Algoritmo per effettuare immunizzazione di una rete mediante d-covering

Alessio Cardillo
http://www.ct.infn.it/~cardillo
Alessio.Cardillo@ct.infn.it

1 Introduzione

In questo (non saprei esattamente come chiamarlo) mini how-to, mi appresto a descrivere quali sono le operazioni da compiere per effettuare il d-covering di un network (pesato o no). In particolare presenterò una prima versione dell'algoritmo che si adatta meglio ai grafi piccoli ed in seguito ne presenterò una seconda più adatta a quelli grandi e che può essere eseguita su macchina poco performanti (come il mio portatile ad esempio ;-)).

2 d-covering

Il d-covering è una tecnica di immunizzazione di reti che si trova a "metà strada" tra l'immunizzazione random e quella targeted. La prima non presuppone conoscenza alcuna della rete su cui si effettua l'immunizzazione mentre la seconda presuppone una conoscenza *COMPLETA* della rete stessa. Il d-covering invece richiede una conoscenza **locale** della rete. Per chi volesse approfondire @@@@@ Articolo Moreno PRE @@@@@ . In sostanza si tratta di andare a considerare di volta in volta i vicini posti entro una certa distanza d da un certo nodo e poi immunizzarne uno mediante un criterio precedentemente definito. In questo modo è possibile immunizzare la rete rendendo un certo numero di nodi non suscettibili all'infezione. Tali nodi prendono il nome di "guardie".

3 Algoritmo di d-covering

L'algoritmo sotto riportato è valido sia per le reti pesate che per quelle non pesate. In genere io mi riferirò a quelle pesate. Supponiamo di avere una rete immagazzinata sotto forma di matrice di adiacenza oppure di lista di adiacenza.

3.1 Operazioni preliminari

Prima di poter procedere con il d-covering è necessario disporre di alcune informazioni.

- la strenght ed il grado di ogni singolo nodo S_i & k_i ;
- Il grado massimo k_max;

- Conoscere i vicini di ogni nodo;

In particolare mentre il secondo è un parametro fissato della rete, gli altri due gruppi di dati vanno conservati su file. Mentre il primo punto è facilmente realizzabile anche nel caso di reti grandi, il terzo lo è assai meno. In seguito affronterò più in dettaglio questa problematica e le eventuali modalità di risoluzione. Una volta note queste quantità è possibile procedere con il d-covering.

4 d-covering

L'algoritmo di d-covering si basa sulle seguenti operazioni:

1. Si estrae un nodo a caso (nodo start) ;
2. Si prendono tutti i d-vicini del nodo considerato ;
3. Tra tutti i d-vicini si cerca quello con strenght massima ;
4. Si identifica il nodo come "guardia" ;
5. Si considerano adesso tutti i d-vicini della guardia ;
6. Si identificano tutti i d-vicini della guardia come *coverati*, facendo attenzione però a non indicare come coverati eventuali altre guardie ;
7. Si procede così (ripetendo le operazioni precedenti) fino a quando tutta la rete è composta esclusivamente da nodi coverati e da guardie.

In particolare, il numero massimo di vicini inizialmente (per grafi piccoli) è stato posto uguale a quello del Caley Tree @@@ Ref su Caley Tree @@@@ . In realtà anche su questo punto è possibile fare delle distinzioni tra grafi piccoli e grandi. Per tenere conto dello stato di covering dei nodi invece, basta considerare un semplice vettore di label.

5 Grafi grandi

Finora non si è fatto in alcun modo accenno alle dimensioni dei grafi. In realtà però quando ho dovuto poi calcolare effettivamente il d-covering su delle reti sono andato a cozzare violentemente contro la dura realtà :-) . Il problema è semplicemente uno: $ALLOCARE\ LA\ MEMORIA$. Infatti, al crescere delle dimensioni del grafo, sorgono diversi problemi che vediamo di affrontare di seguito:

1. **Problema** L'array dei vicini per un grafo di 100000 nodi circa con grado massimo pari a 20000 circa se allocato sotto forma di matrice (1000000×20000) diventa incredibilmente grosso e non è facile (ma non impossibile) allocarlo.
   Soluzione È possibile superare questo problema in due maniere:

(a) Si può dividere l'array in tanti sotto array (per esempio una decina) ogniuno dei quali sarà di dimensione $100000 \times k_{max}^{loc}$, dove k_{max}^{loc} non è altro che il grado massimo di quel gruppo di nodi. Questo sistema (che è quello che io ho adottato per fare i miei conti) richiede però una complicazione del codice in quanto il programma deve essere in grado di capire da che file andare a leggere di volta in volta per trovare i vicini di un certo nodo.

(b) Si può allocare il file dei vicini utilizzando un cosiddetto *array frastagliato*, ovvero un array di stringhe, ciascuna delle quali ha lungezza variabile (pari al grado di quel nodo). Questo sistema è decisamente più efficiente del precedente in quanto necessita solo della memoria effettivamente richiesta per allocare l'array senza bisogno di sprecarne nemmeno un pò. Tuttavia, per potersene avvalere è necessario utilizzare il concetto di puntatore che non è presente in tutti i linguaggi (come per esempio nel Fortran). Infatti l'array frastagliato non è altro che un vettore di puntatori in cui ciascun elemento contiente il puntatore al vettore di lunghezza pari al grado del nodo considerato, contenente i vicini di quel nodo.

2. **Problema** Utilizzando come numero massimo di nodi d-vicini quello di un Caley Tree con grado pari a k_{max} elevato per la distanza d ovvero:

$$N_{d-vicini} = (k_{max} + 1)^d$$

si ha che per reti con k_{max} circa 20000 appena di va d = 3 si ottiene $N_{d-vicini} = 8 \cdot 10^{12}$!!!!! Questo è evidentemente irragionevole perché non sarebbe possibile fare covering con $d \geq 3$.
Soluzione In realtà il numero massimo di nodi d-vicini di un nodo non è altro che $N - 1$ dove N è il numero totale di nodi. Basta quindi predisporre un array di tali dimensioni associato ad un array di label che serva ad evitare che un nodo sia messo nella lista di d-vicini più di una volta ed il gioco è fatto!!!!

6 Conclusioni

Queste problematiche hanno permesso di evidenziare i difetti del codice sviluppato per grafi piccoli consentendo di implementare il codice facendogli raggiungere un grado di maturità superiore. Sino a questo momento non ho riscontrato altri problemi ma qualora qualcuno ne trovi altri può tranquillamente segnalarmeli.