791
	Bad paths	estratti	17257
		reinseriti	14722 (85 %)
Iterazioni	estrazione *bad paths*		279
	reinserimento *bad paths*		13170
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15048
	archi	totale	31219
		orientati	31210 (100 %)
		non orientati	9 (0 %)
27/03/2003 18:00			
AS path	totale		792714
	Bad paths	estratti	15954
		reinseriti	13365 (84 %)
Iterazioni	estrazione *bad paths*		285
	reinserimento *bad paths*		12466
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15047
	archi	totale	31208
		orientati	31201 (100 %)
		non orientati	7 (0 %)
27/03/2003 20:00			
AS path	totale		793827
	Bad paths	estratti	17132
		reinseriti	14617 (85 %)
Iterazioni	estrazione *bad paths*		276
	reinserimento *bad paths*		12744
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15048
	archi	totale	31213
		orientati	31205 (100 %)
		non orientati	8 (0 %)
27/03/2003 22:00			
AS path	totale		793946
	Bad paths	estratti	17127
		reinseriti	14588 (85 %)
Iterazioni	estrazione *bad paths*		278
	reinserimento *bad paths*		12650
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15051
	archi	totale	31227
		orientati	31221 (100 %)
		non orientati	6 (0 %)

Tabella 5.18: Caratteristiche del processo di inferenza

28/03/2003 00:00			
AS path	totale		793002
	Bad paths	estratti	17430
		reinseriti	14942 (86 %)
Iterazioni	estrazione *bad paths*		272
	reinserimento *bad paths*		12375
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15052
	archi	totale	31213
		orientati	31206 (100 %)
		non orientati	7 (0 %)
28/03/2003 02:00			
AS path	totale		792306
	Bad paths	estratti	17904
		reinseriti	15384 (86 %)
Iterazioni	estrazione *bad paths*		285
	reinserimento *bad paths*		12681
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15053
	archi	totale	31227
		orientati	31220 (100 %)
		non orientati	7 (0 %)
28/03/2003 04:00			
AS path	totale		792750
	Bad paths	estratti	16459
		reinseriti	13721 (83 %)
Iterazioni	estrazione *bad paths*		277
	reinserimento *bad paths*		12995
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15050
	archi	totale	31216
		orientati	31209 (100 %)
		non orientati	7 (0 %)
28/03/2003 06:00			
AS path	totale		793487
	Bad paths	estratti	18183
		reinseriti	15612 (86 %)
Iterazioni	estrazione *bad paths*		286
	reinserimento *bad paths*		12875
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15051
	archi	totale	31204
		orientati	31197 (100 %)
		non orientati	7 (0 %)

Tabella 5.19: Caratteristiche del processo di inferenza

28/03/2003 08:00			
AS path	totale		792768
	Bad paths	estratti	17693
		reinseriti	15034 (85 %)
Iterazioni	estrazione *bad paths*		272
	reinserimento *bad paths*		13149
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15052
	archi	totale	31228
		orientati	31218 (100 %)
		non orientati	10 (0 %)
28/03/2003 10:00			
AS path	totale		792885
	Bad paths	estratti	16558
		reinseriti	13799 (83 %)
Iterazioni	estrazione *bad paths*		277
	reinserimento *bad paths*		13564
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15056
	archi	totale	31227
		orientati	31216 (100 %)
		non orientati	11 (0 %)
28/03/2003 12:00			
AS path	totale		793013
	Bad paths	estratti	17002
		reinseriti	14290 (84 %)
Iterazioni	estrazione *bad paths*		279
	reinserimento *bad paths*		13628
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15057
	archi	totale	31220
		orientati	31211 (100 %)
		non orientati	9 (0 %)
28/03/2003 14:00			
AS path	totale		793288
	Bad paths	estratti	17344
		reinseriti	14781 (85 %)
Iterazioni	estrazione *bad paths*		286
	reinserimento *bad paths*		12942
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15057
	archi	totale	31251
		orientati	31245 (100 %)
		non orientati	6 (0 %)

Tabella 5.20: Caratteristiche del processo di inferenza

28/03/2003 16:00			
AS path	totale		793512
	Bad paths	estratti	17725
		reinseriti	15221 (86 %)
Iterazioni	estrazione *bad paths*		276
	reinserimento *bad paths*		12987
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15064
	archi	totale	31256
		orientati	31250 (100 %)
		non orientati	6 (0 %)
28/03/2003 18:00			
AS path	totale		793867
	Bad paths	estratti	17031
		reinseriti	14460 (85 %)
Iterazioni	estrazione *bad paths*		282
	reinserimento *bad paths*		12859
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15063
	archi	totale	31240
		orientati	31232 (100 %)
		non orientati	8 (0 %)
28/03/2003 20:00			
AS path	totale		793723
	Bad paths	estratti	17044
		reinseriti	14525 (85 %)
Iterazioni	estrazione *bad paths*		274
	reinserimento *bad paths*		12644
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15060
	archi	totale	31211
		orientati	31206 (100 %)
		non orientati	5 (0 %)
28/03/2003 22:00			
AS path	totale		812643
	Bad paths	estratti	18232
		reinseriti	15626 (86 %)
Iterazioni	estrazione *bad paths*		275
	reinserimento *bad paths*		13165
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15064
	archi	totale	31278
		orientati	31273 (100 %)
		non orientati	5 (0 %)

Tabella 5.21: Caratteristiche del processo di inferenza

29/03/2003 00:00			
AS path	totale		812039
	Bad paths	estratti	17755
		reinseriti	15348 (86 %)
Iterazioni	estrazione *bad paths*		272
	reinserimento *bad paths*		12717
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15063
	archi	totale	31265
		orientati	31259 (100 %)
		non orientati	6 (0 %)

29/03/2003 02:00			
AS path	totale		811660
	Bad paths	estratti	16456
		reinseriti	13874 (84 %)
Iterazioni	estrazione *bad paths*		273
	reinserimento *bad paths*		12569
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15059
	archi	totale	31284
		orientati	31276 (100 %)
		non orientati	8 (0 %)

29/03/2003 04:00			
AS path	totale		812011
	Bad paths	estratti	17486
		reinseriti	14883 (85 %)
Iterazioni	estrazione *bad paths*		267
	reinserimento *bad paths*		12514
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15061
	archi	totale	31260
		orientati	31255 (100 %)
		non orientati	5 (0 %)

29/03/2003 06:00			
AS path	totale		812412
	Bad paths	estratti	16420
		reinseriti	13743 (84 %)
Iterazioni	estrazione *bad paths*		274
	reinserimento *bad paths*		12796
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15065
	archi	totale	31273
		orientati	31267 (100 %)
		non orientati	6 (0 %)

Tabella 5.22: Caratteristiche del processo di inferenza

29/03/2003 08:00			
AS path	totale		812063
	Bad paths	estratti	18389
		reinseriti	15766 (86 %)
Iterazioni	estrazione *bad paths*		271
	reinserimento *bad paths*		13094
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15062
	archi	totale	31290
		orientati	31284 (100 %)
		non orientati	6 (0 %)
29/03/2003 10:00			
AS path	totale		811903
	Bad paths	estratti	19250
		reinseriti	16532 (86 %)
Iterazioni	estrazione *bad paths*		290
	reinserimento *bad paths*		13939
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15064
	archi	totale	31298
		orientati	31292 (100 %)
		non orientati	6 (0 %)
29/03/2003 12:00			
AS path	totale		812001
	Bad paths	estratti	17243
		reinseriti	14481 (84 %)
Iterazioni	estrazione *bad paths*		279
	reinserimento *bad paths*		13297
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15064
	archi	totale	31302
		orientati	31296 (100 %)
		non orientati	6 (0 %)
29/03/2003 14:00			
AS path	totale		811994
	Bad paths	estratti	18387
		reinseriti	15670 (85 %)
Iterazioni	estrazione *bad paths*		286
	reinserimento *bad paths*		13694
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15060
	archi	totale	31295
		orientati	31290 (100 %)
		non orientati	5 (0 %)

Tabella 5.23: Caratteristiche del processo di inferenza

29/03/2003 16:00			
AS path	totale		812761
	Bad paths	estratti	18327
		reinseriti	15614 (85 %)
Iterazioni	estrazione *bad paths*		285
	reinserimento *bad paths*		13432
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15064
	archi	totale	31303
		orientati	31296 (100 %)
		non orientati	7 (0 %)
29/03/2003 18:00			
AS path	totale		812868
	Bad paths	estratti	17071
		reinseriti	14342 (84 %)
Iterazioni	estrazione *bad paths*		282
	reinserimento *bad paths*		13251
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15064
	archi	totale	31301
		orientati	31295 (100 %)
		non orientati	6 (0 %)
29/03/2003 20:00			
AS path	totale		812943
	Bad paths	estratti	17091
		reinseriti	14334 (84 %)
Iterazioni	estrazione *bad paths*		280
	reinserimento *bad paths*		13410
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15064
	archi	totale	31296
		orientati	31290 (100 %)
		non orientati	6 (0 %)
29/03/2003 22:00			
AS path	totale		812370
	Bad paths	estratti	17153
		reinseriti	14354 (84 %)
Iterazioni	estrazione *bad paths*		284
	reinserimento *bad paths*		13430
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15058
	archi	totale	31277
		orientati	31271 (100 %)
		non orientati	6 (0 %)

Tabella 5.24: Caratteristiche del processo di inferenza

30/03/2003 00:00			
AS path	totale		811235
	Bad paths	estratti	18027
		reinseriti	15305 (85 %)
Iterazioni	estrazione *bad paths*		281
	reinserimento *bad paths*		13315
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15038
	archi	totale	31269
		orientati	31258 (100 %)
		non orientati	11 (0 %)
30/03/2003 02:00			
AS path	totale		810934
	Bad paths	estratti	16854
		reinseriti	14207 (84 %)
Iterazioni	estrazione *bad paths*		279
	reinserimento *bad paths*		13059
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15034
	archi	totale	31258
		orientati	31253 (100 %)
		non orientati	5 (0 %)
30/03/2003 04:00			
AS path	totale		810761
	Bad paths	estratti	19166
		reinseriti	15698 (82 %)
Iterazioni	estrazione *bad paths*		282
	reinserimento *bad paths*		14641
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15035
	archi	totale	31258
		orientati	31252 (100 %)
		non orientati	6 (0 %)
30/03/2003 06:00			
AS path	totale		810852
	Bad paths	estratti	19065
		reinseriti	15584 (82 %)
Iterazioni	estrazione *bad paths*		292
	reinserimento *bad paths*		14382
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15031
	archi	totale	31269
		orientati	31259 (100 %)
		non orientati	10 (0 %)

Tabella 5.25: Caratteristiche del processo di inferenza

30/03/2003 08:00			
AS path	totale		810913
	Bad paths	estratti	16472
		reinseriti	13692 (83 %)
Iterazioni	estrazione *bad paths*		282
	reinserimento *bad paths*		13061
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15033
	archi	totale	31265
		orientati	31258 (100 %)
		non orientati	7 (0 %)
30/03/2003 10:00			
AS path	totale		811001
	Bad paths	estratti	17016
		reinseriti	14239 (84 %)
Iterazioni	estrazione *bad paths*		293
	reinserimento *bad paths*		13417
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15036
	archi	totale	31273
		orientati	31266 (100 %)
		non orientati	7 (0 %)
30/03/2003 12:00			
AS path	totale		810989
	Bad paths	estratti	19375
		reinseriti	15930 (82 %)
Iterazioni	estrazione *bad paths*		290
	reinserimento *bad paths*		14426
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15039
	archi	totale	31265
		orientati	31259 (100 %)
		non orientati	6 (0 %)
30/03/2003 14:00			
AS path	totale		811181
	Bad paths	estratti	18912
		reinseriti	15405 (81 %)
Iterazioni	estrazione *bad paths*		295
	reinserimento *bad paths*		14504
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15044
	archi	totale	31292
		orientati	31284 (100 %)
		non orientati	8 (0 %)

Tabella 5.26: Caratteristiche del processo di inferenza

30/03/2003 16:00			
AS path	totale		811088
	Bad paths	estratti	17122
		reinseriti	14276 (83 %)
Iterazioni	estrazione *bad paths*		287
	reinserimento *bad paths*		13375
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15048
	archi	totale	31296
		orientati	31286 (100 %)
		non orientati	10 (0 %)
30/03/2003 18:00			
AS path	totale		811408
	Bad paths	estratti	19196
		reinseriti	15713 (82 %)
Iterazioni	estrazione *bad paths*		295
	reinserimento *bad paths*		14622
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15044
	archi	totale	31283
		orientati	31274 (100 %)
		non orientati	9 (0 %)
30/03/2003 20:00			
AS path	totale		812170
	Bad paths	estratti	17198
		reinseriti	14354 (83 %)
Iterazioni	estrazione *bad paths*		286
	reinserimento *bad paths*		13531
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15046
	archi	totale	31292
		orientati	31284 (100 %)
		non orientati	8 (0 %)
30/03/2003 22:00			
AS path	totale		811904
	Bad paths	estratti	16703
		reinseriti	13854 (83 %)
Iterazioni	estrazione *bad paths*		287
	reinserimento *bad paths*		13288
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15045
	archi	totale	31296
		orientati	31288 (100 %)
		non orientati	8 (0 %)

Tabella 5.27: Caratteristiche del processo di inferenza

31/03/2003 00:00			
AS path	totale		811900
	Bad paths	estratti	17073
		reinseriti	14330 (84 %)
Iterazioni	estrazione *bad paths*		288
	reinserimento *bad paths*		13390
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15046
	archi	totale	31288
		orientati	31277 (100 %)
		non orientati	11 (0 %)
31/03/2003 02:00			
AS path	totale		812573
	Bad paths	estratti	17339
		reinseriti	14401 (83 %)
Iterazioni	estrazione *bad paths*		296
	reinserimento *bad paths*		13503
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15046
	archi	totale	31291
		orientati	31281 (100 %)
		non orientati	10 (0 %)
31/03/2003 04:00			
AS path	totale		812417
	Bad paths	estratti	17293
		reinseriti	14413 (83 %)
Iterazioni	estrazione *bad paths*		290
	reinserimento *bad paths*		13779
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15048
	archi	totale	31299
		orientati	31287 (100 %)
		non orientati	12 (0 %)
31/03/2003 06:00			
AS path	totale		813961
	Bad paths	estratti	18721
		reinseriti	15147 (81 %)
Iterazioni	estrazione *bad paths*		297
	reinserimento *bad paths*		14689
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15073
	archi	totale	31314
		orientati	31290 (100 %)
		non orientati	24 (0 %)

Tabella 5.28: Caratteristiche del processo di inferenza

31/03/2003 08:00			
AS path	totale		813854
	Bad paths	estratti	19154
		reinseriti	16286 (85 %)
Iterazioni	estrazione *bad paths*		297
	reinserimento *bad paths*		14386
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15076
	archi	totale	31336
		orientati	31322 (100 %)
		non orientati	14 (0 %)
31/03/2003 10:00			
AS path	totale		814135
	Bad paths	estratti	18355
		reinseriti	15416 (84 %)
Iterazioni	estrazione *bad paths*		306
	reinserimento *bad paths*		14329
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15075
	archi	totale	31355
		orientati	31339 (100 %)
		non orientati	16 (0 %)
31/03/2003 12:00			
AS path	totale		813747
	Bad paths	estratti	18404
		reinseriti	15557 (85 %)
Iterazioni	estrazione *bad paths*		299
	reinserimento *bad paths*		14370
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15080
	archi	totale	31361
		orientati	31345 (100 %)
		non orientati	16 (0 %)
31/03/2003 14:00			
AS path	totale		812104
	Bad paths	estratti	18670
		reinseriti	15768 (84 %)
Iterazioni	estrazione *bad paths*		292
	reinserimento *bad paths*		14600
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15078
	archi	totale	31318
		orientati	31301 (100 %)
		non orientati	17 (0 %)

Tabella 5.29: Caratteristiche del processo di inferenza

31/03/2003 16:00			
AS path	totale		812119
	Bad paths	estratti	17164
		reinseriti	14308 (83 %)
Iterazioni	estrazione *bad paths*		290
	reinserimento *bad paths*		13940
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15081
	archi	totale	31336
		orientati	31318 (100 %)
		non orientati	18 (0 %)
31/03/2003 18:00			
AS path	totale		791304
	Bad paths	estratti	16905
		reinseriti	14291 (85 %)
Iterazioni	estrazione *bad paths*		282
	reinserimento *bad paths*		13360
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15073
	archi	totale	31193
		orientati	31178 (100 %)
		non orientati	15 (0 %)
31/03/2003 20:00			
AS path	totale		792356
	Bad paths	estratti	18000
		reinseriti	15354 (85 %)
Iterazioni	estrazione *bad paths*		285
	reinserimento *bad paths*		13632
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15076
	archi	totale	31193
		orientati	31178 (100 %)
		non orientati	15 (0 %)
31/03/2003 22:00			
AS path	totale		811608
	Bad paths	estratti	17688
		reinseriti	14932 (84 %)
Iterazioni	estrazione *bad paths*		298
	reinserimento *bad paths*		13956
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	15074
	archi	totale	31345
		orientati	31330 (100 %)
		non orientati	15 (0 %)

Tabella 5.30: Caratteristiche del processo di inferenza

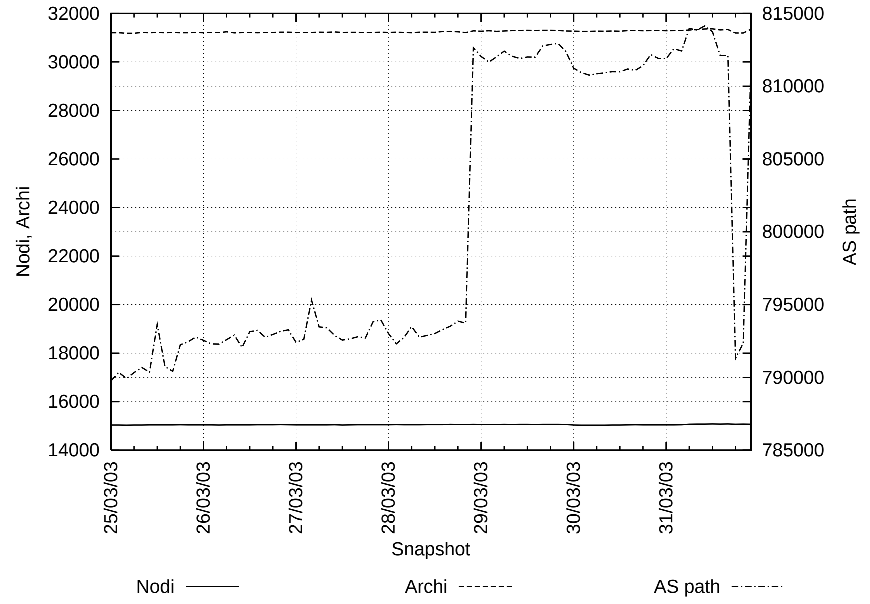

Figura 5.13: Numero di AS path e di nodi ed archi del grafo degli AS visibili da Route Views

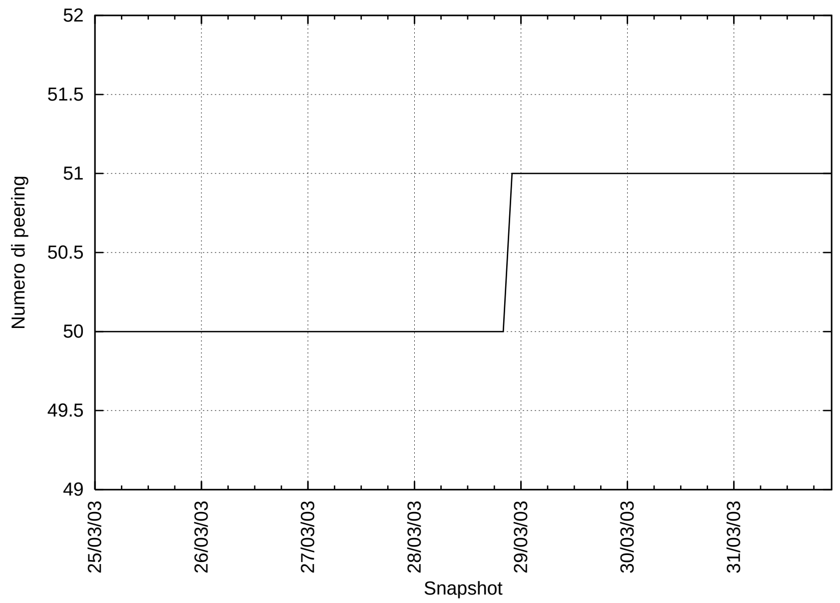

Figura 5.14: Numero di sessioni di peering attive tra Route Views ed altri router

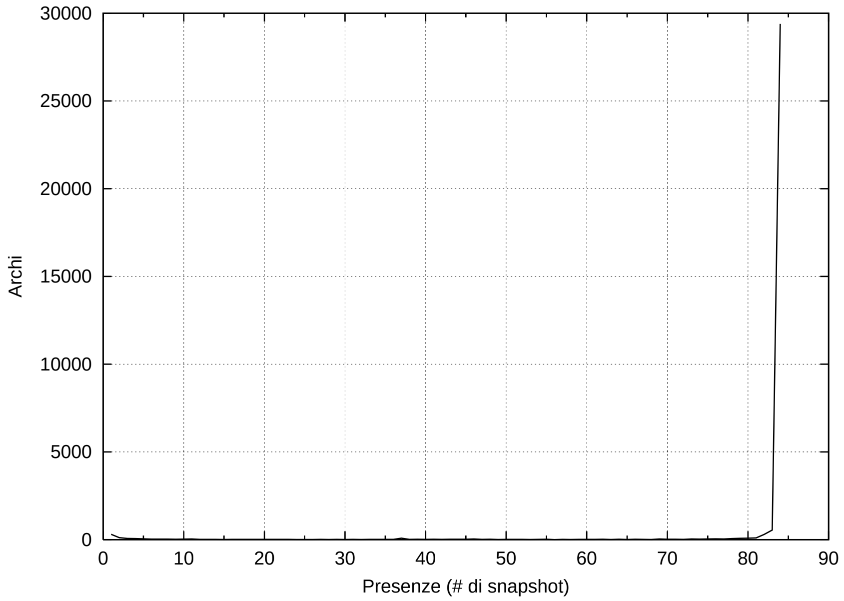

Figura 5.15: Distribuzione del numero di presenze di ciascun arco nei vari snapshot

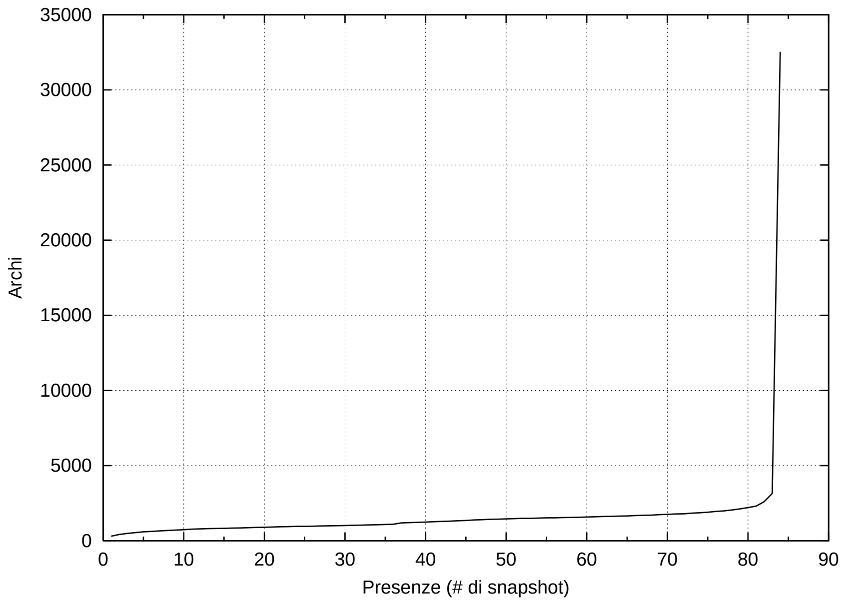

Figura 5.16: Distribuzione cumulativa del numero di presenze di ciascun arco nei vari snapshot (numero y di archi con al più x presenze)

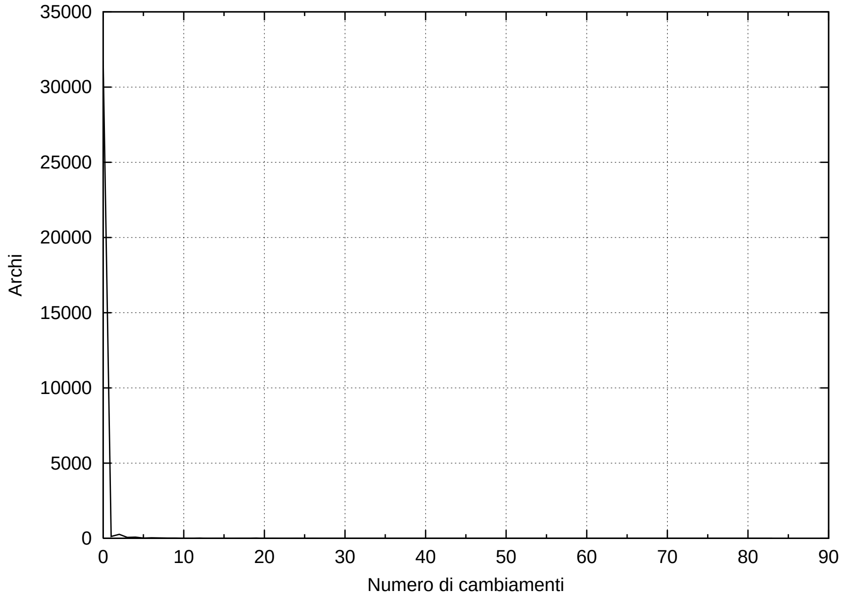

Figura 5.17: Distribuzione del numero di cambiamenti di orientazione che ciascun arco ha subito tra due snapshot consecutivi (inclusi i passaggi dallo stato orientato a quello non orientato e viceversa)

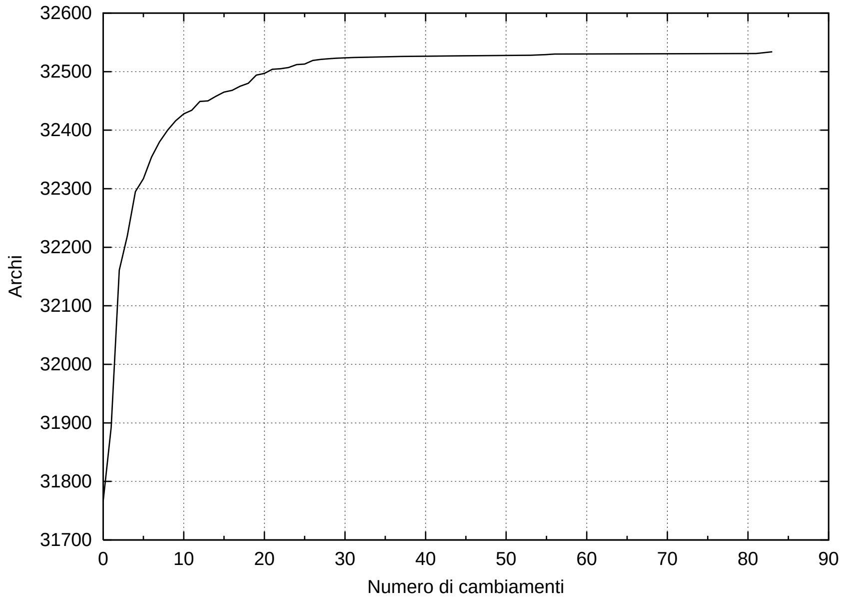

Figura 5.18: Distribuzione cumulativa del numero di cambiamenti di orientazione che ciascun arco ha subito tra due snapshot consecutivi (inclusi i passaggi dallo stato orientato a quello non orientato e viceversa)

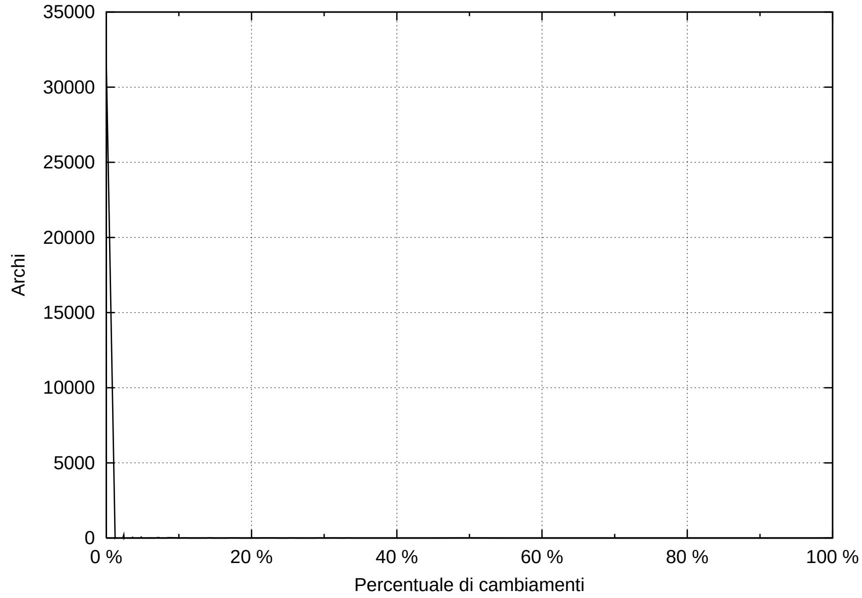

Figura 5.19: Distribuzione del rapporto tra i cambiamenti di orientazione che ciascun arco ha subito ed il numero delle sue presenze (ad esempio, gli archi riportati in corrispondenza del valore 100% hanno cambiato orientazione in ogni snapshot in cui sono stati presenti)

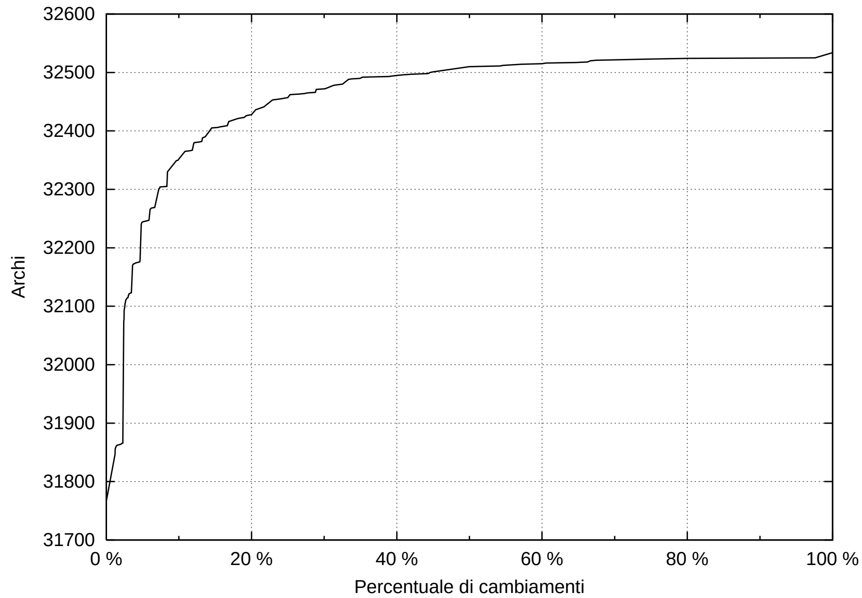

Figura 5.20: Distribuzione cumulativa del rapporto tra i cambiamenti di orientazione che ciascun arco ha subito ed il numero delle sue presenze

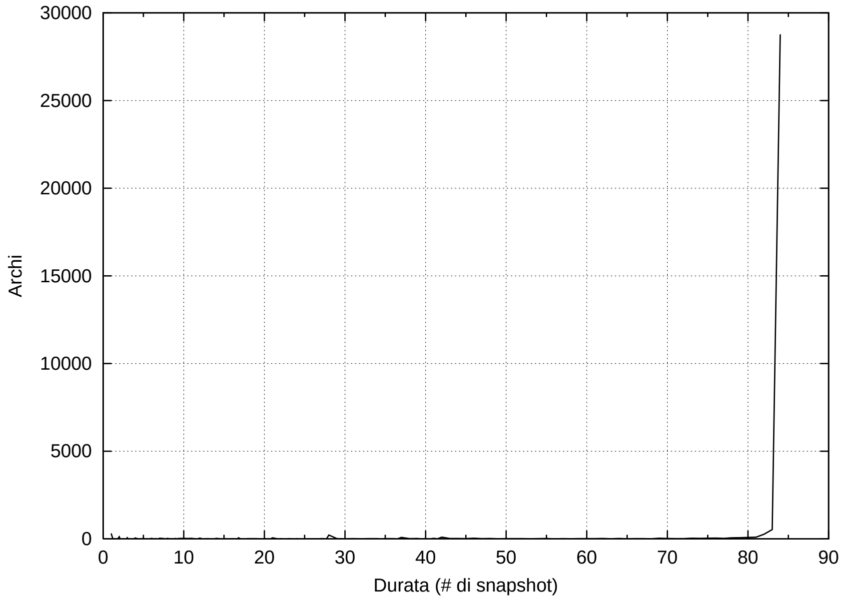

Figura 5.21: Distribuzione della durata media di un'orientazione (in termini di numero di snapshot consecutivi per cui viene mantenuta)

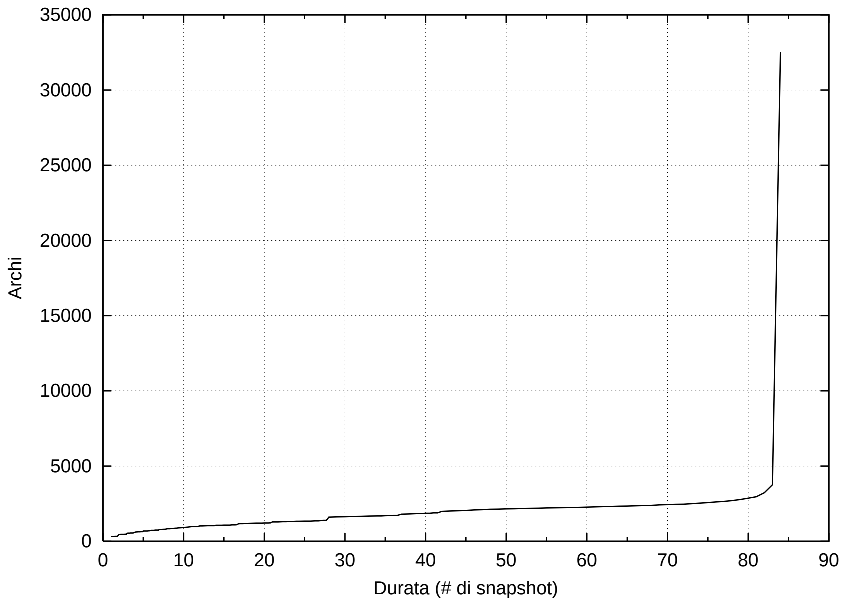

Figura 5.22: Distribuzione cumulativa della durata media di un'orientazione (in termini di numero di snapshot consecutivi per cui viene mantenuta)

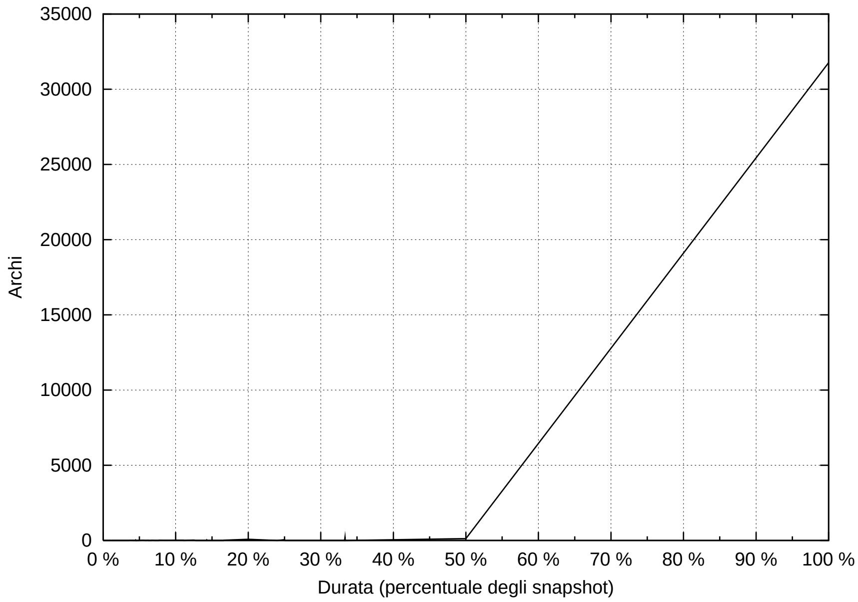

Figura 5.23: Distribuzione della durata media di un'orientazione (in termini di numero di snapshot consecutivi per cui viene mantenuta) rapportata al numero di presenze di ciascun arco (ad esempio, il valore in corrispondenza di 100% è il numero di archi che hanno mantenuto sempre stabile la propria orientazione)

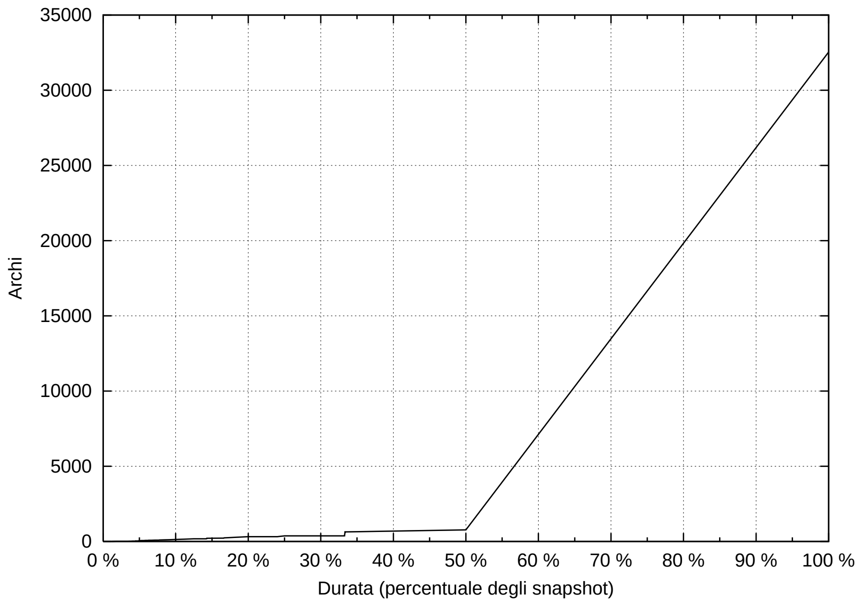

Figura 5.24: Distribuzione cumulativa della durata media di un'orientazione (in termini di numero di snapshot consecutivi per cui viene mantenuta) rapportata al numero di presenze di ciascun arco

5.2.4 Conclusioni

In questo caso la stabilità dell'orientazione ottenuta è ottima: più di 32000 archi (98% del totale), infatti, mantengono sempre inalterata la loro orientazione. Da notare è anche la maggior pulizia dei dati a disposizione: più dell'80% degli archi, infatti, risulta visibile da tutti gli snapshot. Inoltre, la variazione nel numero di peering attivi non sembra aver avuto particolare impatto sull'orientazione calcolata. In particolare, a cavallo dell'istante in cui essa ha luogo, gli archi che cambiano la propria orientazione oscillano tra i 45 ed i 58: evidentemente, una percentuale molto piccola anche rispetto al numero di archi visibili in prossimità di quell'istante (poco più di 15000).

Infine, è possibile notare come la computazione non tragga particolare giovamento dall'utilizzo della strategia di reinserzione descritta nel capitolo 4. In caso di successo, il numero di cammini candidati al reinserimento viene quadruplicato; in caso di fallimento, viene ridotto ad un quarto rispetto al passo precedente. In tutti gli snapshot utilizzati, il rapporto tra il numero di iterazioni necessarie per il reinserimento ed i cammini reinseriti si mantiene sempre molto vicino ad 1 (0.98), il che, evidentemente, non consente di trarre particolare beneficio da questa tecnica. Nel caso dei dati utilizzati nella sperimentazione precedente (in cui la strategia utilizzata per aggiornare il numero di cammini candidati al reinserimento era quella mostrata in figura 4.3), tale rapporto oscillava tra valori inferiori a 0.5 ed altri superiori ad 1, consentendo in particolari situazioni di ottenere un notevole risparmio computazionale. Potrebbe essere interessante cercare di valutare il criterio con cui tarare questo meccanismo per ottenere risultati ottimali anche con i dati di Route Views, ma in questa sede questo problema non viene affrontato.

5.3 Test su un periodo con numerosi eventi commerciali significativi

Per effettuare la terza sperimentazione si è deciso di prendere in considerazione un periodo, possibilmente ancora più prolungato di quello considerato in precedenza, durante il quale avesse avuto luogo un certo numero di cambiamenti commerciali significativi. Con questa prova si vuole cercare di valutare la qualità dell'orientazione calcolata per quanto riguarda la sua proprietà di riuscire a catturare queste variazioni.

5.3.1 Criteri utilizzati per la scelta dei dati

I dati utilizzati sono, come per la sperimentazione precedente, prelevati dall'archivio di Oregon Route Views ([22]). Per individuare un intervallo temporale abbastanza ricco di eventi ci si è basati ancora una volta sulle informazioni di [24, 25, 26]. L'arco temporale scelto si estende questa volta su due settimane, sia per effettuare un test ancora più massiccio sull'algoritmo di inferenza, sia per la relativa povertà di episodi commerciali nel corso di periodi più brevi. Le settimane scelte vanno dal 25 Settembre 2001 all'8 Ottobre 2001, e gli episodi più importanti che le hanno caratterizzate sono i seguenti:

- 25/09/2001: nuova partnership tra Oracle.com e BT Ignite

- 26/09/2001: formalizzazione della bancarotta di Exodus
- 28/09/2001: alleanza sui servizi di telecomunicazione tra Asia Global Crossing e China NetCom
- 01/10/2001: alleanza tra Worldcom e Sun
- 01/10/2001: alleanza tra Valtech e IBM
- 02/10/2001: alleanza tra Akamai e IBM
- 03/10/2001: consolidamento dell'alleanza tra Corio e Sun
- 05/10/2001: annuncio dell'alleanza tra IBM e Invensys
- 05/10/2001: licenziamento di 4000 impiegati o del 9% della forza lavoro della Sun Microsystems
- 08/10/2001: caduta di Exodus

Akamai	9413, 12222, 20940, 21342, 21357, 21399
BT Ignite	3624, 5400, 5556, 8472, 8903, 9060, 12339, 15697
China NetCom	7639, 9810, 9813, 9929, 10206, 17432, 17620, 17621, 17622, 17623, 17788, 17789, 17790, 17791
Corio	19048
Exodus	3967, 4197, 8709, 15870, 15878, 18993, 18997, 19179
Global Crossing	3549, 8259, 10032, 13207, 15774, 18116
IBM	706, 1747, 1997, 2687, 2688, 2693, 3775, 4672, 8871, 9460, 9955, 10120, 10337, 12169, 12980, 14904, 14912, 15006, 16081, 16807, 17657, 18206, 18703, 18713, 19057, 19093, 19152, 19603, 20375, 23257, 25440, 26062, 26416, 26543, 26607
Oracle	792, 793, 794, 1215
Sun	90, 6142, 11479
Worldcom	4183, 5586, 5621, 6066, 7407, 8768, 9194, 9922, 11486, 12651, 12702, 17373, 17605, 18061, 18573, 19973, 20951, 21253

Tabella 5.31: Autonomous Systems assegnati a ciascuna delle organizzazioni che ha subito cambiamenti tra il 25 Settembre 2001 e l'8 Ottobre 2001

Per la settimana scelta sono stati utilizzati 168 snapshot (uno ogni due ore). In seguito alla computazione, è stato rilevato che lo snapshot relativo al 3 Ottobre 2001 alle ore 14:00 è probabilmente corrotto: da esso è infatti possibile estrarre soltanto 2791 cammini.

5.3.2 Procedura seguita

Il procedimento seguito per effettuare le computazioni è identico a quello utilizzato nel caso precedente.

Il calcolo delle orientazioni è stato effettuato su tutti gli snapshot, senza escludere quello anomalo (03/10/2001 14:00).

5.3.3 Risultati ottenuti

Nonostante il processo di inferenza sia stato eseguito su tutti i dati a disposizione, in tutti i risultati qui presentati viene escluso lo snapshot del 3 Ottobre alle 14:00, per evitare che essi vengano perturbati da tale singolarità.

È possibile notare che il periodo scelto risulta piuttosto instabile sia dal punto di vista delle informazioni di routing che del numero di peering attivi.

Le varie distribuzioni riportate sono questa volta fornite sotto tre prospettive diverse, in modo tale da renderle anche comparabili con l'ampiezza dell'intervallo temporale utilizzata nella sperimentazione precedente. In particolare, per ciascuna grandezza viene mostrata sia la distribuzione relativa a quanto accaduto nell'intero arco temporale delle due settimane considerate, sia quella dei cambiamenti registrati nel corso di una sua sola metà. Tranne che nel caso delle distribuzioni percentuali, quest'ultima visione utilizza gli assi secondari (in alto e a destra), poiché riferita ad un diverso numero di snapshot ed a grafi con un diverso numero di archi.

25/09/2001 – 08/10/2001	
Numero di nodi	12317
Numero di archi	27490
25/09/2001 – 01/10/2001	
Numero di nodi	12206
Numero di archi	26679
02/10/2001 – 08/10/2001	
Numero di nodi	12184
Numero di archi	26463

Tabella 5.32: Caratteristiche del grafo unione

25/09/2001 00:00			
AS path	totale		549134
	Bad paths	estratti	11136
		reinseriti	8857 (80 %)
Iterazioni	estrazione *bad paths*		222
	reinserimento *bad paths*		9065
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	11948
	Archi	totale	25065
		orientati	25024 (100 %)
		non orientati	41 (0 %)

25/09/2001 02:00			
AS path	totale		548199
	Bad paths	estratti	11049
		reinseriti	8796 (80 %)
Iterazioni	estrazione *bad paths*		214
	reinserimento *bad paths*		8903
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	11944
	Archi	totale	25023
		orientati	24981 (100 %)
		non orientati	42 (0 %)

25/09/2001 04:00			
AS path	totale		549172
	Bad paths	estratti	11411
		reinseriti	9124 (80 %)
Iterazioni	estrazione *bad paths*		219
	reinserimento *bad paths*		9123
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	11952
	Archi	totale	25041
		orientati	25002 (100 %)
		non orientati	39 (0 %)

25/09/2001 06:00			
AS path	totale		548670
	Bad paths	estratti	10653
		reinseriti	8387 (79 %)
Iterazioni	estrazione *bad paths*		220
	reinserimento *bad paths*		8632
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	11950
	Archi	totale	25080
		orientati	25039 (100 %)
		non orientati	41 (0 %)

Tabella 5.33: Caratteristiche del processo di inferenza

25/09/2001 08:00			
AS path	totale		549352
	Bad paths	estratti	11280
		reinseriti	8952 (79 %)
Iterazioni	estrazione *bad paths*		225
	reinserimento *bad paths*		9136
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	11952
	Archi	totale	25082
		orientati	25040 (100 %)
		non orientati	42 (0 %)
25/09/2001 10:00			
AS path	totale		549388
	Bad paths	estratti	12357
		reinseriti	9991 (81 %)
Iterazioni	estrazione *bad paths*		233
	reinserimento *bad paths*		9700
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	11952
	Archi	totale	25091
		orientati	25052 (100 %)
		non orientati	39 (0 %)
25/09/2001 12:00			
AS path	totale		549295
	Bad paths	estratti	13370
		reinseriti	10880 (81 %)
Iterazioni	estrazione *bad paths*		219
	reinserimento *bad paths*		9356
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	11953
	Archi	totale	25093
		orientati	25057 (100 %)
		non orientati	36 (0 %)
25/09/2001 14:00			
AS path	totale		549246
	Bad paths	estratti	11588
		reinseriti	9286 (80 %)
Iterazioni	estrazione *bad paths*		223
	reinserimento *bad paths*		9274
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	11956
	Archi	totale	25082
		orientati	25046 (100 %)
		non orientati	36 (0 %)

Tabella 5.34: Caratteristiche del processo di inferenza

25/09/2001 16:00			
AS path	totale		549029
	Bad paths	estratti	11690
		reinseriti	9325 (80 %)
Iterazioni	estrazione *bad paths*		224
	reinserimento *bad paths*		9549
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	11953
	Archi	totale	25100
		orientati	25064 (100 %)
		non orientati	36 (0 %)
25/09/2001 18:00			
AS path	totale		549073
	Bad paths	estratti	12207
		reinseriti	9759 (80 %)
Iterazioni	estrazione *bad paths*		226
	reinserimento *bad paths*		9798
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	11957
	Archi	totale	25085
		orientati	25048 (100 %)
		non orientati	37 (0 %)
25/09/2001 20:00			
AS path	totale		548910
	Bad paths	estratti	14173
		reinseriti	11778 (83 %)
Iterazioni	estrazione *bad paths*		221
	reinserimento *bad paths*		9475
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	11951
	Archi	totale	25071
		orientati	25035 (100 %)
		non orientati	36 (0 %)
25/09/2001 22:00			
AS path	totale		549190
	Bad paths	estratti	14165
		reinseriti	11751 (83 %)
Iterazioni	estrazione *bad paths*		224
	reinserimento *bad paths*		9604
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	11950
	Archi	totale	25100
		orientati	25062 (100 %)
		non orientati	38 (0 %)

Tabella 5.35: Caratteristiche del processo di inferenza

26/09/2001 00:00			
AS path	totale		549304
	Bad paths	estratti	14058
		reinseriti	11471 (82 %)
Iterazioni	estrazione *bad paths*		224
	reinserimento *bad paths*		9874
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	11956
	Archi	totale	25087
		orientati	25051 (100 %)
		non orientati	36 (0 %)
26/09/2001 02:00			
AS path	totale		549078
	Bad paths	estratti	14302
		reinseriti	11651 (81 %)
Iterazioni	estrazione *bad paths*		234
	reinserimento *bad paths*		10280
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	11960
	Archi	totale	25097
		orientati	25055 (100 %)
		non orientati	42 (0 %)
26/09/2001 04:00			
AS path	totale		549401
	Bad paths	estratti	12255
		reinseriti	9673 (79 %)
Iterazioni	estrazione *bad paths*		217
	reinserimento *bad paths*		9969
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	11969
	Archi	totale	25104
		orientati	25069 (100 %)
		non orientati	35 (0 %)
26/09/2001 06:00			
AS path	totale		549456
	Bad paths	estratti	11970
		reinseriti	9521 (80 %)
Iterazioni	estrazione *bad paths*		222
	reinserimento *bad paths*		9857
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	11964
	Archi	totale	25101
		orientati	25062 (100 %)
		non orientati	39 (0 %)

Tabella 5.36: Caratteristiche del processo di inferenza

26/09/2001 08:00			
AS path	totale		507790
	Bad paths	estratti	20648
		reinseriti	17078 (83 %)
Iterazioni	estrazione *bad paths*		254
	reinserimento *bad paths*		14167
Rapporto iterazioni/path reinseriti			0.97
Grafo degli AS	nodi	totale	12026
	Archi	totale	25037
		orientati	24990 (100 %)
		non orientati	47 (0 %)

26/09/2001 10:00			
AS path	totale		553027
	Bad paths	estratti	15995
		reinseriti	12944 (81 %)
Iterazioni	estrazione *bad paths*		253
	reinserimento *bad paths*		11603
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12032
	Archi	totale	25180
		orientati	25145 (100 %)
		non orientati	35 (0 %)

26/09/2001 12:00			
AS path	totale		540875
	Bad paths	estratti	15616
		reinseriti	12410 (79 %)
Iterazioni	estrazione *bad paths*		246
	reinserimento *bad paths*		12067
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12037
	Archi	totale	25151
		orientati	25123 (100 %)
		non orientati	28 (0 %)

26/09/2001 14:00			
AS path	totale		538251
	Bad paths	estratti	14987
		reinseriti	12009 (80 %)
Iterazioni	estrazione *bad paths*		251
	reinserimento *bad paths*		11657
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12035
	Archi	totale	25139
		orientati	25100 (100 %)
		non orientati	39 (0 %)

Tabella 5.37: Caratteristiche del processo di inferenza

26/09/2001 16:00			
AS path	totale		535872
	Bad paths	estratti	14377
		reinseriti	11491 (80 %)
Iterazioni	estrazione *bad paths*		251
	reinserimento *bad paths*		11022
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12031
	Archi	totale	25126
		orientati	25092 (100 %)
		non orientati	34 (0 %)

26/09/2001 18:00			
AS path	totale		549195
	Bad paths	estratti	15769
		reinseriti	12610 (80 %)
Iterazioni	estrazione *bad paths*		256
	reinserimento *bad paths*		11640
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12033
	Archi	totale	25176
		orientati	25145 (100 %)
		non orientati	31 (0 %)

26/09/2001 20:00			
AS path	totale		535040
	Bad paths	estratti	17186
		reinseriti	14286 (83 %)
Iterazioni	estrazione *bad paths*		246
	reinserimento *bad paths*		11550
Rapporto iterazioni/path reinseriti			0.97
Grafo degli AS	nodi	totale	12032
	Archi	totale	25137
		orientati	25102 (100 %)
		non orientati	35 (0 %)

26/09/2001 22:00			
AS path	totale		533830
	Bad paths	estratti	14262
		reinseriti	11302 (79 %)
Iterazioni	estrazione *bad paths*		250
	reinserimento *bad paths*		11249
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12028
	Archi	totale	25148
		orientati	25114 (100 %)
		non orientati	34 (0 %)

Tabella 5.38: Caratteristiche del processo di inferenza

27/09/2001 00:00			
AS path	totale		527464
	Bad paths	estratti	14424
		reinseriti	11433 (79 %)
Iterazioni	estrazione *bad paths*		252
	reinserimento *bad paths*		10927
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12026
	Archi	totale	25126
		orientati	25085 (100 %)
		non orientati	41 (0 %)
27/09/2001 02:00			
AS path	totale		537440
	Bad paths	estratti	14016
		reinseriti	11197 (80 %)
Iterazioni	estrazione *bad paths*		245
	reinserimento *bad paths*		10567
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12029
	Archi	totale	25130
		orientati	25094 (100 %)
		non orientati	36 (0 %)
27/09/2001 04:00			
AS path	totale		538837
	Bad paths	estratti	15324
		reinseriti	12278 (80 %)
Iterazioni	estrazione *bad paths*		244
	reinserimento *bad paths*		11104
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12049
	Archi	totale	25177
		orientati	25146 (100 %)
		non orientati	31 (0 %)
27/09/2001 06:00			
AS path	totale		538495
	Bad paths	estratti	15762
		reinseriti	12762 (81 %)
Iterazioni	estrazione *bad paths*		251
	reinserimento *bad paths*		11542
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12052
	Archi	totale	25170
		orientati	25143 (100 %)
		non orientati	27 (0 %)

Tabella 5.39: Caratteristiche del processo di inferenza

27/09/2001 08:00			
AS path	totale		553306
	Bad paths	estratti	15587
		reinseriti	12365 (79 %)
Iterazioni	estrazione *bad paths*		249
	reinserimento *bad paths*		12061
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12056
	Archi	totale	25193
		orientati	25166 (100 %)
		non orientati	27 (0 %)

27/09/2001 10:00			
AS path	totale		553350
	Bad paths	estratti	16388
		reinseriti	13422 (82 %)
Iterazioni	estrazione *bad paths*		260
	reinserimento *bad paths*		12089
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12058
	Archi	totale	25194
		orientati	25164 (100 %)
		non orientati	30 (0 %)

27/09/2001 12:00			
AS path	totale		553136
	Bad paths	estratti	16418
		reinseriti	13451 (82 %)
Iterazioni	estrazione *bad paths*		259
	reinserimento *bad paths*		12220
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12062
	Archi	totale	25196
		orientati	25166 (100 %)
		non orientati	30 (0 %)

27/09/2001 14:00			
AS path	totale		553225
	Bad paths	estratti	14869
		reinseriti	11943 (80 %)
Iterazioni	estrazione *bad paths*		259
	reinserimento *bad paths*		11621
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12062
	Archi	totale	25191
		orientati	25153 (100 %)
		non orientati	38 (0 %)

Tabella 5.40: Caratteristiche del processo di inferenza

27/09/2001 16:00			
AS path	totale		553104
	Bad paths	estratti	14026
		reinseriti	11219 (80 %)
Iterazioni	estrazione *bad paths*		239
	reinserimento *bad paths*		10906
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12034
	Archi	totale	25169
		orientati	25131 (100 %)
		non orientati	38 (0 %)
27/09/2001 18:00			
AS path	totale		538794
	Bad paths	estratti	16078
		reinseriti	12912 (80 %)
Iterazioni	estrazione *bad paths*		259
	reinserimento *bad paths*		11955
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12034
	Archi	totale	25144
		orientati	25100 (100 %)
		non orientati	44 (0 %)
27/09/2001 20:00			
AS path	totale		537150
	Bad paths	estratti	18010
		reinseriti	14973 (83 %)
Iterazioni	estrazione *bad paths*		253
	reinserimento *bad paths*		11963
Rapporto iterazioni/path reinseriti			0.97
Grafo degli AS	nodi	totale	12032
	Archi	totale	25147
		orientati	25111 (100 %)
		non orientati	36 (0 %)
27/09/2001 22:00			
AS path	totale		535942
	Bad paths	estratti	14781
		reinseriti	11735 (79 %)
Iterazioni	estrazione *bad paths*		244
	reinserimento *bad paths*		11140
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12028
	Archi	totale	25158
		orientati	25121 (100 %)
		non orientati	37 (0 %)

Tabella 5.41: Caratteristiche del processo di inferenza

28/09/2001 00:00			
AS path	totale		537616
	Bad paths	estratti	14182
		reinseriti	11338 (80 %)
Iterazioni	estrazione *bad paths*		249
	reinserimento *bad paths*		11234
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12036
	Archi	totale	25178
		orientati	25140 (100 %)
		non orientati	38 (0 %)
28/09/2001 02:00			
AS path	totale		536911
	Bad paths	estratti	14181
		reinseriti	11369 (80 %)
Iterazioni	estrazione *bad paths*		241
	reinserimento *bad paths*		11205
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12032
	Archi	totale	25171
		orientati	25137 (100 %)
		non orientati	34 (0 %)
28/09/2001 04:00			
AS path	totale		536722
	Bad paths	estratti	14304
		reinseriti	11599 (81 %)
Iterazioni	estrazione *bad paths*		246
	reinserimento *bad paths*		10998
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12036
	Archi	totale	25146
		orientati	25109 (100 %)
		non orientati	37 (0 %)
28/09/2001 06:00			
AS path	totale		537483
	Bad paths	estratti	14871
		reinseriti	12220 (82 %)
Iterazioni	estrazione *bad paths*		244
	reinserimento *bad paths*		11143
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12037
	Archi	totale	25174
		orientati	25145 (100 %)
		non orientati	29 (0 %)

Tabella 5.42: Caratteristiche del processo di inferenza

28/09/2001 08:00			
AS path	totale		537520
	Bad paths	estratti	15287
		reinseriti	12606 (82 %)
Iterazioni	estrazione *bad paths*		250
	reinserimento *bad paths*		11261
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12040
	Archi	totale	25180
		orientati	25142 (100 %)
		non orientati	38 (0 %)
28/09/2001 10:00			
AS path	totale		552791
	Bad paths	estratti	15292
		reinseriti	12372 (81 %)
Iterazioni	estrazione *bad paths*		247
	reinserimento *bad paths*		11644
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12037
	Archi	totale	25135
		orientati	25104 (100 %)
		non orientati	31 (0 %)
28/09/2001 12:00			
AS path	totale		551062
	Bad paths	estratti	15764
		reinseriti	12512 (79 %)
Iterazioni	estrazione *bad paths*		249
	reinserimento *bad paths*		11731
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12037
	Archi	totale	25130
		orientati	25090 (100 %)
		non orientati	40 (0 %)
28/09/2001 14:00			
AS path	totale		535834
	Bad paths	estratti	14188
		reinseriti	11384 (80 %)
Iterazioni	estrazione *bad paths*		237
	reinserimento *bad paths*		10966
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12042
	Archi	totale	25109
		orientati	25079 (100 %)
		non orientati	30 (0 %)

Tabella 5.43: Caratteristiche del processo di inferenza

28/09/2001 16:00			
AS path	totale		550498
	Bad paths	estratti	16423
		reinseriti	13279 (81 %)
Iterazioni	estrazione *bad paths*		246
	reinserimento *bad paths*		11732
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12034
	Archi	totale	25142
		orientati	25116 (100 %)
		non orientati	26 (0 %)
28/09/2001 18:00			
AS path	totale		550128
	Bad paths	estratti	13445
		reinseriti	10798 (80 %)
Iterazioni	estrazione *bad paths*		232
	reinserimento *bad paths*		10349
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12032
	Archi	totale	25127
		orientati	25097 (100 %)
		non orientati	30 (0 %)
28/09/2001 20:00			
AS path	totale		550151
	Bad paths	estratti	15354
		reinseriti	12227 (80 %)
Iterazioni	estrazione *bad paths*		243
	reinserimento *bad paths*		11587
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12029
	Archi	totale	25124
		orientati	25084 (100 %)
		non orientati	40 (0 %)
28/09/2001 22:00			
AS path	totale		549945
	Bad paths	estratti	17173
		reinseriti	14061 (82 %)
Iterazioni	estrazione *bad paths*		248
	reinserimento *bad paths*		11843
Rapporto iterazioni/path reinseriti			0.97
Grafo degli AS	nodi	totale	12027
	Archi	totale	25123
		orientati	25094 (100 %)
		non orientati	29 (0 %)

Tabella 5.44: Caratteristiche del processo di inferenza

29/09/2001 00:00			
AS path	totale		547649
	Bad paths	estratti	15593
		reinseriti	12753 (82 %)
Iterazioni	estrazione *bad paths*		255
	reinserimento *bad paths*		11576
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12030
	Archi	totale	25132
		orientati	25101 (100 %)
		non orientati	31 (0 %)
29/09/2001 02:00			
AS path	totale		550236
	Bad paths	estratti	15155
		reinseriti	12381 (82 %)
Iterazioni	estrazione *bad paths*		250
	reinserimento *bad paths*		11617
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12029
	Archi	totale	25131
		orientati	25098 (100 %)
		non orientati	33 (0 %)
29/09/2001 04:00			
AS path	totale		550340
	Bad paths	estratti	16601
		reinseriti	13541 (82 %)
Iterazioni	estrazione *bad paths*		247
	reinserimento *bad paths*		11744
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12029
	Archi	totale	25130
		orientati	25101 (100 %)
		non orientati	29 (0 %)
29/09/2001 06:00			
AS path	totale		550267
	Bad paths	estratti	14842
		reinseriti	11538 (78 %)
Iterazioni	estrazione *bad paths*		248
	reinserimento *bad paths*		12046
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12027
	Archi	totale	25135
		orientati	25094 (100 %)
		non orientati	41 (0 %)

Tabella 5.45: Caratteristiche del processo di inferenza

29/09/2001 08:00			
AS path	totale		550486
	Bad paths	estratti	14499
		reinseriti	11708 (81 %)
Iterazioni	estrazione *bad paths*		250
	reinserimento *bad paths*		11521
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12030
	Archi	totale	25148
		orientati	25114 (100 %)
		non orientati	34 (0 %)
29/09/2001 10:00			
AS path	totale		550367
	Bad paths	estratti	14666
		reinseriti	11901 (81 %)
Iterazioni	estrazione *bad paths*		245
	reinserimento *bad paths*		11347
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12030
	Archi	totale	25140
		orientati	25108 (100 %)
		non orientati	32 (0 %)
29/09/2001 12:00			
AS path	totale		550455
	Bad paths	estratti	15077
		reinseriti	12129 (80 %)
Iterazioni	estrazione *bad paths*		241
	reinserimento *bad paths*		11739
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12039
	Archi	totale	25145
		orientati	25114 (100 %)
		non orientati	31 (0 %)
29/09/2001 14:00			
AS path	totale		550807
	Bad paths	estratti	14975
		reinseriti	12205 (82 %)
Iterazioni	estrazione *bad paths*		242
	reinserimento *bad paths*		11584
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12032
	Archi	totale	25137
		orientati	25106 (100 %)
		non orientati	31 (0 %)

Tabella 5.46: Caratteristiche del processo di inferenza

29/09/2001 16:00			
AS path	totale		550673
	Bad paths	estratti	14418
		reinseriti	11690 (81 %)
Iterazioni	estrazione *bad paths*		242
	reinserimento *bad paths*		11200
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12031
	Archi	totale	25141
		orientati	25109 (100 %)
		non orientati	32 (0 %)
29/09/2001 18:00			
AS path	totale		550656
	Bad paths	estratti	16634
		reinseriti	13413 (81 %)
Iterazioni	estrazione *bad paths*		243
	reinserimento *bad paths*		11924
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12030
	Archi	totale	25139
		orientati	25100 (100 %)
		non orientati	39 (0 %)
29/09/2001 20:00			
AS path	totale		550494
	Bad paths	estratti	14881
		reinseriti	12040 (81 %)
Iterazioni	estrazione *bad paths*		239
	reinserimento *bad paths*		11595
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12031
	Archi	totale	25131
		orientati	25101 (100 %)
		non orientati	30 (0 %)
29/09/2001 22:00			
AS path	totale		545318
	Bad paths	estratti	14818
		reinseriti	12086 (82 %)
Iterazioni	estrazione *bad paths*		245
	reinserimento *bad paths*		11054
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12026
	Archi	totale	25115
		orientati	25082 (100 %)
		non orientati	33 (0 %)

Tabella 5.47: Caratteristiche del processo di inferenza

30/09/2001 00:00			
AS path	totale		549472
	Bad paths	estratti	14770
		reinseriti	11847 (80 %)
Iterazioni	estrazione *bad paths*		248
	reinserimento *bad paths*		11558
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12022
	Archi	totale	25108
		orientati	25073 (100 %)
		non orientati	35 (0 %)
30/09/2001 02:00			
AS path	totale		549621
	Bad paths	estratti	15645
		reinseriti	12735 (81 %)
Iterazioni	estrazione *bad paths*		252
	reinserimento *bad paths*		12273
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12027
	Archi	totale	25083
		orientati	25051 (100 %)
		non orientati	32 (0 %)
30/09/2001 04:00			
AS path	totale		549632
	Bad paths	estratti	17510
		reinseriti	13813 (79 %)
Iterazioni	estrazione *bad paths*		257
	reinserimento *bad paths*		12781
Rapporto iterazioni/path reinseriti			0.97
Grafo degli AS	nodi	totale	12033
	Archi	totale	25098
		orientati	25059 (100 %)
		non orientati	39 (0 %)
30/09/2001 06:00			
AS path	totale		549747
	Bad paths	estratti	15179
		reinseriti	11952 (79 %)
Iterazioni	estrazione *bad paths*		244
	reinserimento *bad paths*		11693
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12035
	Archi	totale	25096
		orientati	25055 (100 %)
		non orientati	41 (0 %)

Tabella 5.48: Caratteristiche del processo di inferenza

30/09/2001 08:00			
AS path	totale		549497
	Bad paths	estratti	14014
		reinseriti	11283 (81 %)
Iterazioni	estrazione *bad paths*		240
	reinserimento *bad paths*		10935
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12031
	Archi	totale	25086
		orientati	25054 (100 %)
		non orientati	32 (0 %)
30/09/2001 10:00			
AS path	totale		549478
	Bad paths	estratti	13637
		reinseriti	10954 (80 %)
Iterazioni	estrazione *bad paths*		239
	reinserimento *bad paths*		10720
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12032
	Archi	totale	25101
		orientati	25067 (100 %)
		non orientati	34 (0 %)
30/09/2001 12:00			
AS path	totale		549685
	Bad paths	estratti	14185
		reinseriti	11466 (81 %)
Iterazioni	estrazione *bad paths*		244
	reinserimento *bad paths*		10923
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12033
	Archi	totale	25093
		orientati	25061 (100 %)
		non orientati	32 (0 %)
30/09/2001 14:00			
AS path	totale		549678
	Bad paths	estratti	14243
		reinseriti	11572 (81 %)
Iterazioni	estrazione *bad paths*		234
	reinserimento *bad paths*		10985
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12037
	Archi	totale	25108
		orientati	25074 (100 %)
		non orientati	34 (0 %)

Tabella 5.49: Caratteristiche del processo di inferenza

30/09/2001 16:00			
AS path	totale		549600
	Bad paths	estratti	13915
		reinseriti	11256 (81 %)
Iterazioni	estrazione *bad paths*		237
	reinserimento *bad paths*		10692
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12036
	Archi	totale	25099
		orientati	25065 (100 %)
		non orientati	34 (0 %)
30/09/2001 18:00			
AS path	totale		550240
	Bad paths	estratti	14673
		reinseriti	11970 (82 %)
Iterazioni	estrazione *bad paths*		241
	reinserimento *bad paths*		11099
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12040
	Archi	totale	25096
		orientati	25064 (100 %)
		non orientati	32 (0 %)
30/09/2001 20:00			
AS path	totale		547487
	Bad paths	estratti	14453
		reinseriti	11751 (81 %)
Iterazioni	estrazione *bad paths*		245
	reinserimento *bad paths*		11154
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12039
	Archi	totale	25087
		orientati	25054 (100 %)
		non orientati	33 (0 %)
30/09/2001 22:00			
AS path	totale		535219
	Bad paths	estratti	14568
		reinseriti	11437 (79 %)
Iterazioni	estrazione *bad paths*		246
	reinserimento *bad paths*		11134
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12040
	Archi	totale	25077
		orientati	25034 (100 %)
		non orientati	43 (0 %)

Tabella 5.50: Caratteristiche del processo di inferenza

01/10/2001 00:00			
AS path	totale		550594
	Bad paths	estratti	14667
		reinseriti	11938 (81 %)
Iterazioni	estrazione *bad paths*		251
	reinserimento *bad paths*		11288
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12039
	Archi	totale	25111
		orientati	25075 (100 %)
		non orientati	36 (0 %)

01/10/2001 02:00			
AS path	totale		551288
	Bad paths	estratti	16998
		reinseriti	13804 (81 %)
Iterazioni	estrazione *bad paths*		253
	reinserimento *bad paths*		12430
Rapporto iterazioni/path reinseriti			0.97
Grafo degli AS	nodi	totale	12044
	Archi	totale	25130
		orientati	25094 (100 %)
		non orientati	36 (0 %)

01/10/2001 04:00			
AS path	totale		550800
	Bad paths	estratti	16149
		reinseriti	12958 (80 %)
Iterazioni	estrazione *bad paths*		252
	reinserimento *bad paths*		11887
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12044
	Archi	totale	25114
		orientati	25076 (100 %)
		non orientati	38 (0 %)

01/10/2001 06:00			
AS path	totale		550866
	Bad paths	estratti	16573
		reinseriti	13351 (81 %)
Iterazioni	estrazione *bad paths*		253
	reinserimento *bad paths*		12254
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12043
	Archi	totale	25122
		orientati	25083 (100 %)
		non orientati	39 (0 %)

Tabella 5.51: Caratteristiche del processo di inferenza

01/10/2001 08:00			
AS path	totale		551851
	Bad paths	estratti	13994
		reinseriti	11075 (79 %)
Iterazioni	estrazione *bad paths*		249
	reinserimento *bad paths*		11323
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12067
	Archi	totale	25156
		orientati	25116 (100 %)
		non orientati	40 (0 %)

01/10/2001 10:00			
AS path	totale		550942
	Bad paths	estratti	14449
		reinseriti	11678 (81 %)
Iterazioni	estrazione *bad paths*		250
	reinserimento *bad paths*		11399
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12042
	Archi	totale	25101
		orientati	25064 (100 %)
		non orientati	37 (0 %)

01/10/2001 12:00			
AS path	totale		535965
	Bad paths	estratti	14360
		reinseriti	11397 (79 %)
Iterazioni	estrazione *bad paths*		250
	reinserimento *bad paths*		11405
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12038
	Archi	totale	25070
		orientati	25033 (100 %)
		non orientati	37 (0 %)

01/10/2001 14:00			
AS path	totale		535813
	Bad paths	estratti	17068
		reinseriti	13762 (81 %)
Iterazioni	estrazione *bad paths*		251
	reinserimento *bad paths*		12366
Rapporto iterazioni/path reinseriti			0.97
Grafo degli AS	nodi	totale	12035
	Archi	totale	25053
		orientati	25017 (100 %)
		non orientati	36 (0 %)

Tabella 5.52: Caratteristiche del processo di inferenza

01/10/2001 16:00			
AS path	totale		550960
	Bad paths	estratti	13978
		reinseriti	11198 (80 %)
Iterazioni	estrazione *bad paths*		250
	reinserimento *bad paths*		11096
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12040
	Archi	totale	25086
		orientati	25052 (100 %)
		non orientati	34 (0 %)
01/10/2001 18:00			
AS path	totale		550870
	Bad paths	estratti	14319
		reinseriti	11442 (80 %)
Iterazioni	estrazione *bad paths*		254
	reinserimento *bad paths*		11255
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12038
	Archi	totale	25089
		orientati	25048 (100 %)
		non orientati	41 (0 %)
01/10/2001 20:00			
AS path	totale		550295
	Bad paths	estratti	16587
		reinseriti	13395 (81 %)
Iterazioni	estrazione *bad paths*		253
	reinserimento *bad paths*		11878
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12037
	Archi	totale	25086
		orientati	25048 (100 %)
		non orientati	38 (0 %)
01/10/2001 22:00			
AS path	totale		550850
	Bad paths	estratti	16845
		reinseriti	13457 (80 %)
Iterazioni	estrazione *bad paths*		254
	reinserimento *bad paths*		12144
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12039
	Archi	totale	25077
		orientati	25043 (100 %)
		non orientati	34 (0 %)

Tabella 5.53: Caratteristiche del processo di inferenza

02/10/2001 00:00			
AS path	totale		552420
	Bad paths	estratti	13915
		reinseriti	10962 (79 %)
Iterazioni	estrazione *bad paths*		246
	reinserimento *bad paths*		11495
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12046
	Archi	totale	25095
		orientati	25063 (100 %)
		non orientati	32 (0 %)

02/10/2001 02:00			
AS path	totale		551008
	Bad paths	estratti	14136
		reinseriti	11339 (80 %)
Iterazioni	estrazione *bad paths*		242
	reinserimento *bad paths*		10850
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12043
	Archi	totale	25076
		orientati	25034 (100 %)
		non orientati	42 (0 %)

02/10/2001 04:00			
AS path	totale		550424
	Bad paths	estratti	14495
		reinseriti	11780 (81 %)
Iterazioni	estrazione *bad paths*		247
	reinserimento *bad paths*		10846
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12042
	Archi	totale	25056
		orientati	25016 (100 %)
		non orientati	40 (0 %)

02/10/2001 06:00			
AS path	totale		551335
	Bad paths	estratti	14012
		reinseriti	11294 (81 %)
Iterazioni	estrazione *bad paths*		237
	reinserimento *bad paths*		10676
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12044
	Archi	totale	25035
		orientati	25001 (100 %)
		non orientati	34 (0 %)

Tabella 5.54: Caratteristiche del processo di inferenza

02/10/2001 08:00			
AS path	totale		551366
	Bad paths	estratti	14113
		reinseriti	11430 (81 %)
Iterazioni	estrazione *bad paths*		235
	reinserimento *bad paths*		10669
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12043
	Archi	totale	25055
		orientati	25019 (100 %)
		non orientati	36 (0 %)

02/10/2001 10:00			
AS path	totale		552356
	Bad paths	estratti	13270
		reinseriti	10489 (79 %)
Iterazioni	estrazione *bad paths*		218
	reinserimento *bad paths*		10688
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12044
	Archi	totale	25070
		orientati	25037 (100 %)
		non orientati	33 (0 %)

02/10/2001 12:00			
AS path	totale		551332
	Bad paths	estratti	13964
		reinseriti	10996 (79 %)
Iterazioni	estrazione *bad paths*		233
	reinserimento *bad paths*		10939
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12034
	Archi	totale	25060
		orientati	25025 (100 %)
		non orientati	35 (0 %)

02/10/2001 14:00			
AS path	totale		550479
	Bad paths	estratti	13612
		reinseriti	10848 (80 %)
Iterazioni	estrazione *bad paths*		237
	reinserimento *bad paths*		10203
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12037
	Archi	totale	24985
		orientati	24951 (100 %)
		non orientati	34 (0 %)

Tabella 5.55: Caratteristiche del processo di inferenza

02/10/2001 16:00			
AS path	totale		550614
	Bad paths	estratti	14008
		reinseriti	11469 (82 %)
Iterazioni	estrazione *bad paths*		241
	reinserimento *bad paths*		10344
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12037
	Archi	totale	24995
		orientati	24958 (100 %)
		non orientati	37 (0 %)

02/10/2001 18:00			
AS path	totale		550943
	Bad paths	estratti	13644
		reinseriti	10832 (79 %)
Iterazioni	estrazione *bad paths*		235
	reinserimento *bad paths*		10421
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12052
	Archi	totale	25012
		orientati	24980 (100 %)
		non orientati	32 (0 %)

02/10/2001 20:00			
AS path	totale		536726
	Bad paths	estratti	13546
		reinseriti	10771 (80 %)
Iterazioni	estrazione *bad paths*		234
	reinserimento *bad paths*		10240
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12054
	Archi	totale	24999
		orientati	24967 (100 %)
		non orientati	32 (0 %)

02/10/2001 22:00			
AS path	totale		547220
	Bad paths	estratti	13784
		reinseriti	11075 (80 %)
Iterazioni	estrazione *bad paths*		243
	reinserimento *bad paths*		10290
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12055
	Archi	totale	25030
		orientati	25009 (100 %)
		non orientati	21 (0 %)

Tabella 5.56: Caratteristiche del processo di inferenza

03/10/2001 00:00			
AS path	totale		551254
	Bad paths	estratti	14768
		reinseriti	11860 (80 %)
Iterazioni	estrazione *bad paths*		236
	reinserimento *bad paths*		10724
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12058
	Archi	totale	25046
		orientati	25014 (100 %)
		non orientati	32 (0 %)
03/10/2001 02:00			
AS path	totale		550960
	Bad paths	estratti	15387
		reinseriti	12372 (80 %)
Iterazioni	estrazione *bad paths*		246
	reinserimento *bad paths*		10698
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12054
	Archi	totale	25049
		orientati	25029 (100 %)
		non orientati	20 (0 %)
03/10/2001 04:00			
AS path	totale		551325
	Bad paths	estratti	13820
		reinseriti	11016 (80 %)
Iterazioni	estrazione *bad paths*		249
	reinserimento *bad paths*		10462
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12054
	Archi	totale	25033
		orientati	24997 (100 %)
		non orientati	36 (0 %)
03/10/2001 06:00			
AS path	totale		551224
	Bad paths	estratti	13631
		reinseriti	10900 (80 %)
Iterazioni	estrazione *bad paths*		242
	reinserimento *bad paths*		10677
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12058
	Archi	totale	25036
		orientati	25001 (100 %)
		non orientati	35 (0 %)

Tabella 5.57: Caratteristiche del processo di inferenza

03/10/2001 08:00			
AS path	totale		551468
	Bad paths	estratti	14793
		reinseriti	11806 (80 %)
Iterazioni	estrazione *bad paths*		245
	reinserimento *bad paths*		11139
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12058
	Archi	totale	25049
		orientati	25012 (100 %)
		non orientati	37 (0 %)
03/10/2001 10:00			
AS path	totale		552880
	Bad paths	estratti	14588
		reinseriti	11939 (82 %)
Iterazioni	estrazione *bad paths*		247
	reinserimento *bad paths*		11119
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12056
	Archi	totale	25031
		orientati	24996 (100 %)
		non orientati	35 (0 %)
03/10/2001 12:00			
AS path	totale		536201
	Bad paths	estratti	14212
		reinseriti	11602 (82 %)
Iterazioni	estrazione *bad paths*		252
	reinserimento *bad paths*		10514
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12057
	Archi	totale	24987
		orientati	24953 (100 %)
		non orientati	34 (0 %)
03/10/2001 14:00			
AS path	totale		2719
	Bad paths	estratti	N/D
		reinseriti	N/D
Iterazioni	estrazione *bad paths*		N/D
	reinserimento *bad paths*		N/D
Rapporto iterazioni/path reinseriti			N/D
Grafo degli AS	nodi	totale	N/D
	Archi	totale	N/D
		orientati	N/D
		non orientati	N/D

Tabella 5.58: Caratteristiche del processo di inferenza

03/10/2001 16:00			
AS path	totale		536874
	Bad paths	estratti	16151
		reinseriti	13150 (81 %)
Iterazioni	estrazione *bad paths*		245
	reinserimento *bad paths*		11501
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12060
	Archi	totale	24979
		orientati	24954 (100 %)
		non orientati	25 (0 %)

03/10/2001 18:00			
AS path	totale		537255
	Bad paths	estratti	19745
		reinseriti	16954 (86 %)
Iterazioni	estrazione *bad paths*		242
	reinserimento *bad paths*		12775
Rapporto iterazioni/path reinseriti			0.97
Grafo degli AS	nodi	totale	12056
	Archi	totale	24974
		orientati	24938 (100 %)
		non orientati	36 (0 %)

03/10/2001 20:00			
AS path	totale		537299
	Bad paths	estratti	15706
		reinseriti	12690 (81 %)
Iterazioni	estrazione *bad paths*		243
	reinserimento *bad paths*		11434
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12058
	Archi	totale	24978
		orientati	24947 (100 %)
		non orientati	31 (0 %)

03/10/2001 22:00			
AS path	totale		537223
	Bad paths	estratti	13923
		reinseriti	11062 (79 %)
Iterazioni	estrazione *bad paths*		236
	reinserimento *bad paths*		10446
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12057
	Archi	totale	24981
		orientati	24951 (100 %)
		non orientati	30 (0 %)

Tabella 5.59: Caratteristiche del processo di inferenza

04/10/2001 00:00			
AS path	totale		537688
	Bad paths	estratti	14555
		reinseriti	11614 (80 %)
Iterazioni	estrazione *bad paths*		239
	reinserimento *bad paths*		11129
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12065
	Archi	totale	24992
		orientati	24957 (100 %)
		non orientati	35 (0 %)
04/10/2001 02:00			
AS path	totale		538514
	Bad paths	estratti	18073
		reinseriti	14220 (79 %)
Iterazioni	estrazione *bad paths*		255
	reinserimento *bad paths*		13103
Rapporto iterazioni/path reinseriti			0.97
Grafo degli AS	nodi	totale	12063
	Archi	totale	25004
		orientati	24969 (100 %)
		non orientati	35 (0 %)
04/10/2001 04:00			
AS path	totale		536241
	Bad paths	estratti	16970
		reinseriti	13357 (79 %)
Iterazioni	estrazione *bad paths*		251
	reinserimento *bad paths*		12476
Rapporto iterazioni/path reinseriti			0.97
Grafo degli AS	nodi	totale	12061
	Archi	totale	24974
		orientati	24944 (100 %)
		non orientati	30 (0 %)
04/10/2001 06:00			
AS path	totale		537952
	Bad paths	estratti	15138
		reinseriti	12138 (80 %)
Iterazioni	estrazione *bad paths*		254
	reinserimento *bad paths*		11346
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12066
	Archi	totale	25039
		orientati	25007 (100 %)
		non orientati	32 (0 %)

Tabella 5.60: Caratteristiche del processo di inferenza

04/10/2001 08:00			
AS path	totale		536735
	Bad paths	estratti	16110
		reinseriti	12717 (79 %)
Iterazioni	estrazione *bad paths*		247
	reinserimento *bad paths*		12038
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12065
	Archi	totale	24984
		orientati	24952 (100 %)
		non orientati	32 (0 %)
04/10/2001 10:00			
AS path	totale		537638
	Bad paths	estratti	15275
		reinseriti	12296 (80 %)
Iterazioni	estrazione *bad paths*		249
	reinserimento *bad paths*		11325
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12070
	Archi	totale	25060
		orientati	25029 (100 %)
		non orientati	31 (0 %)
04/10/2001 12:00			
AS path	totale		552954
	Bad paths	estratti	16499
		reinseriti	13499 (82 %)
Iterazioni	estrazione *bad paths*		250
	reinserimento *bad paths*		11824
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12077
	Archi	totale	25105
		orientati	25075 (100 %)
		non orientati	30 (0 %)
04/10/2001 14:00			
AS path	totale		552806
	Bad paths	estratti	17279
		reinseriti	13966 (81 %)
Iterazioni	estrazione *bad paths*		266
	reinserimento *bad paths*		12355
Rapporto iterazioni/path reinseriti			0.97
Grafo degli AS	nodi	totale	12069
	Archi	totale	25103
		orientati	25073 (100 %)
		non orientati	30 (0 %)

Tabella 5.61: Caratteristiche del processo di inferenza

04/10/2001 16:00			
AS path	totale		552187
	Bad paths	estratti	15129
		reinseriti	12435 (82 %)
Iterazioni	estrazione *bad paths*		244
	reinserimento *bad paths*		11280
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12068
	Archi	totale	25079
		orientati	25048 (100 %)
		non orientati	31 (0 %)
04/10/2001 18:00			
AS path	totale		557444
	Bad paths	estratti	15131
		reinseriti	12143 (80 %)
Iterazioni	estrazione *bad paths*		246
	reinserimento *bad paths*		11638
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12071
	Archi	totale	25091
		orientati	25057 (100 %)
		non orientati	34 (0 %)
04/10/2001 20:00			
AS path	totale		553130
	Bad paths	estratti	14935
		reinseriti	12072 (81 %)
Iterazioni	estrazione *bad paths*		249
	reinserimento *bad paths*		11080
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12071
	Archi	totale	25094
		orientati	25060 (100 %)
		non orientati	34 (0 %)
04/10/2001 22:00			
AS path	totale		552680
	Bad paths	estratti	15135
		reinseriti	12534 (83 %)
Iterazioni	estrazione *bad paths*		247
	reinserimento *bad paths*		11257
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12066
	Archi	totale	25090
		orientati	25056 (100 %)
		non orientati	34 (0 %)

Tabella 5.62: Caratteristiche del processo di inferenza

05/10/2001 00:00			
AS path	totale		553379
	Bad paths	estratti	15193
		reinseriti	12237 (81 %)
Iterazioni	estrazione *bad paths*		246
	reinserimento *bad paths*		11297
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12069
	Archi	totale	25103
		orientati	25072 (100 %)
		non orientati	31 (0 %)
05/10/2001 02:00			
AS path	totale		556742
	Bad paths	estratti	20568
		reinseriti	16770 (82 %)
Iterazioni	estrazione *bad paths*		264
	reinserimento *bad paths*		14421
Rapporto iterazioni/path reinseriti			0.97
Grafo degli AS	nodi	totale	12072
	Archi	totale	25120
		orientati	25093 (100 %)
		non orientati	27 (0 %)
05/10/2001 04:00			
AS path	totale		552508
	Bad paths	estratti	9422
		reinseriti	7341 (78 %)
Iterazioni	estrazione *bad paths*		215
	reinserimento *bad paths*		7756
Rapporto iterazioni/path reinseriti			0.99
Grafo degli AS	nodi	totale	12068
	Archi	totale	25102
		orientati	25072 (100 %)
		non orientati	30 (0 %)
05/10/2001 06:00			
AS path	totale		552445
	Bad paths	estratti	9343
		reinseriti	7207 (77 %)
Iterazioni	estrazione *bad paths*		220
	reinserimento *bad paths*		7903
Rapporto iterazioni/path reinseriti			0.99
Grafo degli AS	nodi	totale	12070
	Archi	totale	25109
		orientati	25080 (100 %)
		non orientati	29 (0 %)

Tabella 5.63: Caratteristiche del processo di inferenza

05/10/2001 08:00			
AS path	totale		551519
	Bad paths	estratti	9930
		reinseriti	8129 (82 %)
Iterazioni	estrazione *bad paths*		225
	reinserimento *bad paths*		7862
Rapporto iterazioni/path reinseriti			0.99
Grafo degli AS	nodi	totale	12073
	Archi	totale	25119
		orientati	25089 (100 %)
		non orientati	30 (0 %)

05/10/2001 10:00			
AS path	totale		553104
	Bad paths	estratti	13597
		reinseriti	11293 (83 %)
Iterazioni	estrazione *bad paths*		235
	reinserimento *bad paths*		9734
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12072
	Archi	totale	25114
		orientati	25085 (100 %)
		non orientati	29 (0 %)

05/10/2001 12:00			
AS path	totale		553200
	Bad paths	estratti	11598
		reinseriti	9120 (79 %)
Iterazioni	estrazione *bad paths*		247
	reinserimento *bad paths*		9449
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12081
	Archi	totale	25113
		orientati	25080 (100 %)
		non orientati	33 (0 %)

05/10/2001 14:00			
AS path	totale		552154
	Bad paths	estratti	10807
		reinseriti	8519 (79 %)
Iterazioni	estrazione *bad paths*		230
	reinserimento *bad paths*		8791
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12082
	Archi	totale	25121
		orientati	25091 (100 %)
		non orientati	30 (0 %)

Tabella 5.64: Caratteristiche del processo di inferenza

05/10/2001 16:00			
AS path	totale		552803
	Bad paths	estratti	10823
		reinseriti	8619 (80 %)
Iterazioni	estrazione *bad paths*		230
	reinserimento *bad paths*		8961
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12075
	Archi	totale	25113
		orientati	25083 (100 %)
		non orientati	30 (0 %)
05/10/2001 18:00			
AS path	totale		552673
	Bad paths	estratti	10555
		reinseriti	8308 (79 %)
Iterazioni	estrazione *bad paths*		227
	reinserimento *bad paths*		8700
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12073
	Archi	totale	25096
		orientati	25069 (100 %)
		non orientati	27 (0 %)
05/10/2001 20:00			
AS path	totale		553252
	Bad paths	estratti	11078
		reinseriti	9089 (82 %)
Iterazioni	estrazione *bad paths*		230
	reinserimento *bad paths*		8583
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12078
	Archi	totale	25098
		orientati	25070 (100 %)
		non orientati	28 (0 %)
05/10/2001 22:00			
AS path	totale		553201
	Bad paths	estratti	10694
		reinseriti	8332 (78 %)
Iterazioni	estrazione *bad paths*		226
	reinserimento *bad paths*		8769
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12078
	Archi	totale	25075
		orientati	25043 (100 %)
		non orientati	32 (0 %)

Tabella 5.65: Caratteristiche del processo di inferenza

06/10/2001 00:00			
AS path	totale		553766
	Bad paths	estratti	10306
		reinseriti	8496 (82 %)
Iterazioni	estrazione *bad paths*		229
	reinserimento *bad paths*		8047
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12081
	Archi	totale	25102
		orientati	25076 (100 %)
		non orientati	26 (0 %)
06/10/2001 02:00			
AS path	totale		553173
	Bad paths	estratti	13808
		reinseriti	11463 (83 %)
Iterazioni	estrazione *bad paths*		239
	reinserimento *bad paths*		9643
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12071
	Archi	totale	25082
		orientati	25053 (100 %)
		non orientati	29 (0 %)
06/10/2001 04:00			
AS path	totale		552849
	Bad paths	estratti	10473
		reinseriti	8488 (81 %)
Iterazioni	estrazione *bad paths*		229
	reinserimento *bad paths*		8278
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12072
	Archi	totale	25093
		orientati	25066 (100 %)
		non orientati	27 (0 %)
06/10/2001 06:00			
AS path	totale		552998
	Bad paths	estratti	8700
		reinseriti	6733 (77 %)
Iterazioni	estrazione *bad paths*		211
	reinserimento *bad paths*		7798
Rapporto iterazioni/path reinseriti			0.99
Grafo degli AS	nodi	totale	12073
	Archi	totale	25093
		orientati	25061 (100 %)
		non orientati	32 (0 %)

Tabella 5.66: Caratteristiche del processo di inferenza

06/10/2001 08:00			
AS path	totale		552958
	Bad paths	estratti	9163
		reinseriti	7197 (79 %)
Iterazioni	estrazione *bad paths*		215
	reinserimento *bad paths*		7809
Rapporto iterazioni/path reinseriti			0.99
Grafo degli AS	nodi	totale	12075
	Archi	totale	25108
		orientati	25071 (100 %)
		non orientati	37 (0 %)

06/10/2001 10:00			
AS path	totale		552984
	Bad paths	estratti	8670
		reinseriti	6863 (79 %)
Iterazioni	estrazione *bad paths*		210
	reinserimento *bad paths*		7592
Rapporto iterazioni/path reinseriti			0.99
Grafo degli AS	nodi	totale	12073
	Archi	totale	25113
		orientati	25085 (100 %)
		non orientati	28 (0 %)

06/10/2001 12:00			
AS path	totale		552770
	Bad paths	estratti	8550
		reinseriti	6630 (78 %)
Iterazioni	estrazione *bad paths*		208
	reinserimento *bad paths*		7615
Rapporto iterazioni/path reinseriti			0.99
Grafo degli AS	nodi	totale	12077
	Archi	totale	25110
		orientati	25080 (100 %)
		non orientati	30 (0 %)

06/10/2001 14:00			
AS path	totale		552759
	Bad paths	estratti	9366
		reinseriti	7470 (80 %)
Iterazioni	estrazione *bad paths*		220
	reinserimento *bad paths*		7949
Rapporto iterazioni/path reinseriti			0.99
Grafo degli AS	nodi	totale	12077
	Archi	totale	25123
		orientati	25092 (100 %)
		non orientati	31 (0 %)

Tabella 5.67: Caratteristiche del processo di inferenza

06/10/2001 16:00			
AS path	totale		552873
	Bad paths	estratti	9316
		reinseriti	7357 (79 %)
Iterazioni	estrazione *bad paths*		217
	reinserimento *bad paths*		7944
Rapporto iterazioni/path reinseriti			0.99
Grafo degli AS	nodi	totale	12075
	Archi	totale	25101
		orientati	25070 (100 %)
		non orientati	31 (0 %)
06/10/2001 18:00			
AS path	totale		552586
	Bad paths	estratti	8904
		reinseriti	6904 (78 %)
Iterazioni	estrazione *bad paths*		209
	reinserimento *bad paths*		7739
Rapporto iterazioni/path reinseriti			0.99
Grafo degli AS	nodi	totale	12071
	Archi	totale	25098
		orientati	25067 (100 %)
		non orientati	31 (0 %)
06/10/2001 20:00			
AS path	totale		537579
	Bad paths	estratti	9007
		reinseriti	7106 (79 %)
Iterazioni	estrazione *bad paths*		208
	reinserimento *bad paths*		7663
Rapporto iterazioni/path reinseriti			0.99
Grafo degli AS	nodi	totale	12075
	Archi	totale	25069
		orientati	25042 (100 %)
		non orientati	27 (0 %)
06/10/2001 22:00			
AS path	totale		553293
	Bad paths	estratti	9453
		reinseriti	7548 (80 %)
Iterazioni	estrazione *bad paths*		214
	reinserimento *bad paths*		8034
Rapporto iterazioni/path reinseriti			0.99
Grafo degli AS	nodi	totale	12076
	Archi	totale	25114
		orientati	25086 (100 %)
		non orientati	28 (0 %)

Tabella 5.68: Caratteristiche del processo di inferenza

07/10/2001 00:00			
AS path	totale		553478
	Bad paths	estratti	9377
		reinseriti	7690 (82 %)
Iterazioni	estrazione *bad paths*		214
	reinserimento *bad paths*		7529
Rapporto iterazioni/path reinseriti			0.99
Grafo degli AS	nodi	totale	12077
	Archi	totale	25121
		orientati	25097 (100 %)
		non orientati	24 (0 %)

07/10/2001 02:00			
AS path	totale		553212
	Bad paths	estratti	9269
		reinseriti	7571 (82 %)
Iterazioni	estrazione *bad paths*		214
	reinserimento *bad paths*		7521
Rapporto iterazioni/path reinseriti			0.99
Grafo degli AS	nodi	totale	12074
	Archi	totale	25119
		orientati	25094 (100 %)
		non orientati	25 (0 %)

07/10/2001 04:00			
AS path	totale		553040
	Bad paths	estratti	9183
		reinseriti	7236 (79 %)
Iterazioni	estrazione *bad paths*		210
	reinserimento *bad paths*		7824
Rapporto iterazioni/path reinseriti			0.99
Grafo degli AS	nodi	totale	12076
	Archi	totale	25115
		orientati	25086 (100 %)
		non orientati	29 (0 %)

07/10/2001 06:00			
AS path	totale		552984
	Bad paths	estratti	8921
		reinseriti	7063 (79 %)
Iterazioni	estrazione *bad paths*		212
	reinserimento *bad paths*		7767
Rapporto iterazioni/path reinseriti			0.99
Grafo degli AS	nodi	totale	12072
	Archi	totale	25103
		orientati	25078 (100 %)
		non orientati	25 (0 %)

Tabella 5.69: Caratteristiche del processo di inferenza

07/10/2001 08:00			
AS path	totale		553302
	Bad paths	estratti	9193
		reinseriti	7234 (79 %)
Iterazioni	estrazione *bad paths*		213
	reinserimento *bad paths*		7946
Rapporto iterazioni/path reinseriti			0.99
Grafo degli AS	nodi	totale	12075
	Archi	totale	25117
		orientati	25088 (100 %)
		non orientati	29 (0 %)
07/10/2001 10:00			
AS path	totale		553096
	Bad paths	estratti	8994
		reinseriti	7318 (81 %)
Iterazioni	estrazione *bad paths*		208
	reinserimento *bad paths*		7444
Rapporto iterazioni/path reinseriti			0.99
Grafo degli AS	nodi	totale	12074
	Archi	totale	25126
		orientati	25101 (100 %)
		non orientati	25 (0 %)
07/10/2001 12:00			
AS path	totale		553324
	Bad paths	estratti	8880
		reinseriti	6892 (78 %)
Iterazioni	estrazione *bad paths*		208
	reinserimento *bad paths*		7819
Rapporto iterazioni/path reinseriti			0.99
Grafo degli AS	nodi	totale	12083
	Archi	totale	25136
		orientati	25100 (100 %)
		non orientati	36 (0 %)
07/10/2001 14:00			
AS path	totale		539161
	Bad paths	estratti	10778
		reinseriti	8635 (80 %)
Iterazioni	estrazione *bad paths*		235
	reinserimento *bad paths*		8615
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12076
	Archi	totale	25113
		orientati	25092 (100 %)
		non orientati	21 (0 %)

Tabella 5.70: Caratteristiche del processo di inferenza

07/10/2001 16:00			
AS path	totale		552091
	Bad paths	estratti	10769
		reinseriti	8655 (80 %)
Iterazioni	estrazione *bad paths*		231
	reinserimento *bad paths*		8723
Rapporto iterazioni/path reinseriti			0.98
Grafo degli AS	nodi	totale	12071
	Archi	totale	25097
		orientati	25069 (100 %)
		non orientati	28 (0 %)

07/10/2001 18:00			
AS path	totale		552477
	Bad paths	estratti	9091
		reinseriti	7211 (79 %)
Iterazioni	estrazione *bad paths*		217
	reinserimento *bad paths*		7750
Rapporto iterazioni/path reinseriti			0.99
Grafo degli AS	nodi	totale	12068
	Archi	totale	25093
		orientati	25070 (100 %)
		non orientati	23 (0 %)

07/10/2001 20:00			
AS path	totale		553303
	Bad paths	estratti	10027
		reinseriti	8031 (80 %)
Iterazioni	estrazione *bad paths*		220
	reinserimento *bad paths*		8072
Rapporto iterazioni/path reinseriti			0.99
Grafo degli AS	nodi	totale	12074
	Archi	totale	25128
		orientati	25102 (100 %)
		non orientati	26 (0 %)

07/10/2001 22:00			
AS path	totale		553086
	Bad paths	estratti	9174
		reinseriti	7082 (77 %)
Iterazioni	estrazione *bad paths*		215
	reinserimento *bad paths*		8150
Rapporto iterazioni/path reinseriti			0.99
Grafo degli AS	nodi	totale	12071
	Archi	totale	25115
		orientati	25083 (100 %)
		non orientati	32 (0 %)

Tabella 5.71: Caratteristiche del processo di inferenza

08/10/2001 00:00			
AS path	totale		553261
	Bad paths	estratti	9828
		reinseriti	7805 (79 %)
Iterazioni	estrazione *bad paths*		223
	reinserimento *bad paths*		8402
Rapporto iterazioni/path reinseriti			0.99
Grafo degli AS	nodi	totale	12077
	Archi	totale	25145
		orientati	25108 (100 %)
		non orientati	37 (0 %)
08/10/2001 02:00			
AS path	totale		553893
	Bad paths	estratti	9672
		reinseriti	7955 (82 %)
Iterazioni	estrazione *bad paths*		221
	reinserimento *bad paths*		8063
Rapporto iterazioni/path reinseriti			0.99
Grafo degli AS	nodi	totale	12078
	Archi	totale	25142
		orientati	25115 (100 %)
		non orientati	27 (0 %)
08/10/2001 04:00			
AS path	totale		553945
	Bad paths	estratti	8990
		reinseriti	7202 (80 %)
Iterazioni	estrazione *bad paths*		221
	reinserimento *bad paths*		7653
Rapporto iterazioni/path reinseriti			0.99
Grafo degli AS	nodi	totale	12082
	Archi	totale	25136
		orientati	25108 (100 %)
		non orientati	28 (0 %)
08/10/2001 06:00			
AS path	totale		553683
	Bad paths	estratti	9052
		reinseriti	7587 (84 %)
Iterazioni	estrazione *bad paths*		214
	reinserimento *bad paths*		7012
Rapporto iterazioni/path reinseriti			0.99
Grafo degli AS	nodi	totale	12079
	Archi	totale	25064
		orientati	25042 (100 %)
		non orientati	22 (0 %)

Tabella 5.72: Caratteristiche del processo di inferenza

08/10/2001 08:00			
AS path	totale		553684
	Bad paths	estratti	8724
		reinseriti	7235 (83 %)
Iterazioni	estrazione *bad paths*		217
	reinserimento *bad paths*		6988
Rapporto iterazioni/path reinseriti			0.99
Grafo degli AS	nodi	totale	12077
	Archi	totale	25074
		orientati	25055 (100 %)
		non orientati	19 (0 %)
08/10/2001 10:00			
AS path	totale		553824
	Bad paths	estratti	9510
		reinseriti	7647 (80 %)
Iterazioni	estrazione *bad paths*		220
	reinserimento *bad paths*		7680
Rapporto iterazioni/path reinseriti			0.99
Grafo degli AS	nodi	totale	12074
	Archi	totale	25095
		orientati	25074 (100 %)
		non orientati	21 (0 %)
08/10/2001 12:00			
AS path	totale		553868
	Bad paths	estratti	9524
		reinseriti	7591 (80 %)
Iterazioni	estrazione *bad paths*		221
	reinserimento *bad paths*		7773
Rapporto iterazioni/path reinseriti			0.99
Grafo degli AS	nodi	totale	12079
	Archi	totale	25100
		orientati	25078 (100 %)
		non orientati	22 (0 %)
08/10/2001 14:00			
AS path	totale		554125
	Bad paths	estratti	9384
		reinseriti	7791 (83 %)
Iterazioni	estrazione *bad paths*		225
	reinserimento *bad paths*		7725
Rapporto iterazioni/path reinseriti			0.99
Grafo degli AS	nodi	totale	12081
	Archi	totale	25109
		orientati	25090 (100 %)
		non orientati	19 (0 %)

Tabella 5.73: Caratteristiche del processo di inferenza

08/10/2001 16:00			
AS path	totale		554123
	Bad paths	estratti	10029
		reinseriti	8109 (81 %)
Iterazioni	estrazione *bad paths*		228
	reinserimento *bad paths*		7828
Rapporto iterazioni/path reinseriti			0.99
Grafo degli AS	nodi	totale	12083
	Archi	totale	25112
		orientati	25090 (100 %)
		non orientati	22 (0 %)

08/10/2001 18:00			
AS path	totale		553938
	Bad paths	estratti	9016
		reinseriti	7492 (83 %)
Iterazioni	estrazione *bad paths*		218
	reinserimento *bad paths*		7266
Rapporto iterazioni/path reinseriti			0.99
Grafo degli AS	nodi	totale	12084
	Archi	totale	25119
		orientati	25097 (100 %)
		non orientati	22 (0 %)

08/10/2001 20:00			
AS path	totale		554070
	Bad paths	estratti	8916
		reinseriti	7435 (83 %)
Iterazioni	estrazione *bad paths*		208
	reinserimento *bad paths*		6916
Rapporto iterazioni/path reinseriti			0.99
Grafo degli AS	nodi	totale	12086
	Archi	totale	25109
		orientati	25088 (100 %)
		non orientati	21 (0 %)

08/10/2001 22:00			
AS path	totale		554006
	Bad paths	estratti	9664
		reinseriti	7839 (81 %)
Iterazioni	estrazione *bad paths*		226
	reinserimento *bad paths*		7681
Rapporto iterazioni/path reinseriti			0.99
Grafo degli AS	nodi	totale	12085
	Archi	totale	25101
		orientati	25077 (100 %)
		non orientati	24 (0 %)

Tabella 5.74: Caratteristiche del processo di inferenza

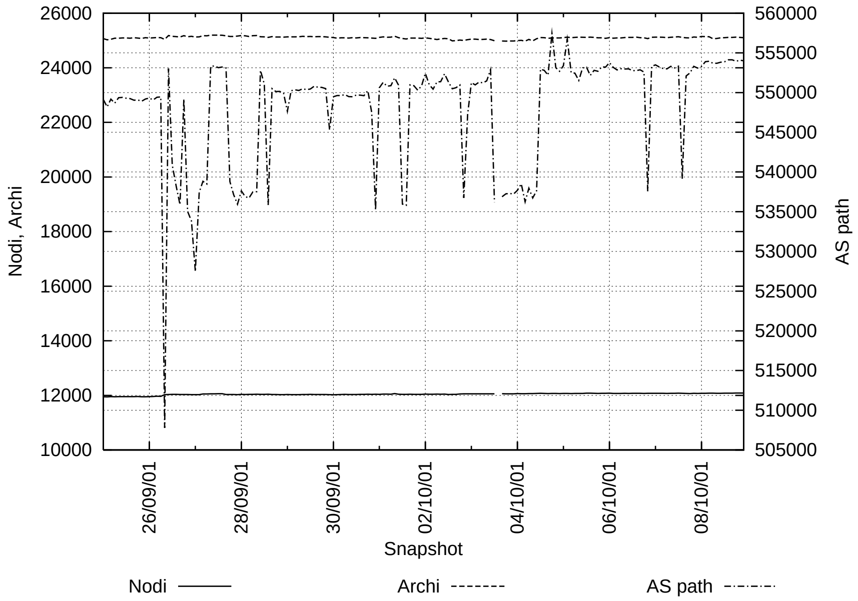

Figura 5.25: Numero di AS path e di nodi ed archi del grafo degli AS visibili da Route Views

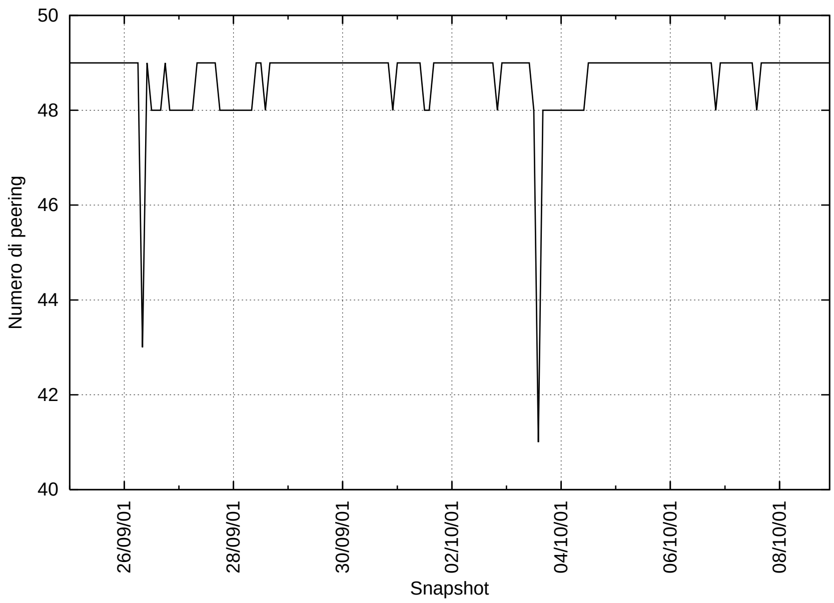

Figura 5.26: Numero di sessioni di peering attive tra Route Views ed altri router

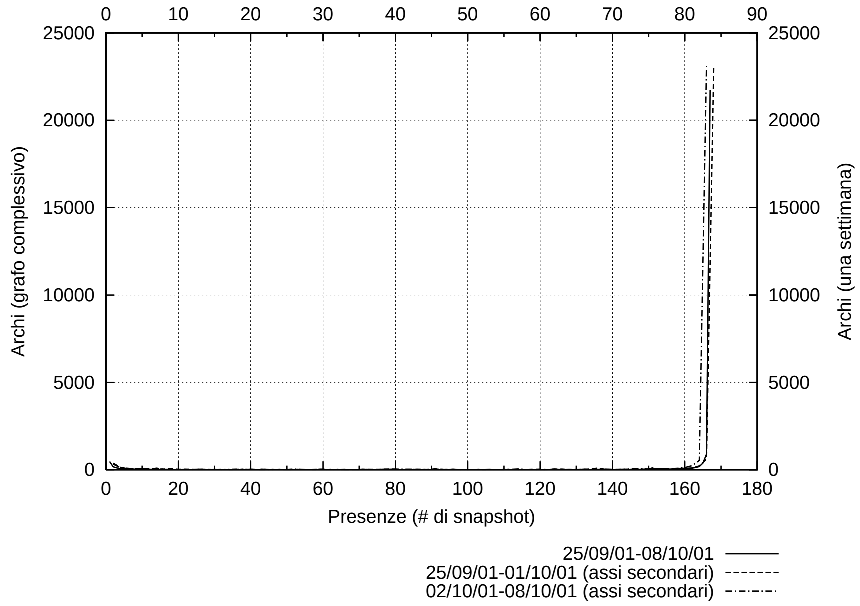

Figura 5.27: Distribuzione del numero di presenze di ciascun arco nei vari snapshot

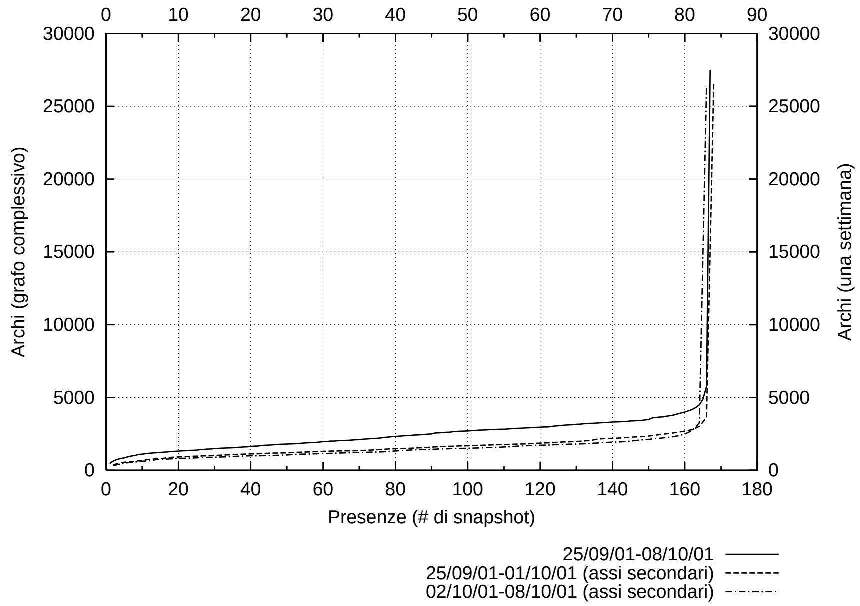

Figura 5.28: Distribuzione cumulativa del numero di presenze di ciascun arco nei vari snapshot (numero y di archi con al più x presenze)

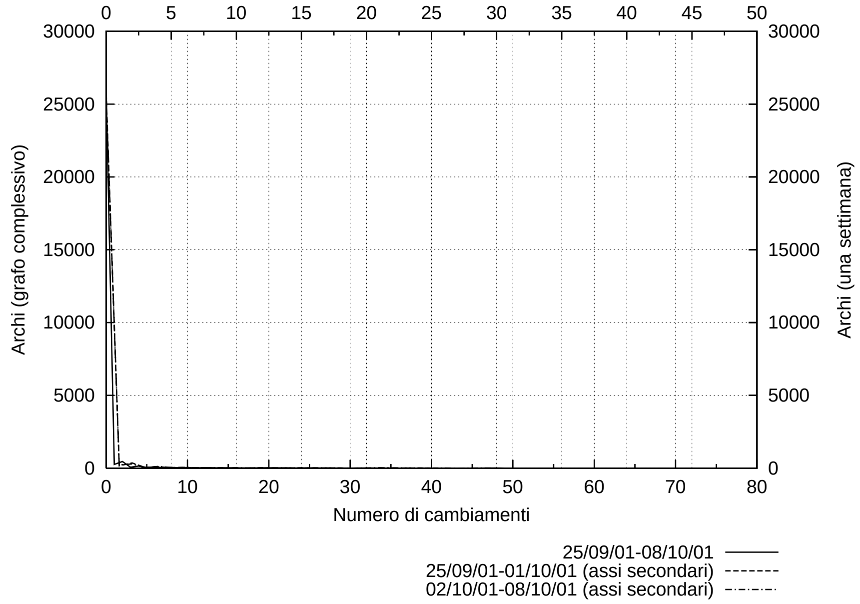

Figura 5.29: Distribuzione del numero di cambiamenti di orientazione che ciascun arco ha subito tra due snapshot consecutivi (inclusi i passaggi dallo stato orientato a quello non orientato e viceversa)

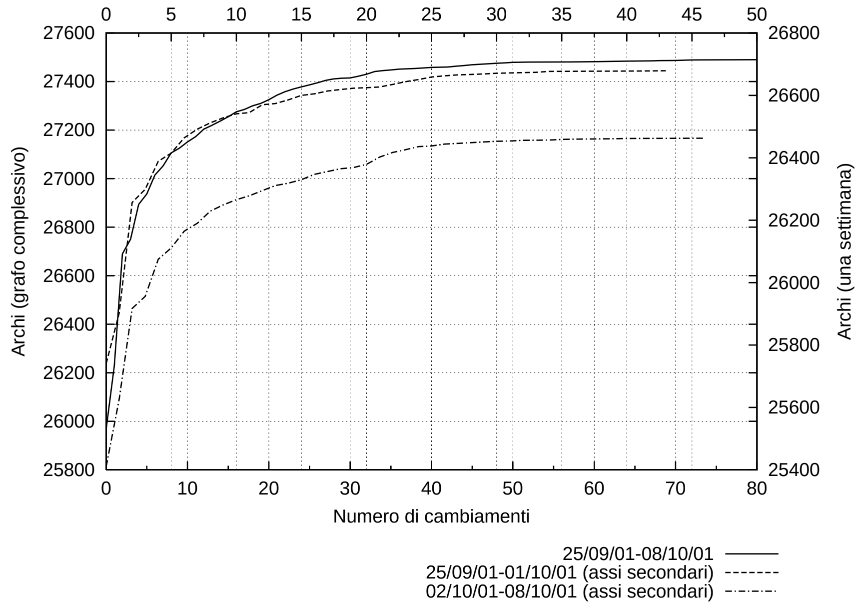

Figura 5.30: Distribuzione cumulativa del numero di cambiamenti di orientazione che ciascun arco ha subito tra due snapshot consecutivi (inclusi i passaggi dallo stato orientato a quello non orientato e viceversa)

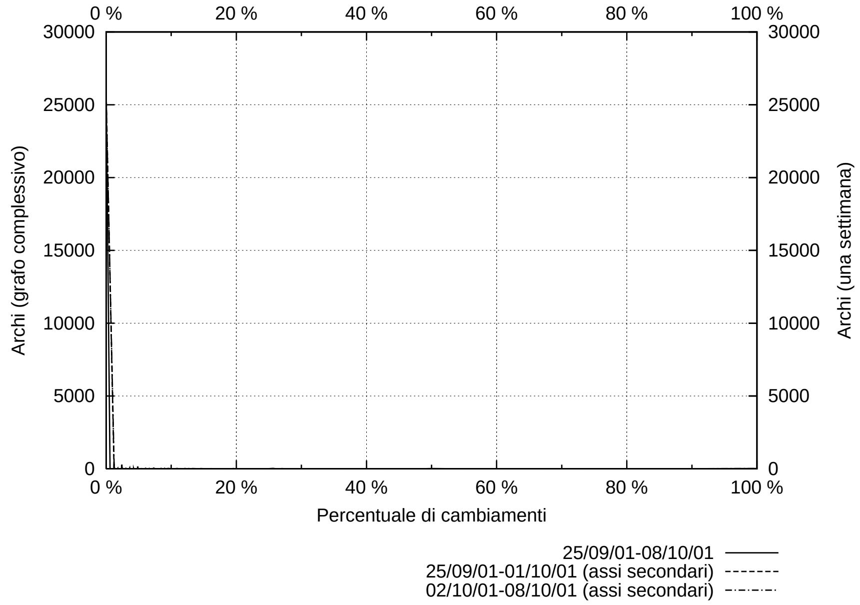

Figura 5.31: Distribuzione del rapporto tra i cambiamenti di orientazione che ciascun arco ha subito ed il numero delle sue presenze (ad esempio, gli archi riportati in corrispondenza del valore 100% hanno cambiato orientazione in ogni snapshot in cui sono stati presenti)

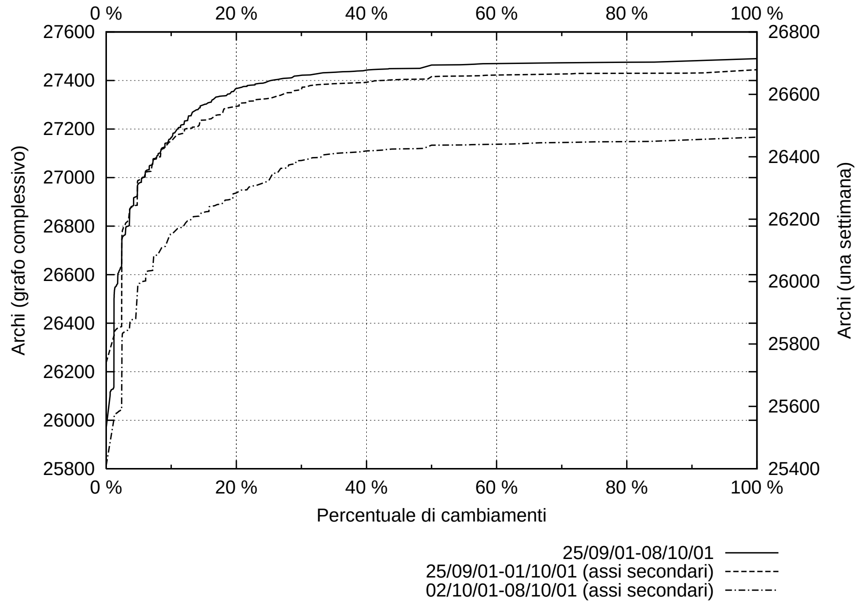

Figura 5.32: Distribuzione cumulativa del rapporto tra i cambiamenti di orientazione che ciascun arco ha subito ed il numero delle sue presenze

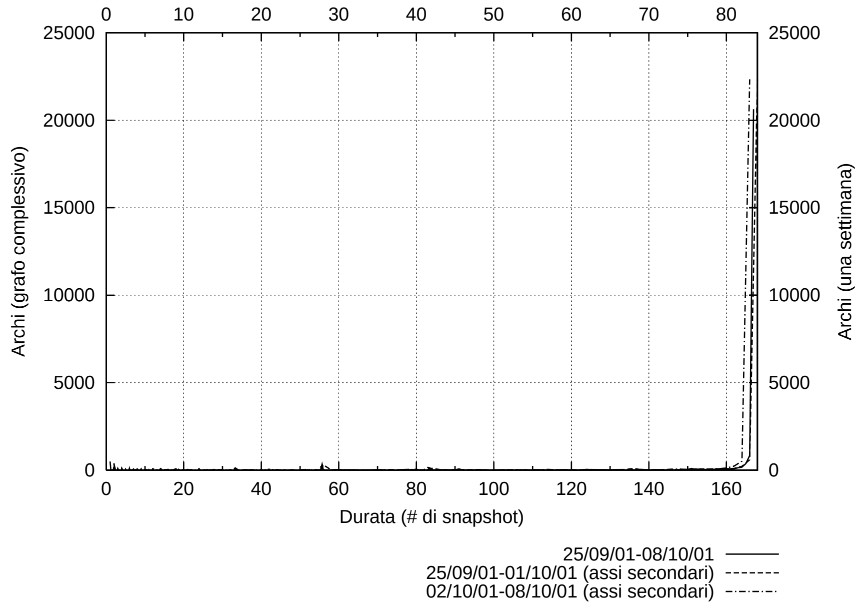

Figura 5.33: Distribuzione della durata media di un'orientazione (in termini di numero di snapshot consecutivi per cui viene mantenuta)

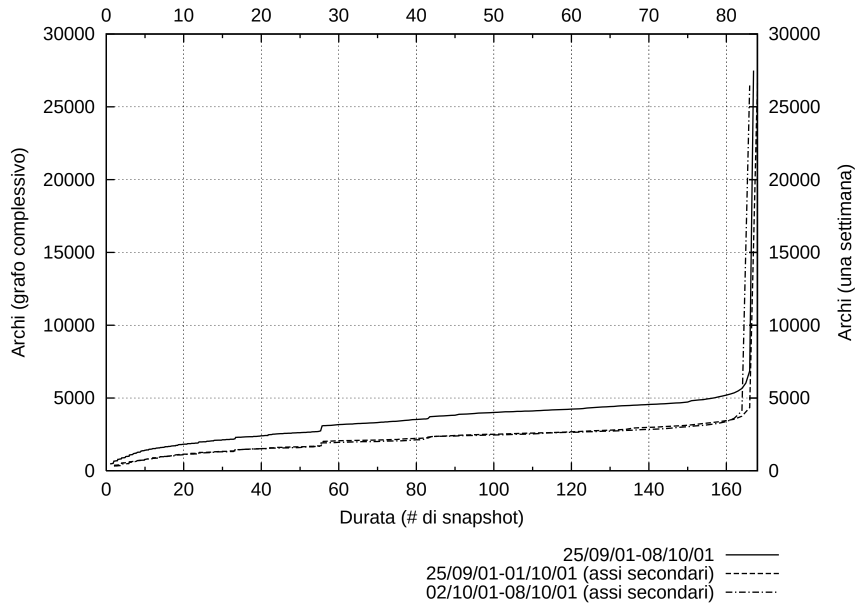

Figura 5.34: Distribuzione cumulativa della durata media di un'orientazione (in termini di numero di snapshot consecutivi per cui viene mantenuta)

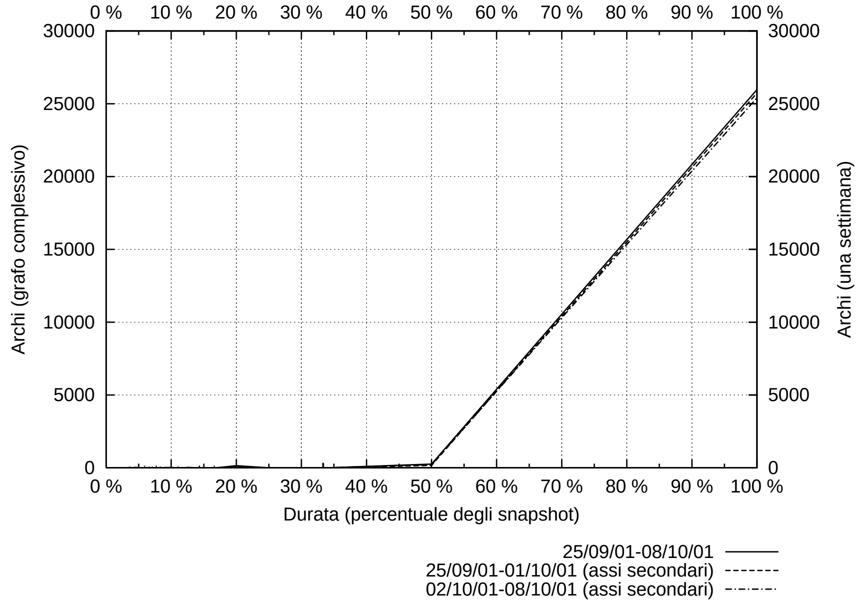

Figura 5.35: Distribuzione della durata media di un'orientazione (in termini di numero di snapshot consecutivi per cui viene mantenuta) rapportata al numero di presenze di ciascun arco (ad esempio, il valore in corrispondenza di 100% è il numero di archi che hanno mantenuto sempre stabile la propria orientazione)

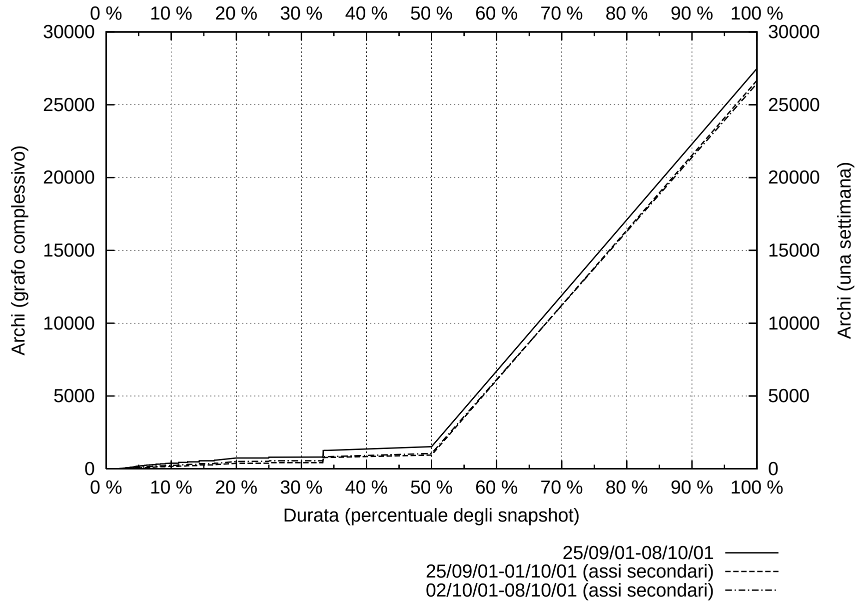

Figura 5.36: Distribuzione cumulativa della durata media di un'orientazione (in termini di numero di snapshot consecutivi per cui viene mantenuta) rapportata al numero di presenze di ciascun arco

5.3.4 Conclusioni

Ad ulteriore conferma della qualità dell'orientazione calcolata, nonostante più del 90% degli archi sia visibile in tutti gli snapshot, non vi sono quasi archi che cambino orientazione più del 40% delle volte in cui possono farlo[1], ed in particolare nessun arco oscilla più di 80 volte (la metà del numero di snapshot utilizzati).

In questo caso il 94% degli archi viene orientato in modo completamente stabile (cioè non subisce mai cambiamenti di orientazione), mentre per la sola prima e seconda settimana tale percentuale sale al 96%. L'estrema vicinanza delle curve relative a tutto l'intervallo temporale considerato e ad una sola delle due settimane che lo compongono mostra come il leggero calo nella stabilità rispetto alla sperimentazione precedente sia da imputare non alla maggiore estensione dell'intervallo stesso, ma piuttosto alla presenza di cambiamenti commerciali. Inoltre, si può ritenere minimamente influente anche l'apporto delle oscillazioni dei dati di routing, poiché nel test precedente è stato mostrato come esse non perturbino la soluzione calcolata.

D'altra parte, ciascuno degli eventi commerciali elencati può avere più o meno impatto sulle stesse informazioni di routing, e pertanto era lecito aspettarsi che la stabilità non fosse compromessa in modo più evidente di quanto non sia accaduto.

Per confermare la correttezza dei risultati, è stata anche effettuata un'analisi delle orientazioni in prossimità degli AS coinvolti nei nuovi accordi commerciali. Tuttavia, essa ha portato alla luce il fatto che non esistono sessioni di peering dirette tra di essi, sicché non è possibile trovare archi che li colleghino. L'unica possibilità per attuare questo tipo di valutazione sarebbe quella di lavorare sulla chiusura transitiva[2] dei vari grafi degli AS, in modo tale da ottenere un arco per ogni relazione esistente (e non in corrispondenza delle sole sessioni di peering attive), ma in questa sede questa operazione non viene effettuata.

Tuttavia, è stata effettuata una valutazione di carattere qualitativo mirata a rilevare l'eventuale presenza di orientazioni oscillanti tra coppie di AS di particolare importanza. L'analisi è stata effettuata considerando lo spazio di indirizzamento annunciato in ogni coppia, e valutando la relazione tra tale spazio ed il numero di cambiamenti di orientazione subiti dal corrispondente arco. Nelle figure 5.37 e 5.38 è possibile apprezzare il fatto che gli archi sui quali hanno avuto luogo molti cambiamenti sono stati attraversati da annunci di insiemi estremamente limitati di indirizzi IP, il che fa pensare che essi corrispondano a coppie con relazioni di limitata importanza. Viceversa, gli archi con annunci più ampi hanno mantenuto più stabile la propria orientazione.

Questo consente di concludere che l'algoritmo di inferenza, in presenza di cambiamenti commerciali noti a priori, non perturba eccessivamente orientazioni di archi non coinvolti in tali cambiamenti, e, se questo accade, lo fa soltanto su AS che rappresentino realtà di piccole dimensioni.

[1] Il numero massimo di volte in cui un arco può cambiare orientazione è, evidentemente, pari al numero delle sue presenze meno 1.

[2] La chiusura transitiva di un grafo $G = (N, E)$ è un grafo $G' = (N', E')$ avente gli stessi nodi di G ed in cui la presenza di un arco $v'_i \rightarrow v'_j$ è biunivocamente correlata con l'esistenza in G di un cammino orientato da v_i a v_j.

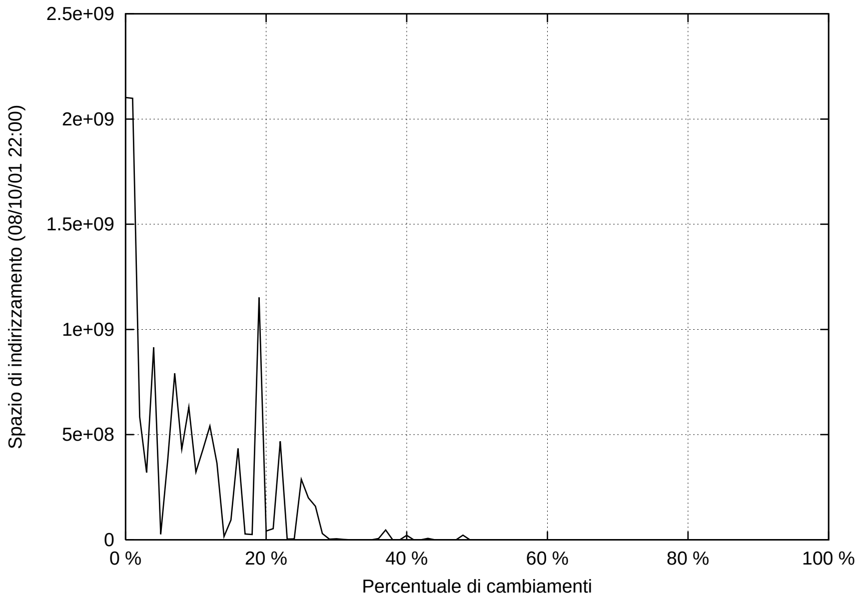

Figura 5.37: Distribuzione della percentuale di cambiamenti di orientazione subiti da ogni arco

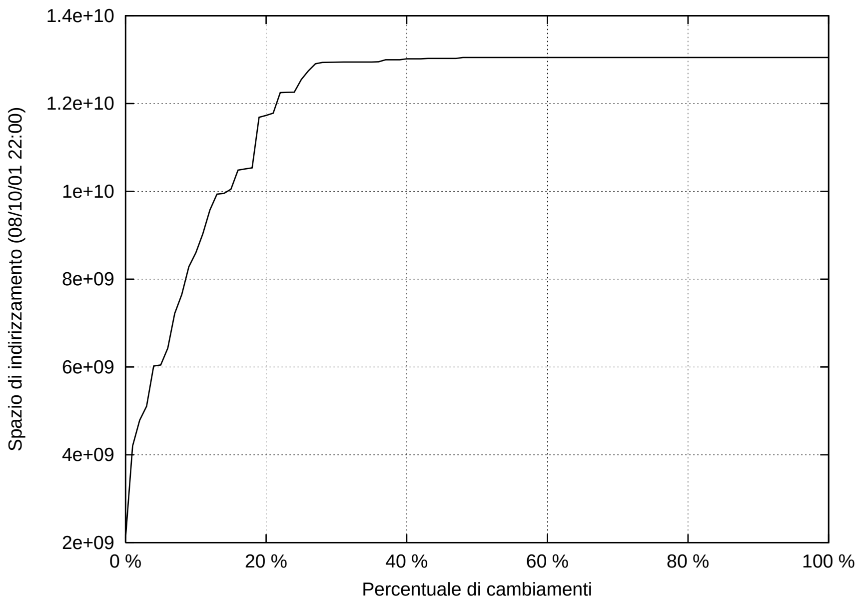

Figura 5.38: Distribuzione cumulativa della percentuale di cambiamenti di orientazione subiti da ogni arco

Capitolo 6

Conclusioni

In questa trattazione è stato illustrato il problema dell'analisi delle relazioni commerciali tra Autonomous Systems sotto una prospettiva temporale. Sono state presentate le motivazioni che generalmente costringono a ricavare tali relazioni basandosi su opportuni algoritmi di inferenza, e sono stati mostrati tre approcci esistenti con i quali sono stati sviluppati tali algoritmi. Sono state inoltre illustrate la progettazione e realizzazione di un ambiente che rendesse lo studio dell'evoluzione temporale delle relazioni il più possibile semplice, sistematico ed efficace. Per utilizzare tale sistema al fine di studiare il comportamento di uno degli algoritmi di inferenza, si è scelto di prendere in esame la soluzione proposta in [20]. Prima di poterla utilizzare, si è reso necessario ottimizzarne l'efficienza, ottenendo un guadagno in termini di tempi necessari per il calcolo decisamente notevole: dalle 15 alle 23 ore. L'implementazione dell'algoritmo così ottimizzato è stata poi eseguita su numerosi dati di routing, prelevati da sorgenti diverse ed in periodi con variazioni di carattere commerciale più o meno rilevanti.

I risultati ottenuti mostrano come le relazioni calcolate si mantengano molto stabili, nonostante la presenza di eventuali impurità ed incoerenze nei dati di routing (oscillazioni nel numero di AS path, variazioni nei looking glass considerati per ogni snapshot). In particolare, quando eseguito sui dati in [9], l'algoritmo mantiene sempre stabile l'orientazione del 95% degli archi del grafo ottenuto dall'unione dei grafi degli AS costruiti ad ogni snapshot; nel caso dei dati di Oregon Route Views ([22]) prelevati in una settimana priva di eventi commerciali significativi tale percentuale sale invece al 98%. Infine, ancora sui dati di Route Views, ma in un arco di due settimane con un numero abbastanza elevato di cambiamenti di relazione tra AS, la percentuale scende al 94%. Tale diminuzione non è da imputarsi alla maggiore ampiezza dell'intervallo temporale, in quanto, anche isolando una sola settimana, il valore non sale oltre il 96%. Piuttosto, è lecito pensare che siano state le stesse variazioni commerciali a far oscillare alcune orientazioni. Nell'impossibilità di effettuare un'analisi diretta delle orientazioni degli archi tra gli AS coinvolti nelle variazioni, si è preferito valutare la distribuzione di queste ultime rispetto agli indirizzi IP annunciati tra le varie coppie di AS. Il risultato ottenuto mostra che sugli archi che cambiano orientazione più del 30% delle volte in cui possono farlo, gli spazi di indirizzamento annunciati sono estremamente piccoli, il che fa pensare che le relazioni che vi corrispondono siano di secondaria importanza. D'altra parte,

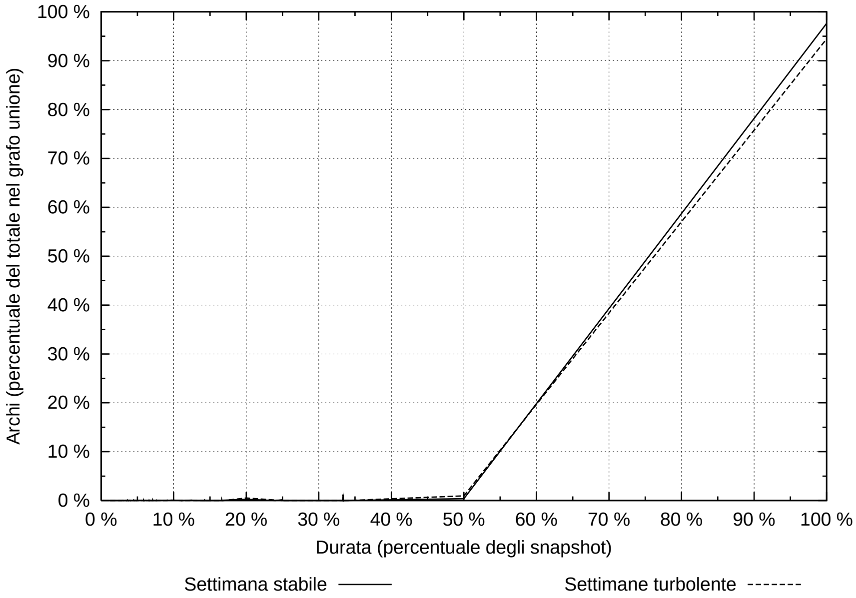

Figura 6.1: Distribuzione percentuale della durata delle orientazioni degli archi

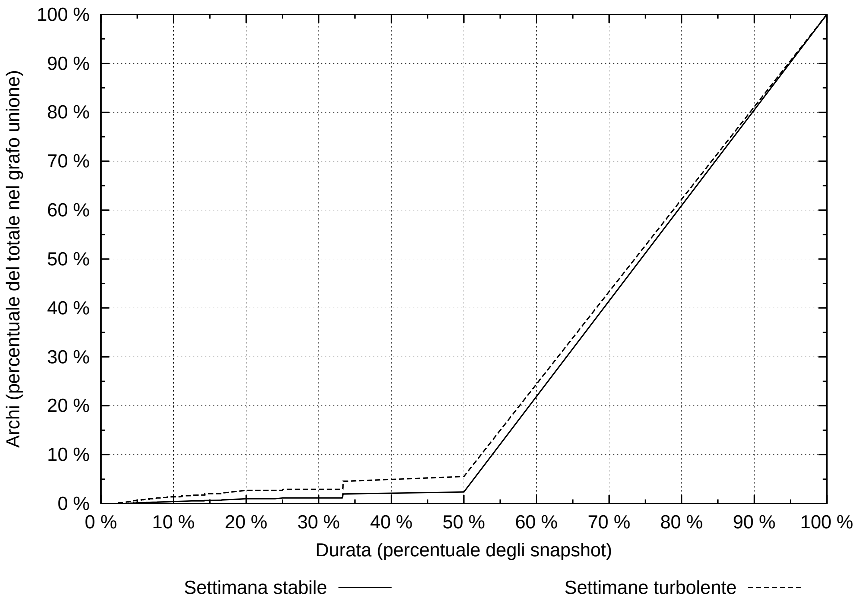

Figura 6.2: Distribuzione percentuale cumulativa della durata delle orientazioni degli archi

l'impatto dei nuovi contratti stipulati sulle politiche di routing può essere abbastanza limitato, ed è effettivamente auspicabile che essi non facciano oscillare le orientazioni in modo eccessivo.

In figura 6.1 è possibile osservare i risultati delle due computazioni sui dati di Oregon Route Views messi a confronto. Si può notare che, a parte alcuni archi che oscillano abbastanza frequentemente (durata inferiore al 50%), la maggior parte degli archi mantiene la propria orientazione in tutti gli snapshot[1], e la percentuale di tali archi risulta leggermente più bassa nel caso del periodo con molti avvenimenti commerciali significativi. La tendenza ad una minore stabilità è ancora più visibile in figura 6.2, in cui si può apprezzare meglio il fatto che il numero di archi la cui orientazione viene mantenuta per un tempo inferiore risulta più alto nel caso delle settimane "turbolente" (per esempio, gli archi stabili in al più il 40% degli snapshot sono di più nel corso di queste ultime che nella settimana stabile; di conseguenza, che quest'ultima deve invece avere più archi che sono stabili in più del 40% dei casi).

In definitiva, l'algoritmo utilizzato sembra comportarsi ottimamente dal punto di vista della stabilità con cui mantiene le relazioni calcolate, ed in presenza di cambiamenti commerciali tende a modificare con maggior frequenza soltanto le relazioni tra coppie di AS di limitata importanza.

6.1 Problemi aperti

Nel corso dell'analisi descritta si sono manifestati molti nuovi problemi che, se risolti, potrebbero portare ad approfondirla ulteriormente ed a migliorare l'algoritmo di inferenza delle relazioni.

- **Grafi delle relazioni** — Innanzitutto, potrebbe essere interessante basare il calcolo di tutti i risultati presentati anziché su grafi di AS orientati in base a quanto dettato da un algoritmo di inferenza delle relazioni, sulla loro chiusura transitiva. Questo consentirebbe di lavorare con *grafi delle relazioni*, che abbiano un arco orientato in corrispondenza di ogni coppia di AS per la quale sia possibile affermare l'esistenza di una relazione. Di conseguenza, sarebbe possibile analizzare più puntualmente le variazioni che hanno avuto luogo durante le due settimane "turbolente".

- **Exchange points** — Molti Autonomous Systems, piuttosto che collegarsi direttamente con tutti i vicini da loro scelti, preferiscono più efficacemente instaurare un'unica sessione di peering con un *exchange point*, cioè una struttura in cui molti AS sono collegati tra loro, instaurando rapporti commerciali di vario tipo. La presenza di numeri identificativi di Autonomous Systems relativi a queste infrastrutture potrebbe alterare il comportamento dell'algoritmo di inferenza, e risulterebbe quindi utile conoscere un sistema per eliminarli preliminarmente in modo automatico.

- **Gradi di libertà** — L'algoritmo di inferenza utilizzato assegna le orientazioni sulla base di un ordinamento topologico dei nodi del grafo dei letterali. In realtà, alcuni archi possono non essere necessariamente costretti ad assumere una certa orientazione: il fatto che gli sia stata assegnata può

[1] In realtà, ad essere molti sono gli archi che mantengono inalterata la propria orientazione *in tutti gli snapshot in cui sono visibili.*

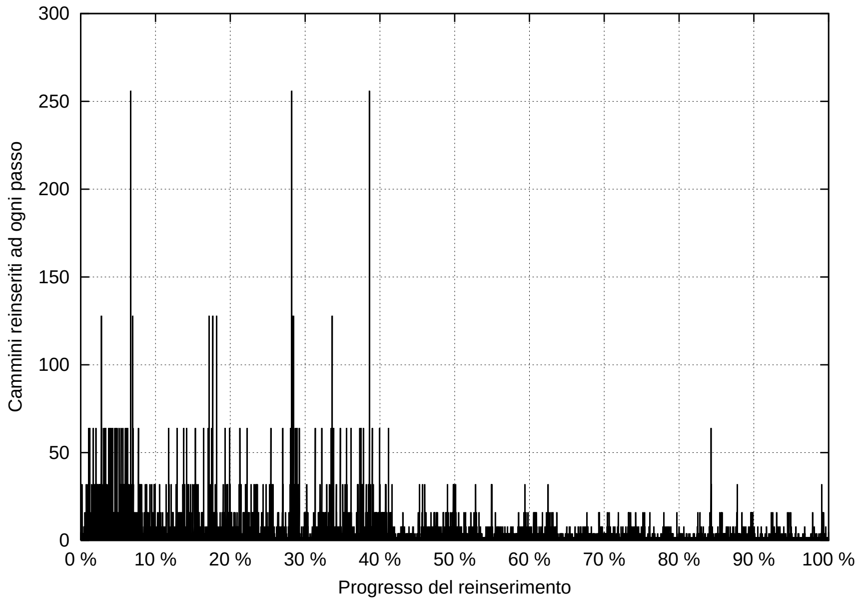

Figura 6.3: Andamento del processo di reinserimento dei *bad paths* per lo snapshot del 18 Aprile 2001 disponibile in [9], strategia `chunk*=2 | chunk=1`.

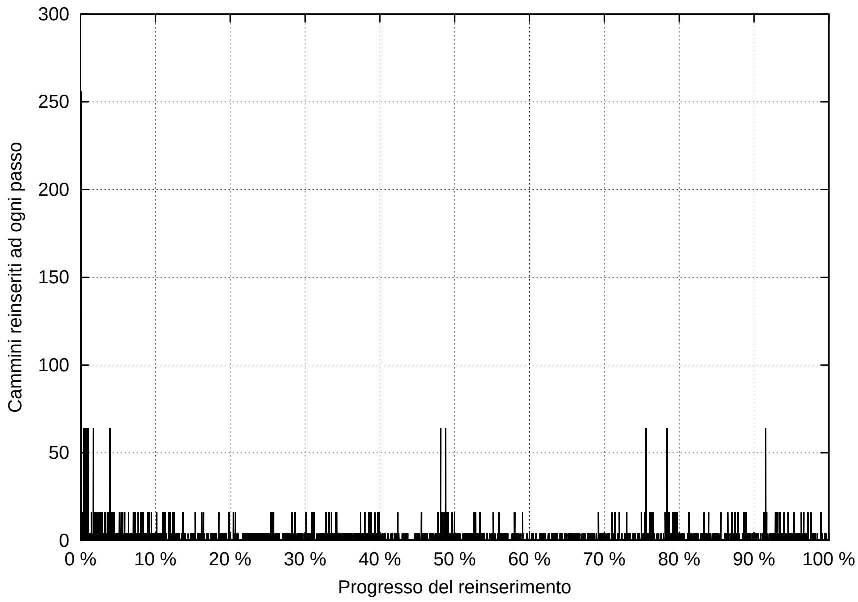

Figura 6.4: Andamento del processo di reinserimento dei *bad paths* per lo snapshot del 25 Marzo 2003 disponibile nell'archivio di Route Views ([22]), strategia `chunk*=4 | chunk/=4`.

essere semplicemente conseguenza del modo in cui sono stati ordinati i nodi (tra l'altro, anch'esso non univoco). Risulterebbe dunque interessante valutare, in ogni istanza, quanti archi siano effettivamente vincolati ad assumere una certa orientazione e quanti, invece, possano essere orientati arbitrariamente[2].

- **Altri tipi di rapporto** — L'algoritmo di inferenza analizzato non consente di ricavare relazioni di tipo diverso da quella customer-provider. Evidentemente, sarebbe estremamente utile poterlo estendere alla ricerca di peering, backup e sibling. In [20] viene anche illustrato come sia possibile alterare un'orientazione assegnata in precedenza per introdurvi archi di peering. L'implementazione disponibile presso [23] mette anche a disposizione questa funzionalità. Tuttavia, potrebbe essere utile approfondire il problema per ricavare queste nuove relazioni con un buon grado di affidabilità (per esempio, sulla base della ricerca di configurazioni notevoli negli AS path: le anomalie di tipo 4 –figura 2.4– sono in [8] considerate possibile indice della presenza di un rapporto di sibling). A tal fine, potrebbe anche essere necessario basarsi su informazioni non presenti nei soli AS path (per esempio, sugli spazi di indirizzamento annunciati in ogni direzione).

- **Best practice** — In tutti gli approcci presentati per il problema dell'inferenza delle relazioni tra Autonomous Systems si assume sempre che le politiche di routing adottate per ogni tipo di relazione siano più o meno sempre le stesse. Sarebbe utile effettuare un'esplorazione mirata a valutare quanto effettivamente questa pratica venga rispettata in situazioni reali.

- **Ulteriore ottimizzazione dell'efficienza** — Nella prima delle sperimentazioni presentate ci sono situazioni in cui la strategia di considerare più cammini in un unico tentativo di reinserimento si rivela particolarmente vantaggiosa. Essa consente, infatti, di dimezzare il numero di iterazioni necessarie per completare la costruzione dell'insieme massimale di cammini. La stessa cosa non accade, invece, sui dati di Route Views e con la versione modificata di tale strategia. Potrebbe essere interessante capire quale sia il modo migliore di calcolare il numero di cammini candidati al reinserimento successivo, per tentare di ottenere le massime prestazioni anche da questa fase dell'algoritmo di inferenza. Ad esempio, tale calcolo potrebbe essere guidato dalla distribuzione della probabilità di successo di un reinserimento. Le figure 6.3 e 6.4 mostrano l'andamento del processo di reinserimento in due casi in cui la strategia è stata, rispettivamente, più o meno efficace.

[2] In realtà, gli archi di quest'ultimo tipo possono assumere un'orientazione arbitraria senza introdurre cammini invalidi soltanto se considerati singolarmente: il fatto di cambiare orientazione ad un arco, infatti, modifica molto probabilmente le scelte fatte per gli archi circostanti, il che non rende possibile l'assegnazione completamente arbitraria di un'orientazione a più di un arco.

Bibliografia

[1] Internet Assigned Numbers Authority,
http://www.iana.org

[2] Asia Pacific Network Information Centre,
http://www.apnic.net

[3] American Registry for Internet Numbers,
http://www.arin.net

[4] Latin American and Caribbean Internet Addresses Registry,
http://lacnic.net

[5] Réseaux IP Européens Network Coordination Centre,
http://www.ripe.net

[6] Y. Rekhter, T. Li, *A Border Gateway Protocol 4 (BGP-4)*, RFC 1771, Marzo 1995.

[7] Lixin Gao, Timothy G. Griffin, Jennifer Rexford, *Inherently Safe Backup Routing with BGP*, Aprile 2001.

[8] Lakshminarayanan Subramanian, Sharad Agarwal, Jennifer Rexford, Randy H. Katz, *Characterizing the Internet Hierarchy from Multiple Vantage Points*, Aprile 2001.

[9] Dati utilizzati in [8] (tabelle BGP e relazioni calcolate),
http://www.cs.berkeley.edu/~sagarwal/research/BGP-hierarchy/

[10] Lixin Gao, *On Inferring Autonomous System Relationships in the Internet*, Novembre 2000.

[11] Implementazione dell'algoritmo presentato in [10],
http://www-net.cs.umass.edu/~ratton/AS/

[12] William B. Norton, *Internet Service Providers and Peering*, Maggio 2001.

[13] Zihui Ge, Daniel R. Figueiredo, Sharad Jaiswal, Lixin Gao, *On the Hierarchical Structure of the Logical Internet Graph.*

[14] Geoff Huston, *Interconnection, Peering and Settlements, part I and II*, Gennaio 1999.

[15] Geoff Huston, *Interconnection and Peering*, Ottobre 2000.

[16] Anwar M. Haneef, Santhosh R. Thampuran, *Verification of Sibling-Sibling Relationship Between Autonomous Systems in the Internet.*

[17] Timothy G. Griffin, F. Bruce Shepherd, Gordon Wilfong, *The Stable Paths Problem and Interdomain Routing.*

[18] Lixin Gao, Jennifer Rexford, *Stable Internet Routing Without Global Coordination*, Giugno 2000.

[19] Timothy G. Griffin, Gordon Wilfong, *A Safe Path Vector Protocol*, Marzo 2000.

[20] Giuseppe Di Battista, Maurizio Patrignani, Maurizio Pizzonia, *Computing the Types of the Relationships between Autonomous Systems*, Proceedings of IEEE INFOCOM 2003.

[21] Thomas Erlebach, Alexander Hall, Thomas Schank, *Classifying Customer-Provider Relationships in the Internet*, Luglio 2002.

[22] University of Oregon Route Views project, archivio tabelle di routing in formato *show ip bgp*,
`http://archive.routeviews.org/oix-route-views/`

[23] Università degli Studi Roma Tre, *Computer Networks Research Group*,
`http://www.dia.uniroma3.it/~compunet/`

[24] `http://www.internetnews.com`

[25] `http://www.isp-planet.com`

[26] Sito web di Geoff Huston, `http://bgp.potaroo.net`

[27] Ramesh Govindan, Anoop Reddy, *An Analysis of Internet Inter-Domain Topology and Route Stability*, Aprile 1997.

[28] Hyunseok Chang, Ramesh Govindan, Sugih Jamin, Scott J. Shenker, Walter Willinger, *On Inferring AS-Level Connectivity from BGP Routing Tables.*

[29] Cengiz Alaettinoğlu, *Scalable Router Configuration for the Internet.*

[30] Hongsuda Tangmunarunkit, Ramesh Govindan, Scott Shenker, Deborah Estrin, *The Impact of Routing Policy on Internet Paths.*

[31] Bill St. Arnaud, *Issues on Peering, Transit and Cost Sharing International Internet Networks,* DRAFT, Novembre 1999.

[32] Daniel O. Awduche, Johnson Agogbua, Jim McManus, *An Approach to Optimal Peering Between Autonomous Systems in the Internet.*

[33] Pio Baake, Thorsten Wichmann, *On the Economics of Internet Peering*, Gennaio 1998.

[34] William B. Norton, *Interconnection Strategies for ISPs*, Maggio 1999.

[35] R. Chandra, P. Traina, T. Li, *BGP Communities Attribute*, RFC 1997, Agosto 1996.