Tecniche distribuite per *trust management* in WSN

Chesini Nicola
Dipartimento di Ingegneria dell'Informazione
Universitá degli Studi di Padova
E-mail: nicola.chesini@gmail.com

Ermon Stefano
Dipartimento di Ingegneria dell'Informazione
Universitá degli Studi di Padova
E-mail: ermonste@gmail.com

Marcolin Giampietro
Dipartimento di Ingegneria dell'Informazione
Universitá degli Studi di Padova
E-mail: giampietro.marcolin@studenti.unipd.it

***Sommario*—La recente crescita esplosiva del *networking* in ambito sia industriale che civile, resa possibile soprattutto dagli enormi progressi nelle comunicazioni wireless, ha reso obsoleto il tradizionale modello di rete statica gestita in modo centralizzato. Nuovi tipi di rete come *wireless sensor networks*, mobile ad hoc networks, P2P networks, dotate di architetture distribuite e *self-organizing* stanno conquistando un ruolo sempre più importante.**

In questo contesto, il *trust management* si sta imponendo come un approccio fondamentale per la difesa di questo tipo di reti da attacchi esterni. Inoltre può essere molto utile nel *decision-making* di vari protocolli e per stimolare la collaborazione tra le entità che compongono la rete.

In questo lavoro proponiamo un'analisi teorica delle prestazioni di tecniche di *trust management* che in letteratura sono state testate solo tramite simulazione, trovando risultati sorprendenti. Inoltre, con l'aiuto di una analisi simulativa più approfondita, abbiamo generalizzato gli approcci esistenti raggiungendo una migliore comprensione delle dinamiche che si instaurano. Infine studiamo quali risorse e quali funzioni dei protocolli di comunicazione siano necessarie ai nodi componenti la rete per poter implementare queste tecniche in modo realmente distribuito, introducendo anche metodi di *reputation propagation* che potrebbero rivelarsi molto utili in una implementazione reale.

I. Introduzione

La recente crescita esplosiva del *networking* in ambito sia industriale che civile è stata resa possibile soprattutto dagli enormi progressi nelle comunicazioni wireless. Anche grazie a queste nuove tecnologie, le moderne reti vengono utilizzate in tantissimi campi e in molteplici applicazioni. Si va dall'uso ormai consolidato per quanto riguarda la connessione ad Internet degli utenti a usi per controllo di processi, dai telefoni cellulari al controllo degli elettrodomestici in casa, e all'uso in un'infinità di applicazioni il cui numero é destinato ad aumentare . Uno dei settori in cui lo sviluppo è stato maggiore è quello delle reti di sensori, soprattutto wireless (WSN , *Wireless Sensor Network*) in cui svariati tipi di sensori (temperatura, pressione, posizione, forza, ecc...) comunicano fra loro attraverso comunicazioni radio. Si ha bisogno quindi di una rete che permetta lo scambio dell'immensa, per non dire astronomica, quantità di dati e di informazioni. Il tradizionale sistema di gestione centralizzato, oltre ad avere bisogno di un'infrastruttura a volte difficile da costruire, molto spesso non riesce più a sopportare queste imponenti moli di dati e, anche quando é in grado di superare questo problema, si ottengono prestazioni abbastanza scarse.

Risulta perció conveniente decentralizzare il più possibile le operazioni, dividendo così la complessità sui vari dispositivi. Faremo alcune considerazioni per le reti di sensori, ma la cosa si potrebbe estendere alle *mobile ad hoc networks* (MANETs), *P2P networks* e a tutte le reti in cui i servizi non si basano su architetture *Client-Server*. Una situazione in cui

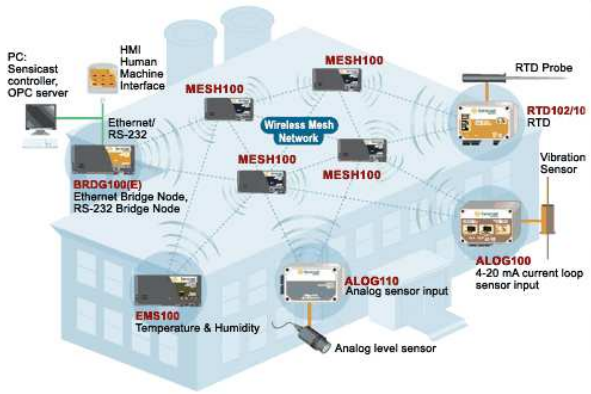

Figura 1. Esempio di rete di sensori wireless.

puó essere vantaggioso usare un sistema decentralizzato si ha, ad esempio, quando si vuole eseguire una media fra varie misure di temperatura. In tal caso non c'è bisogno di trasmettere tutti i dati ad uno stesso elaboratore ma si possono fare medie locali e poi mettere assieme i valori per ottenere lo stesso risultato.

Alla base di questa idea, però, c'è il bisogno di creare nuovi standard e protocolli che permettano alla rete di funzionare correttamente senza un controllore centralizzato. Infatti, in queste tipologie di rete, tutti gli utenti sono paritari, e quindi manca un ente supervisore in grado di decidere per tutti, da cui il bisogno di regole comuni ben definite. Non sorprende quindi che questo settore di ricerca sia tuttora molto attivo come mostrano [11] e [19].

Uno dei problemi che rimangono maggiormente aperti è quello della sicurezza, soprattutto nelle reti wireless, in cui chiunque si può mettere in ascolto delle comunicazioni e cercare di interferire. Infatti viene a mancare il vincolo del cablaggio e si perde quindi il controllo fisico su chi puó connettersi alla rete.

Un approccio per risolvere questo problema é quello di usare tecniche di crittografia volte ad impedire accessi indesiderati alla rete. Dato il notevole dispendio computazionale di queste metodologie, esse non appaiono adatte per le WSN dove

la capacitá di elaborazione é solitamente limitata e la prioritá viene data al risparmio energetico che é determinante per il tempo di vita del dispositivo [12].

Nell'ambito dei problemi di sicurezza, in questo lavoro ci occuperemo del problema del *trust management*, ovvero dei meccanismi grazie ai quali le entitá che compongono la rete sanno se e quanto possono fidarsi le une delle altre. In questo modo, ad esempio, possono evitare di compiere azioni ad alto rischio, come chiedere a nodi poco affidabili di inoltrare pacchetti o aggregare dati. Cosí facendo, la robustezza e le prestazioni della rete crescono. Inoltre questi meccanismi sono un incentivo alla cooperazione. Infatti le entitá della rete si comportano in modo piú responsabile sapendo che la loro reputazione viene danneggiata da azioni egoistiche.

Trattando l'argomento relativamente a reti di sensori wireless, le tecniche usate dovranno essere distribuite e bisognerà tener conto anche della capacità di calcolo, che a volte può essere piuttosto limitata. Finora, nell'ambito della ricerca, gli unici tentativi che sono stati fatti per risolvere il problema del *trust management* in modo distribuito e con un approccio analitico sono dovuti al gruppo guidato da John S. Baras all'Universitá del Maryland. Come si vede in [1] e in [2], egli affronta il problema ispirandosi a un modello proprio della fisica statistica: il modello di Ising. Quest'ultimo affonda le sue radici nella descrizione del comportamento degli *spin* in materiali dotati di proprietá magnetiche, come il ferro, e cerca di spiegare come e perchè essi si allineino in determinati modi attraverso delle iterazioni locali. Recentemente il modello di Ising è stato applicato anche in situazioni che, pur essendo nettamente diverse rispetto a quella per cui è stato pensato, presentano delle dinamiche locali. In altre parole il modello di Ising sembra descrivere bene tutte quelle situazioni in cui vi sono delle singole particelle che interagiscono con quelle vicine dando origine a proprietà globali attribuibili al sistema complessivo. Alcuni esempi sono il modello della memoria associativa per reti neurali e la cooperazione in reti sociali.

Il parallelismo tra il problema di *trust management* distribuito e il modello di Ising è dovuto al fatto che l'affidabilità di un nodo (buono-cattivo) può essere associata all'orientazione dello *spin* (up-down). Inoltre quest'ultima è determinata in base a interazioni con gli *spin* vicini secondo una legge di cui il modello di Ising tiene conto. In [1], dopo aver introdotto una regola di interazione locale, si studia il comportamento globale della rete sfruttando la teoria del modello di Ising. In particolare si eseguono delle votazioni a livello locale sull'affidabilitá di un nodo *target* a cui partecipano i soli vicini. Sfruttando il parallelismo con la fisica statistica si riesce a determinare quanti nodi, nel complesso, siano stati valutati correttamente, stabilendo cosí le prestazioni del sistema di *trust management* distribuito implementato in questo modo. Analizzando nel dettaglio i risultati riportati in [1], abbiamo notato delle discrepanze rispetto a un teorema da noi formulato in merito al problema in esame, e questo ci ha posto dei dubbi sulla reale efficacia dell'algoritmo utilizzato. Un esame piú approfondito ci ha permesso di capire che, con una scelta dei parametri diversa da quella fatta in [1], si ottengono delle buone prestazioni. Abbiamo poi affrontato il problema della verifica di questi risultati ottenuti per via teorica che, essendo asintotici, non sempre vengono effettivamente raggiunti. In particolare all'aumentare delle dimensioni della rete tale problema diventa sempre piú significativo. Poichè nella realtà una rete di sensori è spesso costituita da molti nodi, ci è sembrato interessante lavorare in questa direzione, continuando ad applicare lo stesso algoritmo e osservando i risultati ottenuti per via simulativa. Sulla base di tali risultati si può affermare che l'algoritmo di *trust management* usato funziona bene anche in questi casi, anche se dipende fortemente dalla scelta dei parametri in gioco. Ad un'analisi statica del modello costruito, ottenuta riprendendo e approfondendo alcune problematiche trattate anche in [1], è seguita un'analisi dinamica, in cui si sono considerate anche questioni come la presenza di ritardi aleatori o perdita di pacchetti nella rete. Infine ci si è chiesti quali risorse e quali funzioni dei protocolli di comunicazione siano necessarie per implementare l'algoritmo in modo realmente distribuito. A tal proposito, prendendo spunto da [5], abbiamo preso in considerazione tecniche di propagazione dell'informazione nella rete basate sulla teoria del consenso. Non siamo riusciti a integrare due approcci cosí diversi, tuttavia questi metodi si sono rivelati una possibile soluzione a sé stante del problema di *trust management*. La tecnica proposta in [5] é piú facilmente implementabile da un punto di vista pratico ma le sue prestazioni risultano in molti casi inferiori.

Il lavoro è organizzato nel seguente modo. Nella sezione II-A si tratta lo studio analitico di un algoritmo per il *trust management*, nella II-B un modello di rete, mentre nella II-C viene descritto il modello di Ising. Quanto riportato nella sezione II, cosí suddivisa, verrá ampiamente utilizzato nel lavoro che segue.

Nella sezione III vengono mostrati alcuni risultati asintotici ottenuti per via teorica, alcuni dei quali sono in contrasto con [1]. In questa sezione viene anche discusso il problema di una verifica sperimentale dei risultati ottenuti che viene poi svolta su reti di piccole dimensioni. Per l'uso su reti di grandi dimensioni, che generalmente si trovano nella realtá, vengono effettuate varie prove i cui risultati sono mostrati nella sezione IV.

Trovare un modello che coincida con la realtá é molto difficile perció nella sezione V si introducono varie generalizzazioni (ritardi, perdita di pacchetti,ecc.) per far assomigliare sempre piú il modello utilizzato ad una *wireless sensor network*. Ci si chiede poi se le ipotesi utilizzate fino a questo momento rendano l'algoritmo effettivamente implementabile in pratica, domanda a cui si cerca di dare una risposta nella sezione VI. Prendendo spunto da [5], nella sezione VII si descrive una tecnica di *trust management* basata sulla propagazione dell'informazione. Infine in VIII vengono derivate alcune conclusioni sui risultati ottenuti proponendo anche qualche idea per lavori futuri.

II. Stato dell'arte

In questa sezione vengono descritti alcuni risultati presenti in letteratura e utili per il nostro lavoro. Sono proposti un algoritmo per il trust management e dei modelli di rete che verranno utilizzati in seguito.

A. *Un algoritmo per il* trust management.

In [1] gli autori analizzano procedure e regole attraverso cui i vari nodi della rete possono ottenere una valutazione dell'affidabilitá degli altri nodi. Per valutazione dell'affidabilitá di un nodo intendiamo il fatto di giudicare se esso é buono o cattivo.
Le procedure che vengono descritte non si occupano mai di come l'informazione si propaga attraverso la rete, e vedremo che questo risulterà uno dei limiti principali alla loro applicazione.
Diamo nel seguito una descrizione dell'algoritmo e dei risultati forniti in [1], che costituiscono il punto di partenza per il nostro lavoro.
La rete viene modellizzata come un grafo diretto $G(V, E)$ privo di *self-loop* in cui i vertici del grafo sono i nodi della rete e gli archi rappresentano la relazione di *trust* fra i nodi. Il grafo di trust G può differire da quello fisico perchè non è detto che se è presente un collegamento per la trasmissione di informazione fra due nodi ci sia anche una reciproca relazione di *trust*. Si suppone che i nodi della rete siano $|V| = N$ e che siano etichettati con un indice $V = 1, ..., N$. Inoltre se un nodo é *buono* lo si indica ponendo $t_i = 1$, se é *cattivo* ponendo $t_i = -1$. Il vettore risultante $T = [t_1, ..., t_N]$ è detto vettore reale di *trust*, poiché tiene conto dello stato di affidabilitá reale di tutti i nodi. Viene definito anche un vettore $S = [s_1, ..., s_N]$ che rappresenta la stima di T. L'obiettivo finale risulta quindi quello di determinare S in maniera corretta ($S = T$).
Per calcolare S si hanno a disposizione dei valori di confidenza c_{ij}, cioè il voto che il nodo i dà al nodo j circa la sua affidabilitá. Supponiamo $c_{ij} \in \{-1, 1\}$ con $c_{ij} = 1$ se i ha confidenza in j e viceversa con -1. Il voto si ha solo se i nodi i e j comunicano fra di loro e quindi riescono a valutarsi direttamente, mentre in caso contrario si pone $c_{ij} = 0$.
Questa valutazione si può ottenere con varie tecniche che dipendono dalla particolare applicazione considerata, ma in generale i voti c_{ij} si ottengono osservando se i nodi j, vicini di i, si comportano correttamente (es. inoltrano i pacchetti). Se pensiamo ad una applicazione in WSN, ciò significa valutare se i propri vicini inoltrano i pacchetti quando viene loro richiesto, oppure, nel caso in cui i e j misurino la stessa grandezza, segnalare differenze significative nelle stime ([20], [21]).
In contesti diversi, come le reti di *e-commerce*, il c_{ij} rappresenta sostanzialmente il feedback dell'utente i sull'utente j relativo ad una transazione di natura commerciale avvenuta tra i due ([22],[23]).
Infine, per quanto riguarda le MANETs, le valutazioni che si danno i *peers* sono soprattutto relative all'affidabilità nell'inoltrare pacchetti, ma spesso tengono anche conto se l'accesso al mezzo di comunicazione condiviso avviene in modo responsabile ([25],[24]).
In questo lavoro non considereremo in particolare nessuna di queste tecniche, ma ci limitiamo a considerare una generica politica di attribuzione del voto c_{ij} da parte di un nodo i buono che ammetta una certa probabilità di errore nella stima, definita come

$$p_e = \mathbb{P}\left[c_{ij} \neq t_j | t_i = 1\right] \quad , \ \forall \ i \text{ buono.}$$

Si tratta in sostanza di una probabilità d'errore che può essere commessa nell'attribuzione di un voto da un nodo buono ai suoi vicini, che andrà valutata di volta in volta a seconda della tecnica di stima usata nella particolare applicazione. Naturalmente bisogna tenere presente che la rete è in generale formata sia da nodi buoni che da nodi cattivi, il che complica notevolmente le cose in quanto bisogna decidere come questi ultimi assegnino i voti. In prima battuta si possono prendere in considerazione tre possibilità:

- *voti* ***indipendenti***: i nodi cattivi si comportano indipendentemente gli uni rispetto agli altri e in modo opposto rispetto a quelli buoni, ovvero essi votano 1 un nodo cattivo e -1 un nodo buono, i.e. $c_{ij} \neq t_j, \forall (i, j) \in E$, a meno di una probabilità d'errore p_e definita come

$$p_e = \mathbb{P}\left[c_{ij} = t_j | t_i = -1\right] \quad , \ \forall \ i \text{ cattivo;}$$

- *voti* ***contrastanti***: i nodi cattivi si conoscono a vicenda, e votano *sempre* con $c_{ij} \neq t_j, \forall (i, j) \in E$;
- *voti* ***random***: i nodi cattivi assegnano voti *random* ai vicini, senza alcun criterio logico.

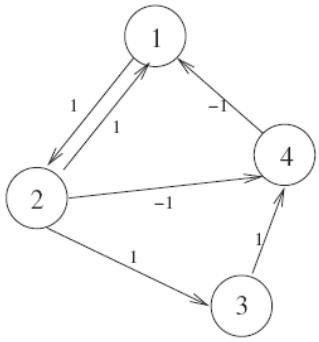

Figura 2. Esempio grafo con i voti di confidenza.

Il principio che sta alla base dell'algoritmo proposto in [1] é una regola di voto, detta *Local Voting Rule*, che consiste nel mettere assieme i valori di confidenza di tutti i vicini di un determinato nodo target tenendo conto anche del valore di fiducia di ogni nodo che partecipa alla votazione.
Si suppone di dover valutare la fiducia nel nodo target i. Per aggregare i voti dei suoi vicini si effettua una somma dei valori di confidenza pesati con i valori di *trust* stimati in S.

Dal momento che i valori di confidenza possono risultare in conflitto ($c_{ij} \neq c_{ji}$), si calcolano dei voti effettivi come

$$\widehat{c}_{ji} = c_{ji} + \alpha c_{ij}$$

dove poniamo $\alpha = 1$ come viene fatto anche in [1].

La valutazione del *trust* del nodo può essere considerata come un sistema dinamico che evolve nel tempo secondo la relazione

$$s_i(k+1) = f(\widehat{c}_{ji} s_j(k) | j \in N_i)$$

dove con N_i si indica l'insieme dei vicini di i.
Come funzione $f : \mathbb{R} \rightarrow \{-1, 1\}$ può essere scelta la funzione a soglia cosi descritta:

$$s_i(k+1) = \begin{cases} 1 & se \quad m_i(k) \geq \eta \\ -1 & se \quad m_i(k) < \eta \end{cases}$$

dove $m_i(k)$ è la somma pesata dei voti dei vicini di i

$$m_i(k) = \sum_{j \in N_i} \widehat{c}_{ij} s_j(k)$$

e η rappresenta la soglia per la decisione.
Poiché in una rete autonoma l'incertezza delle opinioni è inevitabile, in [1] si introduce dell'aleatorietà nella funzione di decisione tenendo comunque conto della somma pesata m_i. Si definisce quindi la regola di voto stocastica

$$\mathbb{P}[s_i(k+1) = 1 | m_i(k)] = \frac{e^{b(m_i(k)-\eta)}}{Z_i(k)}$$

$$\mathbb{P}[s_i(k+1) = -1 | m_i(k)] = \frac{e^{-b(m_i(k)-\eta)}}{Z_i(k)}$$

dove $Z_i(k)$ è il fattore di normalizzazione calcolato come

$$Z_i(k) = e^{b(m_i(k)-\eta)} + e^{-b(m_i(k)-\eta)}$$

e $b > 0$ è una costante che rappresenta il *grado di certezza* della rete: un valore piccolo di b rappresenta una grande incertezza e viceversa.
Combinando assieme le equazioni di decisione in un'unica formula possiamo scrivere:

$$\mathbb{P}[s_i(k+1) | m_i(k)] = \frac{e^{b s_i(k+1)(m_i(k)-\eta)}}{Z_i(k)} \qquad (1)$$

La regola di voto appena descritta è essenzialmente una funzione di aggiornamento per il vettore di *trust* S. Per specificare bene la regola, bisogna decidere in che sequenza effettuare l'aggiornamento delle varie componenti del vettore. Ci sono tre principali possibilitá:

- *aggiornamento sincrono*: i valori di *trust* di tutti i nodi sono aggiornati contemporaneamente;
- *aggiornamento asincrono ordinato*: i valori di *trust* sono aggiornati uno alla volta in sequenza ordinata;
- *aggiornamento asincrono random*: i valori di *trust* sono aggiornati uno alla volta e il nodo da aggiornare viene scelto casualmente.

Per una rete autonoma la tecnica di aggiornamento più semplice da implementare è quella di aggiornamento asincrono random (*Random asynchronous updates*). Per questo motivo essa verrá utilizzata in tutte le considerazioni che seguono, anche se i risultati valgono comunque anche per le altre tecniche, apportando solo poche modifiche all'algoritmo.
Definiamo q_i la probabilità che all'istante k il nodo i sia scelto come target per la valutazione, dove $\sum_{i \in V} q_i = 1$.
L'evoluzione del sistema, con le procedure descritte sopra, viene interpretata come una *catena di Markov* in cui lo stato è rappresentato dal vettore di *trust* S.
Vediamo quindi alcune proprietà di questa catena. Essendo l'aggiornamento di tipo asincrono ad ogni iterazione può cambiare una sola componente di S, come viene mostrato in Fig.3 per una rete con 4 nodi e per uno specifico stato di partenza. La probabilità di transizione dallo stato S allo

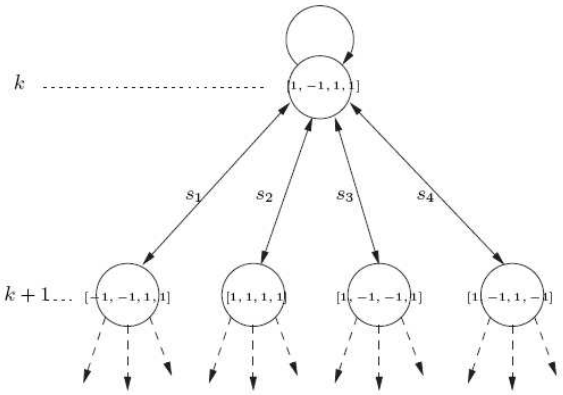

Figura 3. Catena di Markov.

stato $\bar{S}^i = [s_1, ..., \bar{s}_i, ..., s_N]$, dove $\bar{s}_i = -s_i$ indica il cambiamento del *trust* della componente i, è data da

$$p_{S,\bar{S}^i} = q_i \mathbb{P}[\bar{s}_i | S] = q_i \mathbb{P}[\bar{s}_i | m_i]$$

da cui deriva che la probabilità di rimanere nello stesso stato all'istante $k+1$ (self-loop nel grafo) è

$$p_{S,S} = 1 - \sum_{i \in V} p_{S,\bar{S}^i}$$

Il grafo associato alla catena di Markov risultante è fortemente connesso, è composto da un numero finito (2^N) di stati, ognuno dotato di *self-loop*. Queste proprietà sono sufficienti ad affermare che *la catena di Markov è regolare* e quindi, a regime, esiste un'unica densità di probabilità ad essa associata. Perciò, la regola di voto, sotto le banali condizioni $b \in (0, \infty)$ e $q_i > 0 \ \ \forall i$, converge.
Per derivare la distribuzione stazionaria che caratterizza il sistema si introduce la nozione di *catena di Markov reversibile*.

Definizione: Una catena di Markov è reversibile se vale l'equazione di bilancio dettagliata (*detailed balance*

equation)

$$\pi_S p_{S,R} = \pi_R p_{R,S} \quad \text{per ogni } R, S \tag{2}$$

dove R, S sono dei generici stati e π_S è la probabilità di essere nello stato S a regime. In altre parole, per una catena di Markov reversibile, a regime il processo sembra lo stesso sia in un verso che in quello opposto. Forti di questa definizione, il seguente lemma fornisce una via per trovare la distribuzione a regime.

Lemma 1: se π é una distribuzione di probabilità che verifica l'equazione 2, allora π è l'unica distribuzione di probabilità stazionaria.

Dimostrazione: l'equazione (2) implica

$$\sum_S \pi_S p_{S,R} = \pi_R \sum_S p_{R,S} = \pi_R \quad \text{per tutti gli } R \tag{3}$$

e perciò π soddisfa l'equazione di bilancio della distribuzione stazionaria.

□

Interpretando il lemma in termini matriciali, se chiamiamo A la matrice di transizione della catena di Markov associata, è evidente che l'equazione (3) equivale ad affermare che il vettore

$$\pi = [\pi_1 \dots \pi_i \dots \pi_N]'$$

è un autovettore destro di A relativo all'autovalore 1. Infatti vale

$$A\ \pi = \begin{pmatrix} p_{1,1} & p_{2,1} & \cdots & p_{N,1} \\ \vdots & \vdots & & \vdots \\ p_{1,R} & p_{2,R} & \cdots & p_{N,R} \\ \vdots & \vdots & & \vdots \\ p_{1,N} & p_{2,N} & \cdots & p_{N,N} \end{pmatrix} \begin{pmatrix} \pi_1 \\ \vdots \\ \pi_R \\ \vdots \\ \pi_N \end{pmatrix} = \begin{pmatrix} \pi_1 \\ \vdots \\ \pi_R \\ \vdots \\ \pi_N \end{pmatrix}$$

Si definisce prima di tutto l'*energia* di una configurazione S come

$$U(S) = \sum_{(i,j)\in E'} (c_{ij} + c_{ji}) s_i s_j - \eta \sum_{i \in V} s_i$$

dove E' é l'insieme degli archi considerati ottenuto trasformando il grafo $G(V, E)$ in un grafo non diretto $G(V, E')$. Si introduce anche una distribuzione π sullo stato della catena di Markov come segue

$$\pi_S = \frac{e^{bU(S)}}{Z} \tag{4}$$

dove Z è la costante di normalizzazione, chiamata anche funzione di partizione con

$$Z = \sum_S e^{bU(S)}.$$

Si arriva quindi a scrivere il seguente teorema.

Teorema 1: Per la regola di voto stocastica definita dall'equazione 1 e usando la regola di aggiornamento asincrona, se $b \in (0, \infty)$ e $q_i > 0, \forall i \in V$ si ha che

1) con questa regola di voto si converge a regime a un'unica distribuzione stazionaria;
2) la distribuzione $\pi_S = \frac{e^{bU(S)}}{Z}$ è l'unica distribuzione stazionaria.

Dimostrazione: La convergenza è stata verificata sopra in base alla regolarità della catena di Markov. Per provare che π è la distribuzione stazionaria, dobbiamo verificare che π soddisfi l'equazione (2) in accordo con il lemma visto prima. Si considerano due stati adiacenti S e R nella catena di Markov, ad esempio $S = [s_1, \dots, s_i, \dots, s_N]$ e $R = [s_1, \dots, \bar{s}_i, \dots, s_N]$. Perciò nella transizione da S a R si ha solo l'inversione di s_i, mentre tutti gli altri nodi rimangono invariati. Quindi il valore m_i rimane lo stesso per entrambi gli stati essendo

$$m_i = \sum_{j \in N_i} (c_{ji} + c_{ij}) s_j.$$

Rimane quindi da verificare l'equazione (2), che diventa

$$\frac{e^{bU(S)}}{Z} p_{S,R} = \frac{e^{bU(R)}}{Z} p_{R,S}$$

o

$$\frac{p_{S,R}}{p_{R,S}} = \frac{e^{bU(R)}}{e^{bU(S)}}$$

Sappiamo che

$$\begin{aligned} \frac{p_{S,R}}{p_{R,S}} &= e^{b(\bar{s}_i - s_i) m_i + b\eta(s_i - \bar{s}_i)} \\ &= e^{b \sum_{j \in N_i} ((c_{ji}+c_{ij})\bar{s}_i s_j - (c_{ji}+c_{ij}) s_i s_j) + b\eta(s_i - \bar{s}_i)} \\ &= e^{b(U(R) - U(S))} \end{aligned}$$

Perciò l'equazione (2) è soddisfatta e quindi π è la distribuzione stazionaria della regola di voto.

□

Avendo derivato la distribuzione stazionaria è possibile calcolare la probabilità di stima corretta. Sia SS il vettore di *trust* S a regime. Quindi la probabilità di stima corretta, considerando sia i nodi buoni che quelli cattivi si può scrivere come:

$$P_{correct} = \mathbf{E}\left[1 - \frac{\| SS - T \|_1}{2N}\right] \tag{5}$$

dove

$$\| SS - T \|_1 = \sum_{i \in V} |SS_i - T_i|$$

è la somma degli errori. La $P_{correct}$ rappresenta quindi l'aspettazione, fatta rispetto a π_S, del numero di stime corrette e si può scrivere nella forma vista sopra in quanto $SS, T \in \{1, -1\}$.

B. Topologia della rete.

In questo paragrafo è affrontato lo studio della topologia di un grafo $G(V, E)$ che modellizza la rete di affidabilità. Particolare risalto viene dato al *WS model*, il quale implementa un algoritmo relativamente semplice con cui è possibile modellizzare una rete reale preservandone le proprietà fondamentali.

B.1) Proprietà di una rete reale.

Ovviamente, per capire se il modello è una buona approssimazione della realtà, è necessario fissare alcune proprietà che caratterizzano una rete reale. Prima di elencarne le fondamentali, però, è bene definire alcuni concetti della teoria dei grafi utili per comprenderle meglio.

Definizione: Si definisce **coefficiente di raggruppamento** di un grafo diretto, e si indica con C, la media dei $C_i, \forall i \in V$, determinati come

$$C_i = \frac{|E(\Gamma_i)|}{k_i(k_i - 1)}$$

dove Γ_i è l'insieme dei nodi vicini a i, $|E(\Gamma_i)|$ è il numero di archi esistenti tra tutti i nodi in Γ_i, e infine k_i è il numero di archi collegati a i, ovvero la cardinalità di Γ_i. In altre parole, dato un nodo i, il coefficiente C_i esprime il rapporto tra il numero di archi *esistenti* che uniscono nodi vicini a i, e il numero massimo *teorico* di archi costruiti in questo modo, e quindi esprime la misura in cui i nodi vicini al nodo i sono legati tra loro. Ovviamente se $C = 1$ il grafo è completo.

Definizione: Si definisce **diametro medio** di un grafo, e si indica con L, la media degli $L_{ij}, \forall i, j \in V$, ciascuno dei quali è calcolato come il numero di archi che si trovano percorrendo il tragitto *minimo* tra i nodi i e j.

Date queste due definizioni, si possono elencare le proprietà fondamentali di un grafo che modellizza una rete reale:

- coefficiente di raggruppamento elevato;
- diametro medio piuttosto piccolo;
- ogni nodo ha un numero k di nodi vicini che non è costante, ma che solitamente varia secondo una legge esponenziale $\mathbb{P}[k] \sim k^{-\gamma}$, dove $\mathbb{P}[k]$ è la probabilità che un nodo abbia k vicini e γ è una costante fissata secondo qualche criterio;
- il numero complessivo N di nodi del grafo è variabile nel tempo.

Quest'ultima proprietà deriva dal fatto che in una rete reale è possibile aggiungere o togliere le entità[1] connesse tra loro dalla rete stessa ogni qualvolta ve ne sia necessità. Solitamente si parla di struttura *small-world* quando un modello soddisfa le prime due proprietà, e di struttura *scale-free* quando vale almeno la terza.
Negli ultimi anni si è lavorato molto, nell'ambito della ricerca, per trovare un modello matematico, nella fattispecie un grafo, che avesse nel contempo tutte queste proprietà. Fino ad oggi i ricercatori che si sono avvicinati maggiormente a questo obiettivo sono Klemm ed Eguíluz, i quali nel 2002 formularono il *KE model*. Questo modello, ben descritto in [18], soddisfa appieno le prime tre proprietà, mentre la quarta vale solo in parte, poichè si suppone che il numero di nodi del grafo non possa diminuire nel tempo, ma al più solo crescere. L'algoritmo che implementa questa topologia tiene conto di un parametro, $\mu \in [0, 1]$, variando il quale è possibile passare dalla classe dei modelli *small-world*, con $\mu = 0$, a quella dei modelli *scale-free*. Per opportuni valori di μ è possibile ottenere la situazione intermedia in cui valgono tutte le proprietà elencate poc'anzi, tenendo però conto della considerazione fatta per l'ultima.
Nonostante il *KE model* sia sicuramente molto valido, è tuttavia piuttosto difficile da implementare, per cui molto spesso si utilizza un modello, appartenente alla classe *small-world*, che, oltre a non soddisfare la terza proprietà, non tiene conto nemmeno dell'ultima, i.e. il numero di nodi del grafo è costante e fissato a priori. D'altro canto, pur non rispettando tutte le proprietà precedentemente elencate, questo modello, formulato da Watts e Strogatz nel 1998 in [15], e noto in letteratura come *WS model*, risulta relativamente semplice e nello stesso tempo sembra essere in grado di simulare bene il funzionamento di una rete reale. Buona parte delle simulazioni e degli algoritmi presentati in seguito sono basati su tale modello, per cui nel prossimo paragrafo si cercherà di darne una descrizione sufficientemente esauriente.

B.2) Il WS model

La fortuna del *WS model* è dovuta sostanzialmente alla sua facile implementabilità e al fatto che esso riesce a descrivere bene anche reti legate all'ambito sociologico e biologico, oltre che a quello tecnologico.
L'idea che sta alla base di questa topologia consiste nell'interpolare due tipi di grafo: uno in cui i nodi sono disposti lungo un reticolo circolare di dimensione finita e sono collegati tra loro *ordinatamente* secondo un certo criterio, e uno in cui gli stessi nodi vengono uniti in modo *randomizzato*. In letteratura si parla rispettivamente di *regular lattice* e di *random graph*. Nel primo caso si ha un C elevato, ma anche un L elevato, mentre nel secondo avviene l'esatto contrario. L'algoritmo che implementa tale idea necessita dei parametri N, k e p_{rw}, dove N è il numero di nodi della rete, k è il numero di vicini di ciascun nodo, p_{rw} è la probabilità di *rewire*, il cui significato sarà più chiaro in seguito. Va notato che N e k sono parametri fissati a priori, e ciò è in accordo con quanto detto nel precedente paragrafo. Una volta assegnati dei valori a questi tre parametri, facendo attenzione che $N \gg k \gg \ln(N) \gg 1$ (per modellizzare una rete sparsa

[1] Nel caso specifico si tratta di sensori, ma il discorso vale in generale per qualunque tipo di componenti.

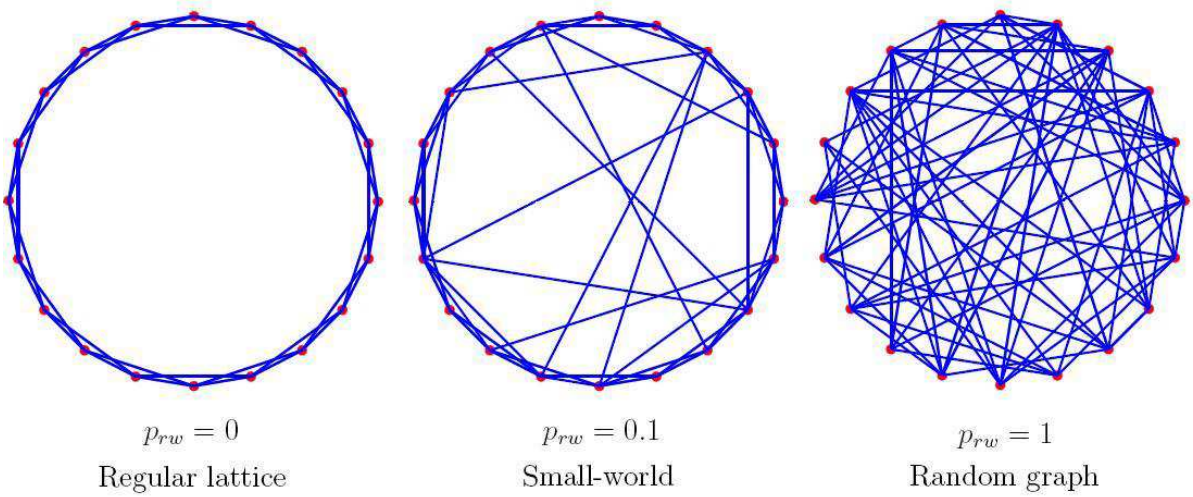

Figura 4. Grafi ottenuti per diversi valori di p_{rw}, con $N = 20$ e $k = 4$.

ma ben connessa) e che $0 \leq p_{rw} \leq 1$, si costruisce un grafo, che nel caso in questione sarà *diretto*, posizionando i nodi su un reticolo circolare di dimensione finita, i.e. una circonferenza se la dimensione è pari a 2, una sfera se è pari a 3, e cosí via. Ora, supponendo di essere nel caso bidimensionale, si connette ciascun nodo con i $\lceil k/2 \rceil$ vicini alla sua destra e i $\lfloor k/2 \rfloor$ vicini alla sua sinistra attraverso degli archi diretti. A questo punto si procede con l'operazione di *rewire*. Si prendono in considerazione gli archi che legano il nodo generico i al nodo $(i \pm 1)$ e con probabilità p_{rw} si ricollegano verso un altro nodo che va scelto secondo una certa densità di probabilità[2] sugli N nodi, evitando di costruire dei *self-loop* o degli archi doppi. Una volta fatta questa operazione $\forall i \in V$, si opera nello stesso modo con gli archi che legano il nodo i a $(i \pm 2)$, e così via, fintantochè tutti gli Nk archi vengono presi in considerazione. Ovviamente variando il parametro p_{rw} da 0 a 1 si ottengono tutte le situazioni intermedie tra il *regular lattice*, con $p_{rw} = 0$, e il *random graph*, con $p_{rw} = 1$, come si può vedere osservando i grafi disegnati in Fig.4, dove si è posto $N = 20$ e $k = 4$.

Per verificare quanto detto nel paragrafo precedente, ovvero che il *WS model* è in grado di descrivere un grafo con un coefficiente di raggruppamento C elevato e un diametro medio L piccolo, è sufficiente calcolare questi due parametri per diversi valori di p_{rw}. Il risultato ottenuto con $N = 1000$ e $k = 10$ è mostrato in Fig.5. In realtà, come chiarito dalla legenda, in Fig.5 sono disegnati gli andamenti di $C(p_{rw})$ e $L(p_{rw})$ normalizzati rispettivamente mediante i valori di C e L calcolati con $p_{rw} = 0$.

Al fine di apprezzare pienamente i risultati ottenuti, si è dovuto utilizzare una scala logaritmica di p_{rw}. Ciò è dovuto essenzialmente al fatto che il diametro medio L del grafo diminuisce fortemente per valori di p_{rw} molto piccoli (nel caso in questione per $p_{rw} < 0.1$), rimanendo poi all'incirca

[2]Nel caso in questione è stata scelta una ddp uniforme.

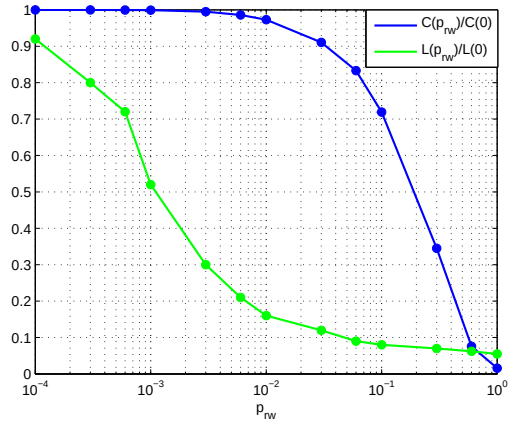

Figura 5. Andamento di $\frac{C(p_{rw})}{C(0)}$ e $\frac{L(p_{rw})}{L(0)}$, con $N = 1000$ e $k = 10$.

costante. Per spiegare tale risultato basta pensare che, ogni qualvolta un arco viene ricollegato, vengono avvicinati non solo i due nodi legati dal nuovo arco, ma anche i loro vicini, i vicini dei vicini, e così via. Per cui bastano poche operazioni di *rewire* per diminuire drasticamente il diametro medio del grafo.

Inoltre si nota che, mentre L cala, il coefficiente di raggruppamento C rimane pressochè costante e pari al valore ottenuto con $p_{rw} = 0$, per poi diminuire fino ad annullarsi con $p_{rw} = 1$. Ciò si spiega pensando al fatto che, al contrario di prima, per com'è definito C l'operazione di *rewire* su pochi archi non ne modifica molto il valore.

Ovviamente, volendo costruire un grafo con le proprietà di un modello *small-world*, si devono utilizzare i valori di p_{rw} per cui si hanno contemporaneamente un diametro medio piccolo e un coefficiente di raggruppamento elevato. Ad esempio, con $N = 1000$ e $k = 10$, ciò si ottiene ponendo $0.001 \leq p_{rw} \leq 0.1$. I grafici in Fig.5 confermano quanto affermato all'inizio di questo paragrafo, ovvero che il *WS*

model interpola due casi estremi combinandone in un certo modo le proprietà.

Si è voluto dare ampio spazio alla descrizione della topologia della rete perchè, come si vedrà in seguito, essa influisce tantissimo sulla prestazione della regola di voto definita nel paragrafo precedente.

C. Il modello di Ising.

Il modello di Ising è un modello matematico sviluppato dal fisico Ernst Ising nell'ambito della fisica statistica. Finora è stato utilizzato per modellare fenomeni in cui i bit di informazione, agendo tra loro a coppie, producono degli effetti collettivi. In particolare si definisce un insieme discreto di variabili $s_1, \ldots, s_N$ dette spin che possono assumere i valori 1 e -1, a volte chiamati stati *up* e *down*. I singoli valori degli spin e le coppie di spin interagenti tra loro determinano l'energia di una certa configurazione (ovvero un insieme di valori delle variabili s_i per $i = 1, \ldots, N$). In particolare si definisce energia o Hamiltoniana di una configurazione S

$$E(S) = -\sum_{i,j} J_{ij} s_i s_j - mH \sum_i s_i$$

Chiaramente due spin si dicono allineati se $s_i s_j = 1$, mentre in caso contrario sono disallineati. Gli spin i e j interagiscono tra loro in 3 modi differenti:

- $J_{ij} > 0$ l'interazione si dice ferromagnetica
- $J_{ij} < 0$ l'interazione si dice antiferromagnetica
- $J_{ij} = 0$ gli spin non interagiscono

Una interazione ferromagnetica tende ad allineare gli spin mentre una interazione antiferromagnetica tende a disallinearli. Il secondo termine della funzione energia rappresenta l'energia magnetica introdotta dalla presenza di un campo magnetico esterno, che spezza la simmetria del problema. La probabilità di una certa configurazione S è calcolata mediante la distribuzione di Bolztmann

$$\mathbb{P}[S] = \frac{e^{-\frac{1}{kT}E(S)}}{Z}$$

dove T rappresenta la temperatura assoluta, k la costante di Bolztmann, Z è detta funzione di partizione e serve per normalizzare al fine di ottenere una densità di probabilità. Questo modello è stato risolto nel caso monodimensionale da Ernst Ising nel 1925 nella sua tesi di dottorato, dove dimostrò che il modello non presenta alcuna transizione di fase (vedi [6]). Quando però si considera il caso di 2 o più dimensioni, il modello esibisce una transizione di fase, da uno stato ordinato a uno disordinato. Un classico esempio è quello dei materiali ferromagnetici, per studiare i quali il modello è stato originariamente pensato. Infatti si immaginava che il magnetismo di questi materiali fosse dovuto a un grande numero di spin atomici orientati nello stesso modo, e il modello di Ising serviva per capire se si potesse orientare un grande numero di spin nella stessa direzione usando solo forze locali. In questo specifico caso la transizione di fase rappresenta il passaggio da uno stato ordinato (per temperature basse, in cui il comportamento ferromagnetico è presente) ad uno stato disordinato in cui questo comportamento svanisce. Nel caso in cui il campo magnetico esterno sia nullo ($H = 0$) la temperatura critica a cui si presenta questo fenomeno, rappresentato in Fig.6, viene detta temperatura di Curie T_c.

Nel 2000 Sorin Instrail ha dimostrato che calcolare l'energia

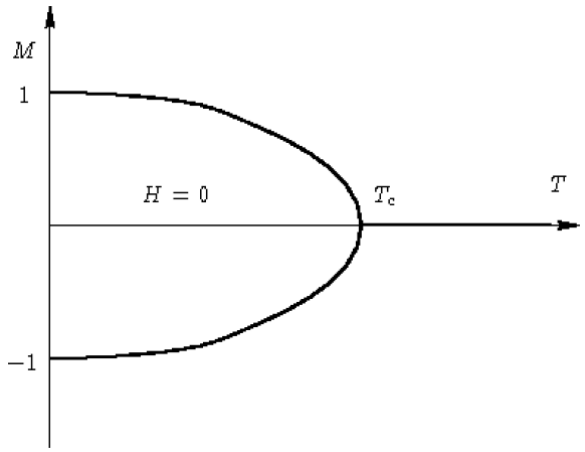

Figura 6. Transizione di fase nel modello ferromagnetico.

libera ($A = -kT\log(Z)$) di un *sottolattice* arbitrario di un *lattice* tridimensionale ricavato da un cubo è computazionalmente intrattabile. Questo significa che in generale per queste topologie è impossibile calcolare in modo efficiente tutte le possibili quantità termodinamiche del sistema, anche se ciò è stato fatto per alcuni casi particolari.

Il modello di Ising si è rivelato molto versatile ed è stato applicato con successo in altri campi, come nello studio della cinetica dei gas (*lattice gas*) e per modellare l'attività dei neuroni nel cervello. Recentemente questo tipo di approccio è stato usato nel campo dell'Ingegneria dell'Informazione per studiare grandi strutture molto complesse in cui però le interazioni tra le varie componenti sono essenzialmente locali.

L'analogia con la regola di voto definita in (1) è chiara: basta porre

$$c_{ij} + c_{ji} = J_{ij}$$

$$b = \frac{1}{kT}$$

$$\eta = -mH$$

Negli ultimi anni ha preso piede un'estensione del modello di Ising, chiamata modello di Edward-Anderson, in cui i valori di J_{ij} sono variabili aleatorie tra loro indipendenti. Questo tipo di approccio si è rivelato molto utile in fisica nello studio dei vetri (*spin glasses*).

III. ANALISI TEORICA ASINTOTICA

In questa sezione vedremo che la scelta dei parametri b,η dell'algoritmo influenza notevolmente le sue prestazioni. Il parametro η, come abbiamo visto, é associabile all'intensità del campo magnetico esterno H nel modello di Ising, e pertanto è facile intuirne gli effetti. Ad esempio nel caso di *virtuous network* ($t_i = 1$, $\forall i$), un campo esterno che tende a forzare gli spin nella direzione 'giusta' ($\eta < 0$) ha effetti positivi, mentre ponendo $\eta > 0$ si ottengono risultati disastrosi. Questo fatto rende evidente che la scelta di η é cruciale per la performance dell'algoritmo. Dall'interpretazione di η nel modello di Ising, appare in prima analisi plausibile la scelta neutrale di prendere $\eta = 0$, che equivale a non privilegiare a priori alcun orientamento. Questa scelta viene portata avanti anche da John Baras e Tao Jiang in [1], ma si rivela ben poco accorta ad una analisi più approfondita. Vale infatti il seguente teorema.

Teorema 2: Qualunque siano T, b e p_e, e per ogni topologia della rete di comunicazione, se $\eta = 0$ allora $P_{correct} = 0.5$.

Dimostrazione: Consideriamo gli stati S e $-S$ ottenuto cambiando il segno degli elementi di S. Dalla definizione della funzione energia U si deduce che $U(S) = U(-S)$ quando $\eta = 0$. Possiamo quindi dedurre che le probabilità in regime stazionario di S e $-S$ sono uguali, poichè $\pi_S = \frac{e^{bU(S)}}{Z} = \pi_{-S}$. Consideriamo ora la funzione

$$P_c(S) = 1 - \frac{||S - T||_1}{2N}$$

che rappresenta l'errore di stima commesso nello stato S. Dalla definizione di $P_{correct}$ (5) abbiamo:

$$P_{correct} = \mathbf{E}_{\pi_S}[P_c(S)] = \sum_S \pi_S Pc(S)$$

Accorpiamo ora i termini relativi a S e $-S$, sommando quindi su metà stati ottenendo:

$$\begin{aligned} P_{correct} &= \sum_{metá\ stati\ S} \pi_S Pc(S) + \pi_{-S} Pc(-S) \\ &= \sum_{metá\ stati\ S} \pi_S \left(Pc(S) + Pc(-S)\right) \end{aligned}$$

cioè

$$P_{correct} = \sum_{metá\ stati\ S} \pi_S \left(2 - \frac{||S - T||_1}{2N} - \frac{||-S - T||_1}{2N}\right)$$

Siano ora $a = |\{s_i | s_i \neq T_i\}|$ il numero di stime sbagliate e $b = |\{s_i | s_i = T_i\}| = N - a$ il numero di stime corrette. Allora

$$\begin{aligned} P_{correct} &= \sum_{metá\ stati\ S} \pi_S \left(2 - \frac{2a}{2N} - \frac{2b}{2N}\right) \\ &= \sum_{metá\ stati\ S} \pi_S \ = \ \frac{1}{2} \end{aligned}$$

□

Quindi $P_{correct}$ è pari a $\frac{1}{2}$ indipendentemente da b e dalla topologia quando $\eta = 0$. In particolare, anche nel caso di *virtuous network*, con questo algoritmo la bontà della stima è pari a quella che si otterrebbe con un algoritmo che per decidere lancia una moneta.

A partire da questo risultato teorico, discutiamo ora il problema da un punto di vista quantitativo, il che permette di capire perchè le simulazioni proposte in [1] non coincidono con il precedente teorema. L'approccio più diretto per calcolare numericamente $P_{correct}$ è quello di utilizzare direttamente la definizione di aspettazione calcolando opportunamente la distribuzione di probabilità π_S. Purtroppo per ottenere π_S occorre calcolare la costante di normalizzazione $Z = \sum_S \pi_S$, che é composta da 2^N addendi, tanti quanti sono gli stati S della catena di Markov associata al problema in esame. Questo metodo si scontra con evidenti difficoltà computazionali, come chiarisce il seguente esempio.

Supponiamo di avere un gigantesco calcolatore parallelo, con 10 milioni di processori ognuno dei quali possa calcolare π_S e sommarlo al fattore di normalizzazione in un nanosecondo. Facendo girare il calcolatore per l'età dell'universo otteniamo:

$$\begin{aligned} &10^7 \text{proc.} \times 10^9 \frac{\text{stati}}{\text{proc. sec}} \times 10^{14} \frac{\text{sec}}{\text{anno}} \times 10^{10} \text{anni} = \\ &= 10^{40} \text{stati} \approx 2^{133} \text{stati} \end{aligned}$$

L'intera età dell'universo permetterebbe quindi di calcolare Z per una rete di appena 133 nodi! Evidentemente questo approccio è possibile solo per reti piccole, ma almeno in questo caso ci ha permesso sia di verificare per via simulativa il teorema, sia di confrontare la distribuzione di probabilità teorica π_S con quella che si ottiene misurando la probabilità di trovarsi in un certo stato della catena di Markov a regime, ottenendo un'ulteriore conferma della teoria.

Per risolvere questo problema computazionale i fisici teorici Metropolis, Teller e Rosenbluth, impegnati nel progetto Manhattan, idearono il cosiddetto algoritmo di Metropolis, che consiste nel costruire una catena di Markov con 2^N stati in modo tale che a regime esista una distribuzione di probabilità stazionaria e che coincida con π_S. Questo risultato si ottiene ponendo la probabilità di transizione da uno stato S ad uno stato R:

$$\mathbb{P}[S \to R] = \begin{cases} 1 & se\ U(R) > U(S) \\ e^{b(U(R)-U(S))} & altrimenti \end{cases}$$

Analogamente a quanto accade con la voting rule proposta in [1], anche in questo caso la matrice delle probabilità di transizione ottenuta in questo modo ha π_S come autovettore destro, ovvero come distribuzione di probabilità a regime. Ciò avviene perchè é soddisfatta l'equazione di *detailed balance* (2) riscrivibile come

$$\frac{\pi_R}{\pi_S} = \frac{\mathbb{P}[S \to R]}{\mathbb{P}[R \to S]}$$

Una volta ottenuta una matrice delle probabilità di transizione A tale da soddisfare questa espressione (ad esempio con

l'algoritmo di Metropolis o la voting rule proposta in [1]), si possono calcolare aspettazioni di funzioni il cui argomento é distribuito secondo la distribuzione di probabilità π_S nel seguente modo:

- Si inizializza uno stato $S(0)$ iniziale qualunque (equivale a fissare un certo nodo di partenza sulla catena di Markov a 2^N stati)
- Si fa evolvere il sistema per n^* iterazioni
- Per n^* abbastanza grande assumiamo di aver raggiunto la distribuzione stazionaria (significa che nel sistema dinamico $x(n+1) = Ax(n)$ di dimensione 2^N , lo stato si è allineato con l'autovettore destro di A)

A questo punto i vari stati $S(n^*+1)$,...,$S(n^*+M)$ possono essere visti come campioni della variabile aleatoria distribuita secondo $\pi_{(.)}$ e quindi per ergodicità (il grafo della catena di Markov è connesso) vale

$$\mathbf{E}_{\pi_S}[f(S)] = \sum_S f(S)\pi_S = \lim_{M\to\infty} \frac{1}{M} \sum_{i=1}^{M} f(S(n^*+i))$$

Sostanzialmente quindi risolviamo il problema con un classico metodo Montecarlo. Il fatto che i risultati simulativi proposti in [1] non coincidano con quelli teorici corretti è molto probabilmente dovuto a una di queste cause

- è stato usato un n^* troppo piccolo
- è stato usato un M troppo piccolo

Anche nelle simulazioni di modelli di Ising fisici, si pone il problema di capire quale sia il valore di n^* da usare, una questione nota come *thermalization* del sistema. Le soluzioni proposte a questo problema (*binning*, autocorrelazioni, *hot and cold starts*, cfr. [3] e [4]) esulano dalla nostra trattazione e non verrano quindi approfondite.
Probabilmente comunque la causa principale di errore è la scelta di un valore di M troppo basso. Infatti da un punto di vista intuitivo per temperature basse (b grandi) la catena di Markov diventa sempre meno connessa e quindi l'erogidicità, che in linea di principio vale per ogni b, si indebolisce, inficiando la validità del metodo Montecarlo sopra presentato. A conferma di queste intuizioni si può notare nelle simulazioni che al crescere di n^* ed M (quando $\eta = 0$) si ottiene una transizione di fase apparente che si sposta a valori di b sempre più grandi, avvicinando sempre più il risultato simulativo a quello teorico $P_{correct} = 0.5$ (Fig.9).
A questo punto si pone la questione di quale dei due risultati, quello teorico e quello ottenuto per via simulativa, sia più realistico. Ovviamente se il numero di votazioni che si effettuano in una applicazione realistica è basso, non ha senso confrontarsi con il risultato teorico, che come abbiamo visto ha bisogno di un numero enorme di *sweeps* per essere effettivamente raggiunto. In questo caso ha certamente più senso prendere in considerazione il risultato simulativo, che però viene a dipendere dallo stato iniziale, da n^* e da M. D'altra parte se la rete rimane attiva per tempi molto lunghi continuativamente, diventa necessario confrontarsi con i risultati asintotici previsti dal teorema.

In questo secondo scenario più formale e matematicamente trattabile ed elegante, torniamo a chiederci come scegliere i parametri b ed η alla luce di quanto visto fin qui. Ora supponiamo di conoscere la percentuale di nodi cattivi presenti nella rete, ipotesi senz'altro realistica (ad esempio in applicazioni reali non sembra riduttivo pensare che ci siano più nodi buoni che cattivi). Qual'e la scelta ottima dei parametri sotto queste ipotesi?
Iniziamo col trattare dei casi limite per aiutare l'intuizione. Nel caso di *virtuous network* chiaramente la scelta ottima risulta $\eta = -\infty$, ovvero un campo esterno forzante che forza i nodi nella direzione giusta di ampiezza infinita (significa mettere $s_i = 1$ con probabilità uno con la *voting rule* usata). Evidentemente nel caso opposto di soli nodi cattivi ($T = [-1, \ldots, -1]$) la scelta ottima risulta essere $\eta = \infty$ (in questo caso significa mettere $s_i = -1$ con probabilità uno con la *voting rule* usata). Risulta interessante notare come questi risultati siano validi indipendentemente dalla topologia della rete considerata e indipendentemente da quale sia il comportamento dei nodi cattivi quando si tratta di votare.
Il problema di studiare cosa accade nei casi intermedi appare difficilmente trattabile per via analitica, soprattutto perchè dipende da:

- Topologia della rete
- Politica di voto dei nodi cattivi

Studieremo i casi intermedi su un *lattice 2-D* con *periodic boundaries* e per dare più simmetria al problema supporremo che i nodi cattivi attribuiscano voti contrastanti. Un ovvio vantaggio nel considerare un *lattice 2-D* è che si può effettivamente parlare di percentuale di nodi buoni e di nodi cattivi, dal momento che non c'è ragione di distinguere un nodo dall'altro, mentre in altre topologie bisognerebbe decidere quali sono i nodi buoni e quelli cattivi, dal momento che ad esempio non tutti i nodi comunicano con lo stesso numero di vicini. Inoltre per problemi di natura computazionale considereremo una rete da $N = 16$ nodi, in modo tale da poter calcolare in tempi ragionevoli la funzione di partizione e quindi $P_{correct}$ senza bisogno di ricorrere a metodi Montecarlo.
Vale il seguente teorema.

Teorema 3: In una rete strutturata come un *lattice 2-D* con *periodic boundaries*, detta p la frazione di nodi buoni su N, e supponendo che i nodi cattivi attribuiscano voti contrastanti, si ha che

$$P_{correct}(b, \eta, p) = P_{correct}(b, -\eta, 1-p)$$

Dimostrazione:

$$P_{correct}(b, \eta, p) =$$
$$= \frac{1}{Z} \sum_S \left(1 - \frac{||S-T||_1}{2N}\right) e^{b(\sum (c_{ij}+c_{ji}) s_i s_j - \eta \sum s_i)}$$

$$P_{correct}(b,-\eta,1-p) = \\ = \frac{1}{Z}\sum_{S}\left(1-\frac{||S+T||_1}{2N}\right)e^{b(\sum(c_{ij}+c_{ji})s_i s_j+\eta\sum s_i)}$$

dove i c_{ij} non variano perchè vengono scambiati sia T_i che T_j e i nodi cattivi danno voti contrastanti. Siccome l'insieme degli stati S è simmetrico rispetto all'origine vale:

$$P_{correct}(b,-\eta,1-p) = \\ = \frac{1}{Z}\sum_{S}\left(1-\frac{||-S+T||_1}{2N}\right)e^{b(\sum(c_{ij}+c_{ji})(-s_i)(-s_j)+\eta\sum -s_i)}$$

ovvero

$$P_{correct}(b,-\eta,1-p) = P_{correct}(b,\eta,p)$$

□

Una naturale interpretazione intuitiva di questo teorema è quella di pensare di ribaltare l'asse rispetto al quale vengono misurati gli spin (ovvero guardare il sistema dalla direzione opposta). Così facendo si trova un sistema dove $\eta' = -\eta$ e la percentuale di nodi buoni risulta $1-p$. Siccome la dinamica del sistema non può dipendere dal punto di riferimento, chiaramente $P_{correct}$ (che è la percentuale di nodi allineati correttamente) non cambia.
Cercheremo ora di trovare dei valori di

$$\eta_{ott} = \arg\max_{\eta} P_{correct}(b,\eta,p)$$

con b e p fissati.
Grazie al teorema 3, possiamo limitarci a cercare il valore di η ottimo η_{ott} solo per $p \geq 0.5$, dal momento che i valori per $p < 0.5$ si ottengono per simmetria. Intuitivamente si capisce che se $p > 0.5$ allora $\eta_{ott} < 0$, perchè dal momento che sappiamo a priori che ci sono più nodi buoni che cattivi si cercherà di privilegiare la stima $s_i = 1$ (nodo i-esimo buono) rispetto a $s_i = -1$.
Nel caso di $p = 0.5$, per via simulativa si vede che $P_{correct} = 0.5$ indipendentemente da b e da η. Questo risultato è intuitivo se si pensa che dal momento che ci sono metà nodi buoni e metà cattivi non ha senso privilegiare un certo orientamento degli spin. Prendiamo ad esempio $\eta_{ott} = 0$.
Nelle tabelle vengono riportati alcuni valori di η_{ott} ottenuti risolvendo il problema di massimizzazione in modo numerico con l'ausilio di MATLAB.

b	η_{ott}	$P_{correct}$
0.2	−2.0093	0.9017
0.25	−1.5947	0.9629
0.3	−1.4481	0.9862
0.35	−1.3786	0.9945
0.4	−1.3327	0.9977
0.6	−1.2095	0.9999
0.8	−1.1321	1
1	−1.0829	1

Tabella I
Valori di η ottimo in funzione di b, $p = 0.75$

b	η_{ott}	$P_{correct}$
0.2	−1.9528	0.7808
0.25	−1.7623	0.8926
0.3	−1.7552	0.9531
0.35	−1.7800	0.9794
0.4	−1.8049	0.9907
0.6	−1.8527	0.9995
0.8	−1.8484	1
1	−1.8228	1

Tabella II
Valori di η ottimo in funzione di b, $p = \frac{10}{16}$

b	η_{ott}	$P_{correct}$
0.2	−2.9133	0.9569
0.25	−2.0581	0.9822
0.3	−1.6316	0.9928
0.35	−1.3946	0.9970
0.4	−1.2479	0.9987
0.6	−1.2479	0.9987
0.8	−1.8484	1
1	−1.8228	1

Tabella III
Valori di η ottimo in funzione di b, $p = \frac{14}{16}$

Se quindi la scelta $\eta = 0$ rendeva praticamente inutile l'algoritmo ($P_{correct} = 0.5$ è un valore inaccettabile a fini pratici) queste simulazioni dimostrano come con una accorta scelta dei parametri η e b si possano ottenere risultati eccellenti ($P_{correct}$ prossima a 1). Naturalmente tutto questo è possibile solo se si conosce una stima di p, che in questo contesto rappresenta la soglia con cui va confrontata la $P_{correct}$. Infatti un *naive algorithm* che stima $s_i = 1\ \forall\ i$ quando $p > 0.5$ e $s_i = -1\ \forall i$ quando $p < 0.5$ ottiene una probabilità di stima corretta, con cui si deve confrontare la $P_{correct}$, pari a $\max\{p, 1-p\}$.

IV. Analisi simulativa: reti statiche

Nella sezione precedente lo studio di reti composte da pochi nodi ha permesso di verificare tramite simulazione alcuni importanti risultati teorici. Nelle applicazioni reali, però, si ha spesso a che fare con reti di grandi dimensioni, per le quali le affermazioni fatte precedentemente continuano a valere da un punto di vista teorico ma non sono più verificabili in pratica. Infatti, com'è già stato fatto intuire poc'anzi parlando di reti di piccole dimensioni, per provare tali affermazioni tramite simulazione nel caso di N elevato è necessario calcolare $P_{correct}$ usando il metodo Montecarlo e attribuendo a n^* e a M valori sufficientemente elevati. Purtroppo, però, ragionando anche solo intuitivamente, si capisce che, a parità di b, n^* ed M, tanto più aumenta il numero N di nodi, tanto più diminuisce la connettività del grafo della catena di Markov con 2^N stati associata al problema in esame. Questo fatto ha due conseguenze:

1) serve molto più tempo affinchè la variabile aleatoria $\mathbf{S}(n)$ sia distribuita secondo la π_S;

2) l'ergodicità del processo aleatorio $\mathbf{S}(\cdot)$, e quindi anche la validità del metodo Montecarlo, continuano a sussistere solo se M aumenta.

In altre parole, se n^* ed M non variano, al crescere di N il metodo Montecarlo approssima sempre peggio l'aspettazione $\mathbf{E}_{\pi_S}[f(S)]$. Per aggiustare le cose, quindi, risulta necessario aumentare n^* ed M fintantochè questa tecnica computazionale non approssimi bene l'oggetto da calcolare. D'altro canto, operando per via simulativa, si vede che un piccolo incremento di N rende necessario un aumento elevato di n^* ed M. Sembra perciò esserci un *trade-off* tra la capacità di ottenere una buona approssimazione dell'aspettazione e il numero di iterazioni dell'algoritmo necessarie affinchè questo accada, che solitamente è molto più grande rispetto a quello che si riesce effettivamente a ottenere nelle simulazioni e nelle applicazioni reali. Per questo, in generale, usando il metodo Montecarlo con due valori computazionalmente accettabili di n^* ed M, non si ha una buona approssimazione di $\mathbf{E}_{\pi_S}[f(S)]$, e quindi neanche della $P_{correct}$.
Risulta perciò necessario definire una nuova variabile, $\bar{P}_{correct}$, che misuri la probabilità di stima corretta di T a regime. Rifacendoci all'espressione di $P_{correct}$ usata finora possiamo scrivere

$$\bar{P}_{correct} = \frac{\sum_{j=1}^{n_{st}} \sum_{i=1}^{M} \left[1 - \frac{\|S(n^*+i)-T\|_1}{2N}\right]}{n_{st} M} \tag{6}$$

dove n_{st} è il numero di stati iniziali di S su cui si media e $\|S(n^*+i)-T\|_1 = \sum_{i \in V} |s_i - t_i|$ all'iterazione n^*+i. Ovviamente, operando in questo modo, non ha più senso confrontarsi con i risultati teorici asintotici, poichè vengono a mancare molte delle ipotesi su cui essi si basavano.
Oltre a dipendere da due nuovi parametri, n^* ed M, le simulazioni su reti di grandi dimensioni sono influenzate anche dallo stato iniziale del vettore S. Ciò è ragionevole se si pensa che, data la scarsa connettività del grafo della catena di Markov, n^* ed M sono troppo piccoli per permettere che, a partire da un qualunque stato iniziale, anche tutti gli altri vengano toccati. É molto più plausibile che, dato uno stato iniziale di S, le successive n^*+M iterazioni portino a stati ad esso vicini.
Per chiarezza è bene dire fin da subito che, se non viene specificato altrimenti, ogni simulazione presentata in questa sezione e nella prossima è stata ottenuta modellizzando una rete con il *WS model*, e ponendo $N = 100$ e $k = 4$.

A. *Un caso elementare.*

Per prima cosa è bene prendere in esame il caso più elementare, ovvero quello in cui la rete è interamente composta da nodi buoni che non commettono errori nel votare i loro vicini. Ciò implica che $t_i = 1$, $\forall i \in V$ e $c_{ij} = 1$, $\forall (i,j) \in E$. Sotto queste ipotesi il parametro di valutazione della regola di voto, ovvero $\bar{P}_{correct}$, assume la seguente forma

$$\bar{P}_{correct} = \frac{\sum_{j=1}^{n_{st}} \sum_{i=1}^{M} [\langle S(n^*+i) \rangle] + N}{2N n_{st} M}$$

dove $\langle S(n^*+i) \rangle = \sum_{i \in V} s_i$ all'iterazione n^*+i. Bisogna poi fissare i parametri simulativi n^* ed M, nonchè lo stato iniziale di S. Quest'ultimo è scelto in base a un parametro, p_g, nel modo seguente: l'elemento i-esimo di S è uguale a l'elemento i-esimo di T con probabilità p_g. In altre parole, tale parametro dice quanto si pensa di essere vicini al vettore T reale che dev'essere stimato. Non volendo sbilanciarsi troppo si può fare la scelta neutrale $p_g = 0.5$, che equivale ad avere incertezza completa su quale possa essere il T reale. I valori di n^* ed M vanno fissati senza alcun criterio particolare, ricordando però che tanto più essi sono elevati, tanto più $\bar{P}_{correct}$ tende ad approssimare bene $P_{correct}$, tanto più i risultati ottenuti per via simulativa dovrebbero avvicinarsi a quelli teorici. Normalmente M è scelto in modo direttamente proporzionale ad n^*.
Innanzitutto si può iniziare a tracciare i grafici di $\bar{P}_{correct}$ in funzione dei due parametri più importanti della regola di voto, ovvero b e η. In Fig.7 è disegnato l'andamento di $\bar{P}_{correct}$ per valori di b equidistribuiti tra 0 e 1, e per $\eta = -1, 0, 1$. Inoltre si pone $p_g = 0.5$, $n^* = 140000$, $M = 10000$ e $n_{st} = 50$. Si vede che, mentre per $\eta = \pm 1$

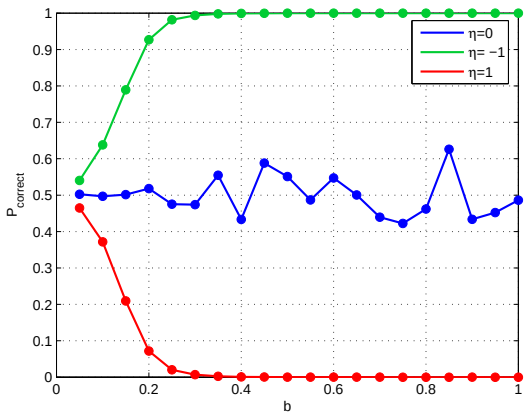

Figura 7. Andamenti di $\bar{P}_{correct}(b)$ per $\eta = -1, 0, 1$, fissando $p_g = 0.5$.

continuano a valere i commenti già fatti parlando della scelta di η nel caso di *virtuous network* di piccole dimensioni, per $\eta = 0$ il grafico di $\bar{P}_{correct}$ si allontana dal risultato teorico espresso nel teorema 2. Infatti non si ottiene $\bar{P}_{correct} = 0.5$, $\forall b \in [0,1]$, bensì un andamento oscillatorio attorno al valore 0.5. D'altro canto, se si provasse a fare una media dei valori assunti da $\bar{P}_{correct}$, si vedrebbe che essa si avvicina molto a 0.5. Come si spiega questo fenomeno? Innanzitutto, sulla base dei ragionamenti condotti all'inizio di questa sezione, non stupisce il fatto di vedere sorgere una discrepanza tra risultati teorici e simulativi. Volendo poi motivare l'andamento di $\bar{P}_{correct}$, si può riprendere l'idea

intuitiva su cui si basa il teorema 2. Per com'è fatta la regola di voto, è chiaro che porre $\eta = 0$ equivale ad avere una completa incertezza a priori sul vettore T. Dal momento che anche il parametro p_g è legato all'incertezza a priori su T, porre $\eta = 0$ e $p_g = 0.5$ equivale ad avere incertezza totale su T, motivo per cui si osserva un andamento oscillatorio di $\bar{P}_{correct}$ attorno a 0.5. Questa spiegazione è confermata dagli andamenti di $\bar{P}_{correct}$ con $\eta = \pm 1$. Infatti in quei due casi, anche se $p_g = 0.5$, il parametro η fa ben comprendere quale sia la stima a priori di T: se $\eta = -1$ significa che si è ottimisti e si pensa che tutti i nodi siano buoni, per cui, essendo $t_i = 1$, $\forall i \in V$, $\bar{P}_{correct}$ arriva sicuramente a 1 per valori di b sufficientemente elevati ($b > 0.3$), mentre se $\eta = 1$ avviene l'esatto contrario.
Si può quindi pensare di variare p_g mantenendo fisso η, e vedere cosa succede. Ovviamente, se $\eta = \pm 1$, cambiando p_g concordemente alla scelta fatta su η gli andamenti di $\bar{P}_{correct}(b)$ non possono che migliorare. Il caso interessante si ha con $\eta = 0$. In Fig.8 sono mostrati gli andamenti di $\bar{P}_{correct}(b)$ per diversi valori di p_g e $\eta = 0$. Si osserva

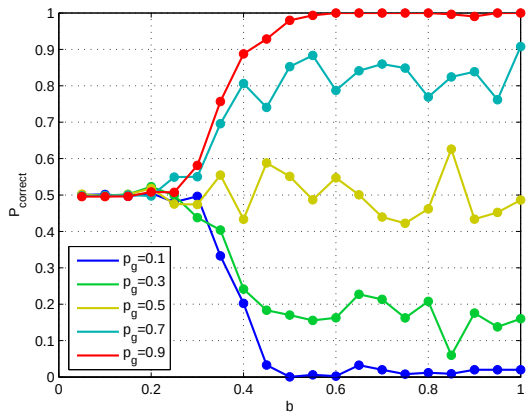

Figura 8. Andamenti di $\bar{P}_{correct}(b)$ con $\eta = 0$ e per diversi valori di p_g.

che essi dipendono fortemente dal valore di p_g, e quindi dalla condizione iniziale di S: tanto più esso si avvicina a 1, tanto più $\bar{P}_{correct}$ tende a 1 al crescere di b, e viceversa. Valgono gli stessi ragionamenti fatti nel caso in cui $p_g = 0.5$ e $\eta = \pm 1$. Avere p_g vicino a 1 significa essere ottimisti, e quindi scegliere un vettore S di stima iniziale vicino al vettore reale T, e viceversa. D'altro canto non è così assurdo pensare di assegnare a p_g un valore elevato, dal momento l'algoritmo che si sta studiando andrà poi applicato nel tempo, e sarà difficile, per non dire impossibile, che il vettore S e T differiscano in tantissimi punti. Ma di questo si parlerà meglio in seguito.
Un altro aspetto che va notato è che per valori di b inferiori a 0.3 il valore di $\bar{P}_{correct}$ si assesta sempre attorno a 0.5 indipendentemente dal valore di p_g. Ciò si comprende se si pensa che b è legato, in un certo senso, all'incertezza sulla validità della regola di voto: più piccolo è b, più lo scenario è incerto, più aumenta la connettività della catena di Markov, più $\bar{P}_{correct}$ tende a 0.5. Basti pensare al caso estremo in cui $b = 0$, che comporta $\mathbb{P}[s_i = 1] = \mathbb{P}[s_i = -1] = 0.5$.
Si vede inoltre che, scegliendo un buon valore di p_g, ad esempio 0.9, per $0.3 \leq b \leq 0.5$ $\bar{P}_{correct}$ subisce un veloce incremento da 0.5 (*fase random*) a valori molto più vicini a 1 (*fase deterministica*). Questo fenomeno, che in letteratura è noto come *phase transition*, è analogo a quello che avviene nel modello di Ising quando il campo magnetico è nullo e si abbassa la temperatura al di sotto di una certa soglia T_c, che rappresenta proprio la temperatura di Curie.
Complessivamente si può concludere che, quando si ha a che fare con reti di grandi dimensioni, i parametri legati all'incertezza a priori su T sono tre: η, p_g e b. Ponendo quest'ultimo abbastanza elevato, i.e. $b > 0.6$, basta operare anche solo su uno degli altri due parametri per vedere migliorare le prestazioni della regola di voto.
Mediando sugli andamenti di $\bar{P}_{correct}(b)$ relativi a diversi stati iniziali di S si ricavano dei valori molto prossimi a 0.5. Notato ciò, è ragionevole pensare che, mediando ipoteticamente su tutti i possibili stati iniziali di S, si ottenga $\bar{P}_{correct} = 0.5$, $\forall b \in [0, 1]$, che corrisponde proprio al risultato teorico espresso nel teorema 2. Quindi sembra che mediare su tutti i possibili andamenti relativi a diversi stati iniziali di S e ottenuti usando il metodo Montecarlo con n^* ed M non troppo elevati sia equivalente ad adoperare il metodo Montecarlo con valori di n^* e M grandi quanto basta per poter applicare il teorema 2. In realtà questa supposizione non appare affatto assurda se si pensa che, con n^* ed M piccoli, partendo da uno stato iniziale di S si possono raggiungere solo gli stati ad esso più prossimi, mentre con n^* ed M sufficientemente elevati si riesce a spaziare su tutti gli stati della catena di Markov. Notato ciò, è chiaro che mediando su andamenti ottenuti con n^* ed M piccoli si ricava un risultato molto simile a quello relativo a un solo andamento di $\bar{P}_{correct}$ calcolato con n^* ed M elevatissimi.
Un'ultima osservazione che merita di essere fatta riguarda la scelta dei parametri n^* ed M. In Fig.9 sono mostrati diversi grafici di $\bar{P}_{correct}$, con $\eta = 0$ e $p_g = 0.9$, al variare di n^*, e con $n^* \simeq 14M$. Si osserva che al crescere di n^* la transizione di fase tende a spostarsi sempre più verso destra, ovvero tende a verificarsi per valori sempre più grandi di b. Ciò è in perfetto accordo con il teorema 2, in quanto, se ipoteticamente si potesse fissare un n^* infinito, la transizione di fase si vedrebbe per valori di $b \to \infty$, che equivale a dire che $\bar{P}_{correct} = 0.5$, $\forall b \in \mathbb{R}^+$.
Dall'analisi fatta in questo paragrafo si capisce chiaramente che l'algoritmo proposto in [1] presenta una forte dipendenza dai parametri, per cui è necessaria una buona operazione di *tuning* prima di utilizzare questo sistema per la stima dell'affidabilità dei nodi di una rete di sensori.

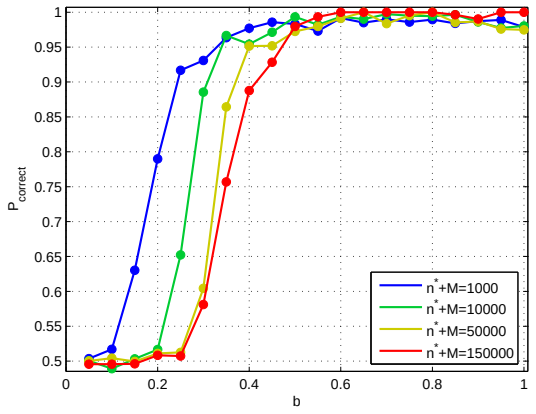

Figura 9. Andamenti di $\bar{P}_{correct}(b)$ con $\eta = 0$ e $p_g = 0.9$, per diversi valori di n^*.

B. *Rete con nodi buoni ma errori nei voti.*

Una modifica che può essere introdotta per complicare un po' le cose, e per renderle nello stesso tempo più attinenti alla realtà, consiste nel considerare la *probabilità d'errore* p_e precedentemente definita come

$$p_e = \mathbb{P}\left[c_{ij} \neq t_j | t_i = 1\right] \quad , \ \forall\ i \text{ buono.}$$

Continuando a supporre che nodi siano tutti buoni, i.e. $t_i = 1, \forall i \in V$, si ha che, per questo caso particolare,

$$\mathbb{P}\left[c_{ij} = 1\right] = 1 - p_e \qquad \mathbb{P}\left[c_{ij} = -1\right] = p_e$$

Valutando l'andamento di $\bar{P}_{correct}(b)$ al variare di p_e e fissando $\eta = 0$ e $p_g = 0.9$, che sembra poi essere il caso più interessante e nel contempo quello che si avvicina maggiormente a un'applicazione reale, si ottengono i grafici in Fig.10. Si vede che la transizione di fase esiste ancora, ma

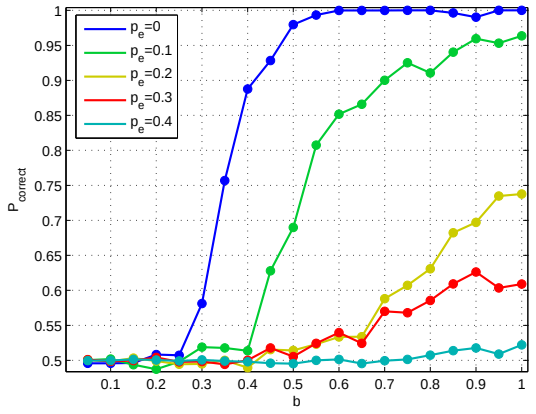

Figura 10. Andamenti di $\bar{P}_{correct}(b)$ con $\eta = 0$ e $p_g = 0.9$, per diversi valori di p_e.

solo per valori di p_e inferiori a una certa soglia dipendente anche da n^*. Nel caso in esame è sufficiente che $p_e = 0.4$ affinchè non si verifichi più questo fenomeno, e si abbia $\bar{P}_{correct} \simeq 0.5$, $\forall b \in [0,1]$. In altre parole, la transizione di fase si sposta rapidamente verso destra al crescere di p_e, ovvero si verifica per valori sempre più elevati di b, fino a scomparire. D'altro canto è giusto che succeda questo, in quanto al crescere di p_e i voti sbagliati tendono a destabilizzare gradualmente i voti corretti, per cui è molto più difficile stimare bene il vettore T e si tende a rimanere sempre di più nella fase *random*.

C. *Rete con nodi anche cattivi.*

Il sistema appena descritto si può complicare ancora imponendo che nella rete ci possano essere dei nodi cattivi, i.e. $t_i = -1$ per qualche $i \in V$. Naturalmente, apportando questa modifica al problema, bisogna utilizzare l'espressione più generale di $\bar{P}_{correct}$ per valutare le prestazioni della regola di voto, ovvero la (6). Ovviamente in questo caso stimare correttamente T non significa più avere $s_i = 1$, $\forall i \in V$ a regime, bensì identificare correttamente i nodi buoni e quelli cattivi.
Innanzitutto bisogna decidere come si comporta un nodo cattivo. Come già detto si possono prendere in considerazione tre diversi criteri di attribuzione dei voti:

- *voti* ***indipendenti***
- *voti* ***contrastanti***
- *voti* ***random***

Risulta interessante vedere cosa cambia, in termini di prestazione della regola di voto, ipotizzando un comportamento piuttosto che un altro. In Fig.11 sono disegnati quattro andamenti di $\bar{P}_{correct}$ in funzione della percentuale p_m di nodi cattivi nella rete. Nel primo si suppone che tutti i nodi siano buoni, mentre nei restanti tre si assume la presenza di nodi cattivi che si comportano nei tre modi appena descritti. Inoltre si pone $\eta = 0$, $p_e = 0.2$, $b = 1$ e $p_g = 0.9$, in modo da avere prestazioni buone e abbastanza verosimili a regime. Il primo comportamento

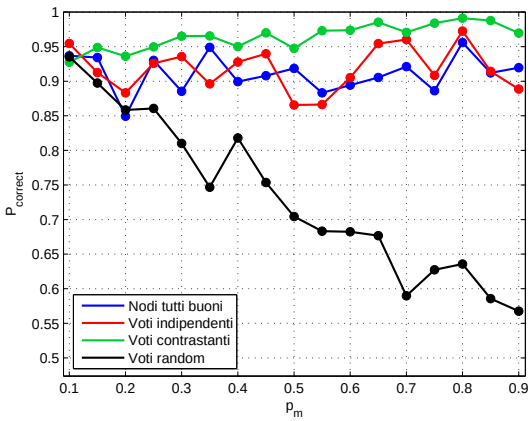

Figura 11. Andamenti di $\bar{P}_{correct}$ in funzione della percentuale p_m di nodi cattivi, con $\eta = 0$, $p_e = 0.2$, $b = 1$ e $p_g = 0.9$.

che può essere assunto dai nodi cattivi porta a prestazioni molto simili a quelle ottenute nel caso di nodi tutti buoni,

il secondo va addirittura meglio, mentre nel terzo caso le prestazioni degradano notevolmente al crescere di p_m.
Per quanto riguarda la presenza di nodi cattivi che danno voti indipendenti vale il seguente teorema dovuto a Baras e Jiang.

Teorema 4: Supponendo di costruire un grafo G^* avente la stessa topologia di G e la stessa probabilità d'errore p_e, ma con nodi *tutti* buoni, e definendo $\bar{P}^*_{correct}$ la probabilità di stima corretta in G^*, si ha che $\bar{P}_{correct}$, relativa alla presenza di nodi cattivi che danno voti indipendenti, è uguale a $\bar{P}^*_{correct}$.

In altre parole questo teorema, per la cui dimostrazione si rimanda in [1], afferma che, nel caso di voti indipendenti, $\bar{P}_{correct}$ non dipende dalla percentuale di nodi cattivi presenti nella rete. Tale risultato è molto importante, in quanto ci permette di affermare che tutti i grafici e le osservazioni fatti finora continuano a valere anche nel caso in cui vi siano nodi cattivi che attribuiscono voti indipendenti ai vicini.
Passando al secondo tipo di comportamento, si può dire che la prestazione della regola di voto è migliore rispetto al caso precedente, e quindi anche rispetto al caso di nodi tutti buoni, in quanto, non essendoci una probabilità d'errore p_e, è più facile identificare i nodi cattivi, a prescindere da quanti siano, poichè votano sempre con $c_{ij} \neq t_j, \forall (i,j) \in E$.
Nel caso di voti *random* è impossibile stimare correttamente l'affidabilità dei nodi, e la cosa si complica sempre di più al crescere del numero di nodi cattivi, proprio perchè essi non seguono una legge deterministica nell'assegnare i voti ai vicini.

D. *Influenza della topologia.*

In questo paragrafo si mostrerà che la topologia della rete di affidabilità gioca un ruolo notevole nella prestazione della regola di voto. Finora tutti i grafici analizzati sono stati ottenuti modellizzando la rete secondo il *WS model*, con $N = 100$, $k = 4$ e $p_{rw} = 0$. Può essere interessante capire cosa succede variando questi parametri. Nella fattispecie si faranno dei grafici di $\bar{P}_{correct}(b)$ al variare di p_{rw}, di k o di N.
Dapprima si considera l'andamento di $\bar{P}_{correct}(b)$ al variare di p_{rw}. Com'è stato spiegato nella sezione II-B, la condizione $p_{rw} = 0$ equivale ad avere un grafo in cui gli archi sono sistemati ordinatamente secondo un criterio deterministico. Sebbene finora si sia supposto di essere proprio in questa situazione, è pur vero che essa non descrive bene ciò che avviene in una rete reale, in cui i collegamenti tra un sensore e l'altro sono quasi sempre distribuiti in modo casuale. Volendo tener conto anche di questo aspetto si deve porre $p_{rw} \neq 0$. In Fig.12 sono rappresentati gli andamenti di $\bar{P}_{correct}(b)$ per diversi valori di p_{rw}, supponendo $p_e = 0.2$, $p_g = 0.9$, $\eta = 0$ e i soliti valori di n^* ed M. Inoltre si ipotizza che, nel caso in cui vi siano nodi cattivi, essi si comportino attribuendo voti indipendenti. Si vede chiaramente che, al crescere di p_{rw},

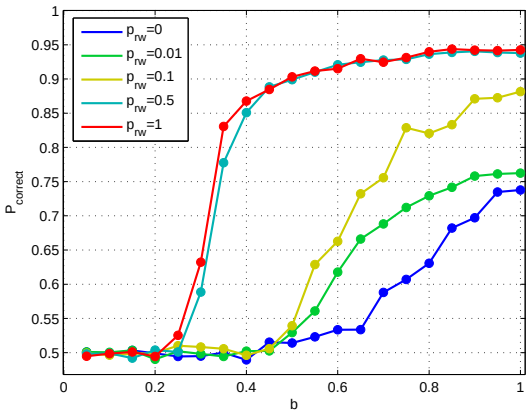

Figura 12. Andamenti di $\bar{P}_{correct}(b)$ con $\eta = 0$, $p_g = 0.9$ e $p_e = 0.2$ per diversi valori di p_{rw}.

la prestazione della regola di voto aumenta notevolmente. Ciò è dovuto al fatto che un incremento di p_{rw} implica un diametro medio più piccolo, ovvero una comunicazione più veloce nella rete di affidabilità. Per questo motivo, a parità di tempo di simulazione $n^* + M$, $\bar{P}_{correct}$ assume valori più elevati al crescere di p_{rw}. Inoltre si nota che i miglioramenti più considerevoli nelle prestazioni avvengono soprattutto per valori piccolissimi di p_{rw}. Ma anche questo è corretto se si pensa alla curva del diametro medio L del grafo in funzione di p_{rw}, un tipico andamento della quale è rappresentatato in Fig.5. Si vede che L diminuisce notevolmente proprio per valori di p_{rw} che corrispondono a una topologia *small-world*.
In Fig.13, invece, si valuta la prestazione della regola di voto al variare del numero k di nodi vicini a un nodo target. Si suppone $p_{rw} = 0$ e tutti gli altri parametri invariati rispetto alla simulazione precedente. Ovviamente si ottiene

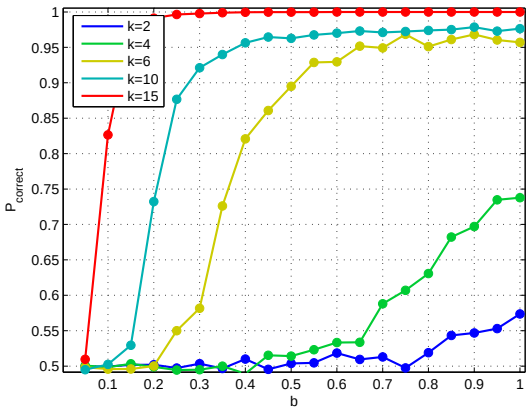

Figura 13. Andamenti di $\bar{P}_{correct}(b)$ con $\eta = 0$, $p_{rw} = 0$, $p_e = 0.2$ e $p_g = 0.9$, per diversi valori del numero di vicini.

una probabilità di stima corretta sempre maggiore al crescere

di k, in quanto più archi ci sono nella rete, più diminuisce il diametro medio del grafo associato, più l'informazione viaggia velocemente.
Infine rimane da vedere cosa succede fissando p_{rw} e k, e facendo variare il numero N di nodi della rete. I grafici relativi a questo scenario sono mostrati in Fig.14, dove si è scelto $p_{rw} = 0$ e $k = 3$. É evidente che una variazione di N

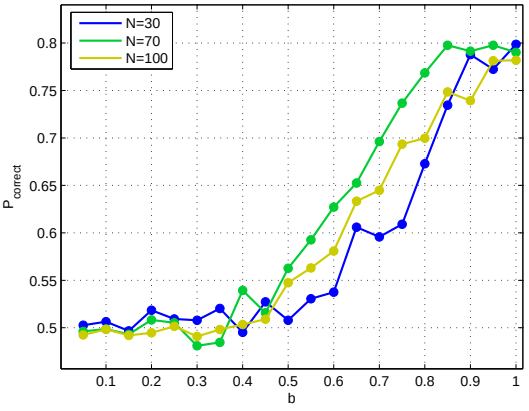

Figura 14. Andamenti di $\bar{P}_{correct}(b)$ con $\eta = 0$, $p_{rw} = 0$, $k = 3$, $p_e = 0.2$ e $p_g = 0.9$, per diversi valori di N.

con k fissato non comporta cambiamenti nella prestazione dell'algoritmo a regime. In altre parole quest'ultima dipende soltanto dal numero k di vicini che ogni nodo della rete possiede. Ciò è abbastanza ragionevole se si pensa che la regola di voto è una legge di aggiornamento locale che dipende soltanto dal numero di vicini che votano un nodo target, e non da quanti sono i nodi che complessivamente compongono la rete.
Tutti questi ragionamenti ci permettono di concludere che la topologia della rete di affidabilità ha una notevole importanza nella prestazione della regola di voto.

V. Analisi simulativa: reti dinamiche

Nella sezione precedente si è affrontato lo studio della regola di voto applicandola a modelli via via sempre più complessi, ma mantenendo i parametri in gioco costanti nel tempo e andando a valutare $\bar{P}_{correct}$ solo dopo un certo numero n^* di iterazioni. In realtà, però, un sistema di *trust management* come quello che è stato studiato finora lavora *dinamicamente* nel tempo. Da qui l'interesse nell'analisi di reti grandi[3] dinamiche.

A. Rete con c_{ij} tempo-varianti

I voti c_{ij} che il nodo i dà al nodo j sono senza dubbio variabili nel tempo in una rete reale. Essi, infatti, dovendo dare la propria opinione sull'affidabilità di un nodo, che nella pratica non è sicuramente una proprietà costante nel tempo, dovranno essere calcolati a intervalli di tempo fissati a priori. Inoltre, se $p_e \neq 0$, cosa molto verosimile in pratica, si ottiene inesorabilmente una variazione dei c_{ij} anche se il vettore reale T di affidabilità non varia nel tempo. In questo paragrafo si farà proprio tale ipotesi, ovvero si supporrà che le componenti di T siano costanti nel tempo e che i voti c_{ij} possano variare solo a causa della probabilità d'errore p_e. Per di più, immaginando che tutti i sensori installati nella rete siano buoni, si può porre $t_i = 1, \forall i \in V$, e quindi in questo caso avere un T costante nel tempo significa non avere danneggiamenti nei sensori stessi.
Innanzitutto sarebbe molto interessante capire se, anche con queste ipotesi, a regime esiste una densità di probabilità di una catena di Markov associata opportunamente costruita, proprio come la π_S nel caso di c_{ij} tempo-invarianti. A questo proposito vale il seguente teorema.

Teorema 5: Data la consueta regola di voto stocastica, con aggiornamenti randomizzati e asincroni, $b \in \mathbb{R}^+$ e $q_i > 0, \forall i \in V$, si ha che, se i voti c_{ij} sono processi aleatori a variabili indipendenti e identicamente distribuite secondo la π_C, tra loro indipendenti, allora la regola di voto converge a regime con un'unica densità di probabilità stazionaria, π_{SC}, la quale, date una configurazione per S e una per i c_{ij}, determina la probabilità di avere simultaneamente tali configurazioni a regime.

Dimostrazione: Per dimostrare il teorema basta ricordare che, se si ha a che fare con una catena di Markov *regolare*, allora a regime esiste un'unica densità di probabilità stazionaria ad essa associata. Affinchè una catena di Markov si possa definire regolare è sufficiente che il grafo ad essa associato sia fortemente connesso e composto da un numero finito di stati, ognuno dotato di *self-loop*. Nel caso in questione, operando come quando si aveva a che fare con c_{ij} tempo-invarianti, si può associare al nostro problema una catena di Markov con al più $2^N 5^{\frac{Nk}{2}}$ stati, in quanto per ognuna delle 2^N configurazioni di S esistono al massimo $5^{\frac{Nk}{2}}$ configurazioni di $\hat{c}_{ij}$ possibili, poichè si è tenuto conto del fatto che, con $p_{rw} \neq 0$, gli Nk pesi c_{ij} possono portare ad avere $\hat{c}_{ij} = 0, \pm 1, \pm 2$. Una volta creata questa grande catena di Markov basta verificare che le suddette condizioni sufficienti per la convergenza siano soddisfatte. Analizzando il grafo associato, e tenendo conto delle condizioni su b e q_i, è evidente che il numero di stati è enorme ma sicuramente finito, e che per ognuno di questi stati esiste un *self-loop* il cui peso (che è la probabilità di rimanere nello stato in cui si è già) può essere anche molto piccolo ma sempre strettamente positivo[4]. Inoltre il grafo è fortemente connesso, in quanto, fissati due nodi $i, j \in V$, esiste di sicuro un tragitto da i a j, che al limite può avere probabilità bassissima, ma non nulla, di essere percorso. Si può perciò

[3] Si considerano sempre $N = 100$ e $k = 4$.

[4] Si noti che tanto più p_e tende a zero, tanto più i pesi associati ai *self-loop* crescono.

concludere che la regola di voto converge a regime con un'unica densità di probabilità stazionaria π_{SC}.

□

Tale dimostrazione si basa sostanzialmente sul teorema di Perron-Frobenius, secondo cui una matrice P stocastica per colonne, irriducibile e con elementi strettamente positivi nella diagonale ha un autovalore massimo, pari a 1, al quale corrisponde un autovettore destro $\mathbf{v}_0$ con cui lo stato $\mathbf{x}(n)$ del sistema dinamico $\mathbf{x}(n+1) = P\mathbf{x}(n)$ tende ad allinearsi a regime. Ora, alla catena di Markov costruita nella dimostrazione è associata una matrice di transizione di stato, che chiameremo A_c, la quale ovviamente è quadrata e stocastica per colonne. Inoltre, dal momento che il grafo associato alla catena è fortemente connesso e dato che ogni suo nodo possiede un *self-loop*, si deduce che A_c è irriducibile e con elementi strettamente positivi nella diagonale, e quindi é una matrice primitiva. Essendo soddisfatte le ipotesi del teorema di Perron-Frobenius, si può affermare che, facendo evolvere in libera il sistema $\mathbf{x}(n+1) = A_c\mathbf{x}(n)$, a regime lo stato $\mathbf{x}(n)$ sarà allineato con l'autovettore destro di A_c relativo all'autovalore 1, che è proprio la densità di probabilità stazionaria π_{SC}. Questa osservazione è interessante perchè fornisce un metodo di calcolo per trovare π_{SC}. In realtà non è per nulla banale costruire numericamente la matrice quadrata A_c, sia per il suo ordine elevato, che al massimo può essere pari a $2^N 5^{\frac{Nk}{2}}$, sia perchè gli elementi che la costituiscono vanno calcolati con operazioni computazionalmente abbastanza onerose, come si può intuire osservandone le espressioni riportate di seguito:

$$p_{SC_a,\bar{s}_i C_b} = q_i \frac{e^{b\bar{s}_i(m_{ia}-\eta)}}{Z_{ia}} \pi_{C_b}$$

$$p_{SC_a,SC_b} = \left(1 - \sum_{i\in V} q_i \frac{e^{b\bar{s}_i(m_{ia}-\eta)}}{Z_{ia}}\right) \pi_{C_b}$$

dove, per com'è fatta la regola di voto,

$$Z_{ia} = e^{b(m_{ia}-\eta)} + e^{-b(m_{ia}-\eta)}$$

Abbiamo lavorato molto allo scopo di trovare una formula esplicita per π_{SC}, e nel far questo sono sorte delle considerazioni che potrebbero tornate utili in sviluppi futuri del problema.

Innanzitutto va detto che il metodo usato in [1] per trovare la densità di probabilità π_S, ovvero quello basato sul concetto di reversibilità di una catena di Markov, ora non porta a nessun risultato.

In secondo luogo si può notare che

$$\pi_{SC} = \pi_{S_c}\pi_C \tag{7}$$

dove π_{S_c} è la probabilità di ottenere la configurazione S a regime, e π_C è la densità di probabilità di $\hat{c}_{ij} = c_{ij} + c_{ji}, \forall\,(i,j) \in E$. Quest'ultima è determinabile come funzione di N, k e p_e. Tale osservazione si basa sulla formula di probabilità condizionata degli eventi S (essere nella configurazione S del vettore di *trust* stimato) e C (avere una configurazione C dei c_{ij}). Infatti si ha che $\mathbb{P}[SC] = \mathbb{P}[C|S]\mathbb{P}[S]$, dove $\mathbb{P}[C|S] = \mathbb{P}[C]$ poichè la scelta dei c_{ij} è indipendente dai valori degli elementi contenuti in S. Perciò $\mathbb{P}[SC] = \mathbb{P}[C]\mathbb{P}[S]$, che concettualmente ha lo stesso significato della (7).

Un altro aspetto che va messo in evidenza è la netta differenza che c'è tra π_S, calcolata nel caso statico, e π_{S_c}. Quest'ultima densità di probabilità soddisfa la seguente espressione:

$$\pi_{S_c} = \sum\nolimits_C \pi_{SC} \tag{8}$$

Infatti, dal momento che $\mathbb{P}[S]$ può essere scritta come $\mathbb{P}[S] = \sum_C \mathbb{P}[S|C]\mathbb{P}[C]$, dove $\mathbb{P}[S|C]\mathbb{P}[C] = \mathbb{P}[SC]$, si ha che $\mathbb{P}[S] = \sum_C \mathbb{P}[SC]$, che equivale alla (8).

Osservando la (7) e la (8) si può concludere che, se si riuscisse a trovare un'espressione per π_{S_c} o per π_{SC}, allora, usando tali relazioni, se ne ricaverebbe automaticamente una anche per l'altra densità di probabilità.

É chiaro che, anche qualora si trovasse un'espressione per π_{SC}, essa godrebbe di scarsa applicabilità pratica, com'è poi per la π_S. Nonostante ciò un simile risultato potrebbe essere utile per fare un parallelismo con il caso statico.

Fatta questa lunga digressione teorica, è bene capire cosa possa succedere, in pratica, quando si hanno delle variazioni temporali nei c_{ij}. A questo proposito definiamo un terzo parametro di valutazione della regola di voto nel seguente modo:

$$\hat{P}_{correct}(n) = \frac{\sum_{j=1}^{n_{st}} \left[1 - \frac{\|S(n)-T(n)\|_1}{2N}\right]}{n_{st}}$$

dove la lettera n tra parentesi indica che si sta considerando l'ennesima iterazione.

In Fig.15 sono stati riportati gli andamenti di $\hat{P}_{correct}(n)$, $n \in [0, 1000]$, con $b = 1$, e per diversi valori di p_e. Confrontando questo grafico con quello relativo alla variazione di p_e nel caso statico, si nota che i due sono in accordo pressochè perfetto. Si vede ancora che al crescere di p_e la prestazione della regola di voto peggiora, fintantochè, con

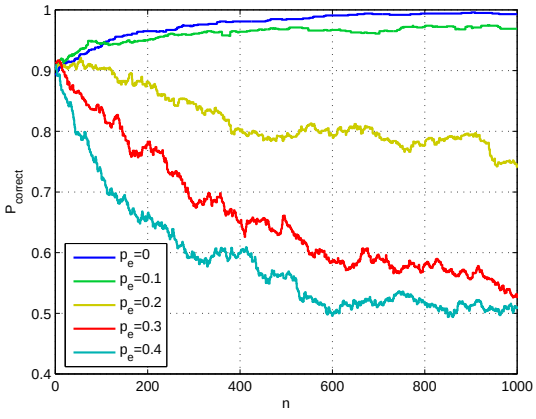

Figura 15. Andamenti di $\hat{P}_{correct}(n)$, $n \in [0, 1000]$, con $b = 1$, $\eta = 0$, $p_g = 0.9$, $p_{rw} = 0$ e per diversi valori di p_e.

$p_e = 0.4$, $\hat{P}_{correct} \simeq 0.5$ per istanti temporali sufficientemente elevati. Ci si potrebbe aspettare un'uguaglianza tra i valori di $\bar{P}_{correct}$ in $b = 1$ nel grafico relativo al caso statico e quelli di $\hat{P}_{correct}$ in $n = 1000$ nel grafico di Fig.15. In realtà ciò non avviene, in quanto $\hat{P}_{correct} \neq \bar{P}_{correct}$ e anche i tempi di simulazione sono diversi.
L'analisi dinamica si presta poi ad altre considerazioni. Si può vedere che inizialmente $\hat{P}_{correct} \simeq 0.9, \forall p_e$, e ciò è corretto se si pensa che è stato scelto uno stato iniziale S con $p_g = 0.9$. In generale si avrà che

$$\lim_{t \to 0^+} P_{correct}(t) \simeq p_{giusto}$$

Inoltre al crescere di p_e si percepiscono discontinuità sempre più marcate nell'andamento di $\hat{P}_{correct}$, ed è giusto che avvenga questo poichè tanto più p_e assume valori elevati, tanto più varia da un istante all'altro lo scenario su cui opera la regola di voto.

B. Rete con T tempo-variante

Un caso probabilmente più interessante, e anche più realistico, si ha quando i c_{ij} variano temporalmente non solo a causa di una $p_e \neq 0$, ma anche per modifiche del vettore reale di *trust*, T, nel tempo. Per simulare tale situazione si devono introdurre due parametri, n_{ch} e p_{ch}. Essi indicano rispettivamente ogni quanto può esserci una variazione di elementi in T, ovvero ogni quanto un sensore può danneggiarsi, i.e. $t_i(n) = 1$, $t_i(n+1) = -1$ o essere aggiustato, i.e. $t_i(n) = -1$, $t_i(n+1) = 1$, e con che probabilità questo avviene. Si suppone che i nodi cattivi diano voti indipendenti ai vicini. In Fig.16 è disegnato l'andamento di $\hat{P}_{correct}(n)$, $n \in [0, 1000]$, con $b = 1$, $p_e = 0.1$, $n_{ch} = 300$ e per due valori di p_{ch}: 0 e 0.05. Si osserva che l'andamento ottenuto con $p_{ch} = 0.05$, a

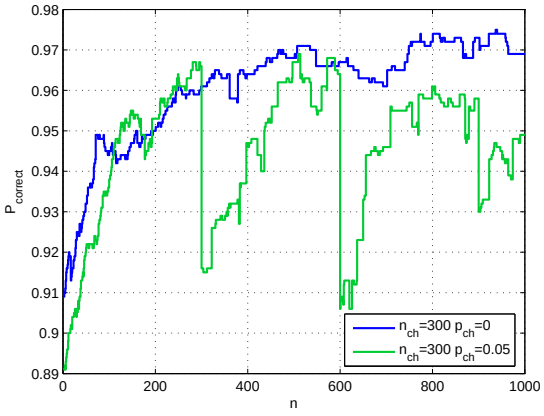

Figura 16. Andamenti di $\hat{P}_{correct}(n)$, $n \in [0, 1000]$, con $b = 1$, $p_e = 0.1$, $\eta = 0$, $p_g = 0.9$, $p_{rw} = 0$, $n_{ch} = 300$ e $p_{ch} = 0.05$.

differenza di quello con $p_{ch} = 0$, decresce rapidamente negli istanti temporali multipli di n_{ch}, per poi risalire più lentamente. Ciò si spiega nel seguente modo. Innanzitutto i parametri b, p_e e p_g sono stati scelti, sulla base dell'analisi condotta fin qui, in modo tale da ottenere una buona prestazione per valori temporali sufficientemente elevati. Inoltre è chiaro che ogni n_{ch} istanti la prestazione cala notevolmente, in quanto vengono modificati con probabilità p_{ch} gli elementi contenuti in T. Questo significa che, se all'istante $(\beta n_{ch} - 1)$, $\beta \in \mathbb{Z}^+$ il vettore S stima bene il vettore T, all'istante successivo, avendo $p_{ch} \neq 0$, la differenza tra i due aumenta sicuramente, causando una notevole diminuzione di $\hat{P}_{correct}$. A questo punto, però, avendo scelto dei buoni parametri, la prestazione della regola di voto tende ad aumentare gradualmente fino all'istante $(\beta + 1)n_{ch} - 1$.
Ovviamente, per riuscire sempre a far fronte alle variazioni di T, i parametri n_{ch} e p_{ch} devono essere scelti sulla base di alcune considerazioni. Infatti, se n_{ch} è troppo piccolo può non esserci tempo sufficiente affinchè $\hat{P}_{correct}$ torni a valori accettabili prima che vi sia un altro cambiamento di T. In altre parole n_{ch} dipende da p_{ch}: tanto più quest'ultimo è grande tanto più tempo serve per colmare la differenza venutasi a creare tra S e T, e viceversa. Inoltre p_{ch} dipende anche dal numero k di vicini di ciascun nodo: tanto più k è grande, tanto più velocemente S converge a T, tanto più ci si può permettere un p_{ch} grande a parità di n_{ch}. Quindi, se n_{ch} e p_{ch} non vengono scelti adeguatamente, può succedere che nel tempo si arrivi ad avere una pessima prestazione della regola di voto, i.e. $\hat{P}_{correct}(n) \simeq 0.5, n > \bar{n}$, come si vede dal grafico in Fig.17. In realtà nelle applicazioni

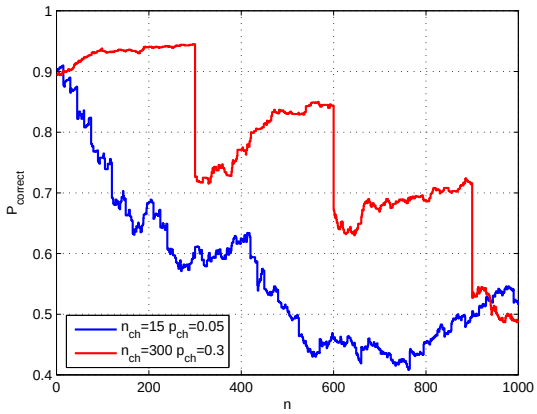

Figura 17. Un esempio di cattiva scelta dei parametri n_{ch} e p_{ch}.

è impossibile che il vettore T cambi in istanti temporali equispaziati, e con probabilità p_{ch} così elevate. É molto più verosimile avere $n_{ch} = 1$ e p_{ch} molto bassa, ad esempio $p_{ch} = 0.0001$. Tale situazione è presentata in Fig.18. Dal momento che n_{ch} è molto piccolo non si vede più un andamento altalenante come quello della curva verde in Fig.16. Tuttavia bisogna continuare a scegliere accuratamente p_{ch}, poichè, pur amentando anche solo di poco il suo valore, le prestazioni della regola di voto tendono a degenerare notevolmente, come si vede osservando il grafico rosso in

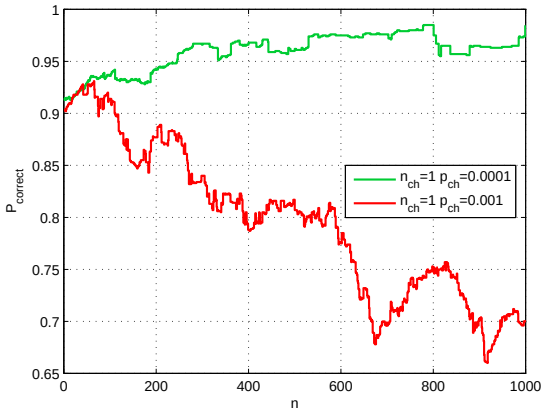

Figura 18. Andamento di $\hat{P}_{correct}$ con $n_{ch} = 1$, $p_{ch} = 0.0001$ e con $n_{ch} = 1$, $p_{ch} = 0.001$.

Fig.18.

C. Rete con ritardi e perdita pacchetti

Finora non si è ancora tenuto conto del fatto che la rete di affidabilità può essere affetta dalla presenza di ritardi aleatori o addirittura di perdita di pacchetti durante lo scambio dell'informazione, ovvero dei voti c_{ij}, tra i diversi sensori. Volendo considerare anche questo aspetto, inizializziamo un vettore $\lambda = [\lambda_0 \lambda_1 \dots \lambda_n]$, dove $\lambda_i = \mathbb{P}[\tau \leq i + 1]$ e τ è il ritardo con cui arriva il pacchetto. Sostanzialmente il vettore λ implementa una distribuzione di probabilità di arrivo dei pacchetti. Ipotizzando che λ abbia dimensione *finita* L_λ, si possono calcolare il ritardo massimo τ_{max} di arrivo di un pacchetto e la probabilità λ_{loss} di perderlo rispettivamente come $\tau_{max} = L_\lambda - 1$ e $\lambda_{loss} = 1 - \lambda_n$. Ovviamente, se $\tau_{max} = 0$ significa che non ci sono ritardi nella rete, e se $\lambda_n = 1$ vuol dire che prima o poi tutta l'informazione arriva a destinazione. Una volta fissata la distribuzione di probabilità λ, si calcolano i ritardi $\tau_{ij}(n)$, $\forall (i,j) \in E$ e $\forall n \in \mathbb{Z}^+$. Ciò significa che il generico voto $c_{ij}(n)$ generato all'istante n ha un tempo di arrivo indipendente da quello di tutti gli altri voti.

A questo punto devono essere introdotte un paio di ipotesi. Innanzitutto supponiamo che ad *ogni* istante n vengano trasmessi *tutti* i voti c_{ij}, $\forall (i,j) \in E$. Il motivo per cui viene introdotta tale ipotesi sarà più chiaro successivamente. Essa è assolutamente necessaria nel caso di ritardi aleatori, ma non nell'eventualità in cui vi sia una rete affetta solo da perdite di pacchetti. Chiaramente questa supposizione è molto forte, in quanto la trasmissione di tutti i voti c_{ij} generati ad ogni istante n comporta un notevole dispendio di risorse della rete, sia a livello di banda occupata, sia a livello di energia con cui vengono alimentati i sensori. In molte applicazioni reali può addirittura essere impossibile soddisfare questa prima ipotesi. Immaginiamo poi di avere dei sensori dotati di memoria, ciascuno dei quali sia cioè in grado di memorizzare i voti c_{ij} che ha generato negli ultimi n_{mem} istanti (escluso quello corrente). Anche questa seconda ipotesi, com'è facilmente intuibile, è necessaria solo nel caso di ritardi aleatori.

Sulla base di tali supposizioni l'algoritmo può essere modificato nel seguente modo. Immaginiamo che all'istante n sia stato scelto il nodo target j-esimo. Ciascun sensore che è in grado di dire qualcosa circa l'affidabilità di j attribuisce a una nuova variabile $\bar{c}_{ij}$ il valore dell'ultimo c_{ij} arrivatogli. Ovviamente, per essere ancora memorizzato nel sensore i-esimo, tale c_{ij} dev'essere stato generato al più n_{mem} istanti precedenti a quello corrente. In caso contrario $\bar{c}_{ij} = 0$. Per facilitare l'implementazione di questa parte dell'algoritmo abbiamo introdotto la variabile γ_k^t, definita come

$$\gamma_k^t = \begin{cases} 1 & \text{se il voto } c_{ij} \text{ generato all'istante } k \text{ è} \\ & \text{arrivato all'istante } t \\ 0 & \text{altrimenti} \end{cases}$$

Ad ogni istante temporale gli ultimi $n_{mem} + 1$ voti c_{ij} memorizzati in ciasuno dei sensori chiamati in causa saranno associati a $n_{mem} + 1$ valori di γ_k^t, ovvero a $\gamma_{t-n_{mem}}^t \dots \gamma_t^t$. Una volta calcolati i $\bar{c}_{ij}$ l'algoritmo procede senza ulteriori modifiche, ma prestando attenzione al fatto che nella regola di voto devono essere considerati i $\bar{c}_{ij}$.

Avendo spiegato per sommi capi le principali modifiche apportate al modello, e soprattutto alla sua implementazione, si possono fare alcune simulazioni. Innanzitutto va considerato il caso più semplice, ovvero quello in cui $\lambda = [\lambda_0]$, corrispondente a una rete in cui non vi sono ritardi aleatori e un pacchetto arriva con probabilità λ_0. Ovviamente in questo scenario non è necessario che ad ogni istante temporale vengano trasmessi tutti i $c_{ij}, \forall (i,j) \in E$, in quanto non bisogna memorizzare nessuno di questi valori. É sufficiente che vengano trasmessi quelli che servono per stimare il *trust* del nodo *target*. In Fig.19 sono tracciati gli andamenti di $\hat{P}_{correct}(n)$ relativi alla situazione appena descritta e per diversi valori di λ_0. Inoltre si suppone $n_{ch} = 1$ e $p_{ch} = 0.0001$, poichè, come spiegato precedentemente, sembrano essere i valori più realistici da attribuire a tali parametri. Si

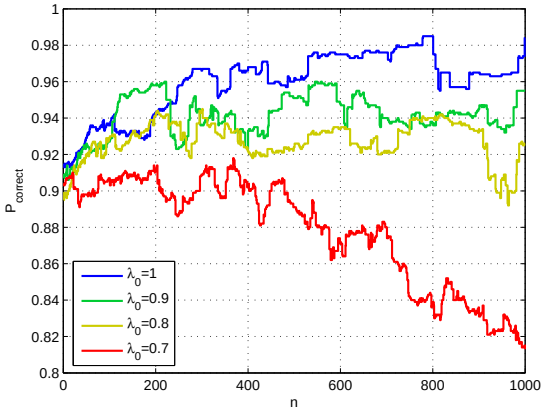

Figura 19. Andamento di $\hat{P}_{correct}(n)$ con $n_{ch} = 1$, $p_{ch} = 0.0001$, $p_e = 0.1$ e per diversi valori di λ_0.

osserva che al diminuire di λ_0 è associato un peggioramento della prestazione della regola di voto. D'altro canto non potrebbe accadere altrimenti, in quanto tanto più piccolo è λ_0, tanta meno informazione giunge a destinazione, e tanto più difficilmente il vettore S riesce a stimare correttamente il vettore T. Sulla base di questi ragionamenti è abbastanza spontaneo pensare che una diminuzione di λ_0 sia equivalente a una diminuzione di k, dove k è il numero di vicini di ciascun nodo. In Fig.20, partendo da questa supposizione, si confrontano due andamenti di $\hat{P}_{correct}$, il primo ottenuto con $k = 4$ e $\lambda_0 = 1$, il secondo con $k = 6$ e $\lambda_0 = 2/3$. Si vede

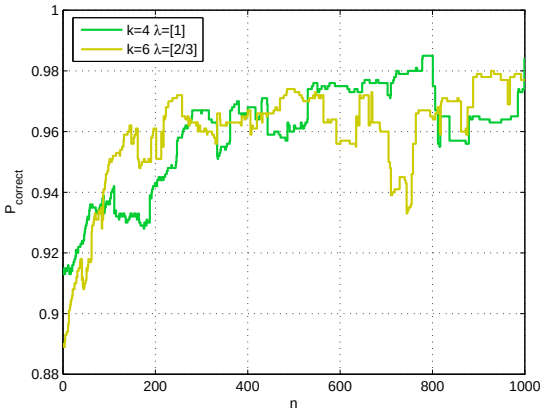

Figura 20. Confronto di due andamenti di $\hat{P}_{correct}(n)$: il primo con $k = 4$ e $\lambda_0 = 1$, il secondo con $k = 6$ e $\lambda_0 = 2/3$.

chiaramente che i due grafici hanno andamenti pressochè uguali, e questo conferma la sostanziale equivalenza tra una diminuzione di k e un'appropriata diminuzione di λ_0.
Volendo complicare leggermente le cose, si possono considerare anche ritardi aleatori nella trasmissione di pacchetti. Per far questo basta inizializzare un vettore λ che abbia almeno dimensione due. Ad esempio, ponendo $\lambda = [0, 0.5, 0.8]$, si ha $\tau_{max} = 2$, e inoltre la probabilità di non avere ritardi è nulla, quella di avere un ritardo di un passo è pari a 0.5, quella di un ritardo di due passi è pari a 0.3 e infine $p_{loss} = 0.2$. In Fig.21 si confrontano gli andamenti di $\hat{P}_{correct}$ per tre diverse distribuzioni dei ritardi, $\lambda = [0.8]$, $\lambda = [0, 0.5, 0.8]$ e $\lambda = [0, 0, 0.8]$, e ponendo $n_{mem} = \tau_{max}$, in modo tale che tutti i pacchetti non persi possano venire considerati. I tre grafici sono abbastanza sovrapposti, e ciò è dovuto essenzialmente al fatto che, avendo $n_{mem} = \tau_{max}$, i loro andamenti dipendono soltanto dalla probabilità di perdita dei pacchetti, p_{loss}, che nel caso in questione è sempre pari a 0.2. In altre parole i sensori hanno memoria sufficiente per tener conto dell'informazione portata da tutti i pacchetti che subiscono un ritardo *finito*, e quindi in questo caso una differenza tra i vari andamenti potrebbe sorgere solo nel caso in cui ai diversi λ fossero associate diverse p_{loss}.
Ovviamente, se viene meno la condizione $n_{mem} = \tau_{max}$, non è detto che distribuzioni λ aventi al stessa p_{loss} diano

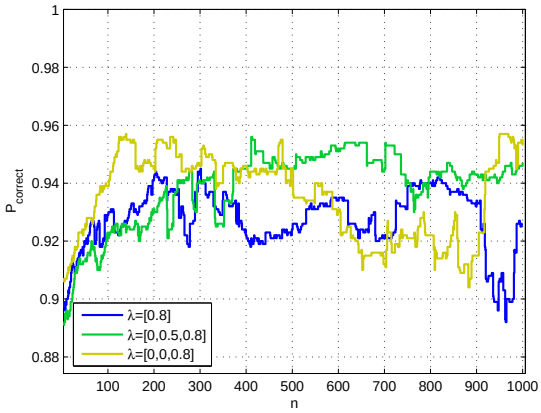

Figura 21. Andamento di $P_{correct}(n)$ con $n_{mem} = \tau_{max}$ e per diversi valori di λ.

origine a grafici i cui andamenti siano simili tra loro. La Fig.22 conferma questa affermazione. Infatti, porre ad

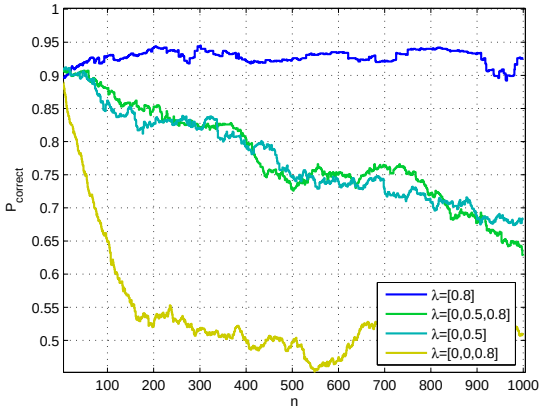

Figura 22. Andamento di $\hat{P}_{correct}(n)$ con $n_{mem} = 1$ e per diversi valori di λ.

esempio $n_{mem} = 1$, vuol dire avere memoria sufficiente per considerare al più solo i voti dati all'istante precedente quello corrente. Così facendo, mentre nel caso di $\lambda = [0.8]$ non ci sono variazioni, se $\lambda = [0, 0.5, 0.8]$ si ottiene un andamento di $\hat{P}_{correct}$ pressochè uguale a quello ricavato con $\lambda = [0, 0.5]$. Infatti in questo caso vanno persi sia il 20% dei voti dovuti a p_{loss}, sia il 30% dei voti che arrivano con due istanti di ritardo, per un totale di metà voti persi. Ovviamente se $\lambda = [0, 0, 0.8]$ tutti i voti vanno persi per cui $\hat{P}_{correct}$ tende a 0.5.

D. Rete con topologia tempo-variante

Infine si potrebbe considerare anche una topologia tempo-variante, ad esempio ricostruendo di tanto in tanto un nuovo modello di rete secondo il *WS-model*, e attribuendo a p_{rw} un valore costante o variabile nel tempo. Questa modifica avvicina ancor più il modello al caso reale, ma non si presta

ad osservazioni particolari che non siano già state prese in considerazione.

VI. Problema della località dell'algoritmo

Considerando la tecnica proposta in [1] in un'ottica implementativa sorgono una serie di problemi. Innanzitutto non è chiaro chi nella rete memorizzi il vettore S di *trust* stimato. Una possibilità è quella di pensare che ogni nodo memorizzi un vettore S e lo aggiorni indipendentemente. Questa soluzione ha l'ovvio svantaggio di occupare molte risorse in termini di memoria, soprattutto se la rete è grande. Inoltre occorre che ad ogni votazione tutti i nodi della rete ricevano i voti dati dai vicini al nodo target per poter fare l'*update*. Questo implicherebbe un notevole utilizzo della rete di comunicazione con conseguente aumento dell'occupazione di banda. Infatti occorre che nel momento in cui viene votato il nodo i-esimo, tutti i nodi della rete ottengano l'insieme $\{c_{ij}+c_{ji},\ j \in N_i\}$, dove N_i è l'insieme dei vicini di i. Sotto queste ipotesi, possiamo considerare una catena di Markov a 2^{N^2} stati, dove uno stato S_c è formato dai vettori di *trust* S_i di ciascun nodo nella rete.

$$S_c = \begin{bmatrix} S_1 \\ S_2 \\ ... \\ S_N \end{bmatrix} = \begin{bmatrix} S_1(1) \ ... \ S_1(N) \\ S_2(1) \ ... \ S_2(N) \\ ... \\ S_N(1) \ ... \ S_N(N) \end{bmatrix}$$

Se ogni nodo j aggiorna indipendentemente il proprio vettore S_j secondo la *voting rule* stocastica definita in [1] nel momento in cui viene votato il nodo i, si ottiene la seguente probabilità di transizione da uno stato S_c a R_c.

$$\mathbb{P}(S_c \to R_c) = \prod_{j=1}^{N} \mathbb{P}(S_j \to R_j) = \prod_{j=1}^{N} \frac{e^{bS_j(i)(m_j(i)-\eta)}}{Z_j}$$

Analogamente

$$\mathbb{P}(R_c \to S_c) = \prod_{j=1}^{N} \mathbb{P}(R_j \to S_j) = \prod_{j=1}^{N} \frac{e^{bR_j(i)(m_j(i)-\eta)}}{Z_j}$$

Quindi

$$\begin{aligned} \frac{\mathbb{P}(S_c \to R_c)}{\mathbb{P}(R_c \to S_c)} &= \frac{\prod_{j=1}^{N} \mathbb{P}(S_j \to R_j)}{\prod_{j=1}^{N} \mathbb{P}(R_j \to S_j)} \\ &= \prod_{j=1}^{N} e^{b((R_j(i)-S_j(i))m_j(i)+\eta(S_j(i)-R_j(i)))} \\ &= \prod_{j=1}^{N} e^{b(U(R_j)-U(S_j))} \end{aligned}$$

Si può perciò definire

$$\pi(S_c) = \frac{\prod_{j=1}^{N} e^{bU(S_j)}}{Z}$$

dove la funzione $U(\cdot)$ rappresenta la funzione energia precedentemente definita, e $\pi(S_c)$ soddisfa la *detailed balance equation* e quindi è proprio la distribuzione stazionaria della catena di Markov. Questo risultato non sorprende affatto: esso traduce in termini matematici il fatto che stiamo considerando N catene di Markov a 2^N stati che evolvono ciascuna indipendetemente dalle altre. Appare quindi evidente che la probabilità di trovare a regime un certo stato aumentato S_c è data proprio dalla produttoria delle probabilità di trovare ciascun sottosistema nello stato S_j, considerato come un sistema isolato.
Naturalmente in questo caso anche l'onere computazionale di calcolare gli aggiornamenti di S pesa su tutti i nodi allo stesso modo e non viene quindi distribuito. D'altra parte essendo a priori tutti i nodi uguali, non ha molto senso pensare che vi sia un nodo deputato a calcolare gli aggiornamenti di S e poi distribuirli, anche perchè di fatto si tratterebbe di una soluzione centralizzata.
Risulta quindi evidente che per avere la località dell'algoritmo è necessario che ogni nodo abbia un suo vettore S, ma come vedremo non è necessario che ogni nodo conosca il valore di *trust* stimato di tutti gli altri (ovvero l'intero S), ma solo di una parte. Proponiamo quindi la seguente soluzione:

- Consideriamo la votazione del nodo i-esimo
- I vicini di i esprimono un voto. Si crea quindi l'insieme $V_{N_i} = \{c_{ji}\ , \forall j \in N_i\}$.
- Il nodo i esprime un voto sui suoi vicini, definendo
$$V_i = \{c_{ij}\ , \forall j \in N_i\}.$$
- Ogni nodo $s \in N_i$ ottiene V_{N_i} e V_i.
- Ogni nodo $s \in N_i$ calcola
$$m_s(i) = \sum_{j \in N_i} (c_{ji} + c_{ij}) S_s(i).$$
- Ogni nodo $s \in N_i$ aggiorna $S_s(i)$ con la voting rule stocastica usando il proprio $m_s(i)$.
- Ogni nodo $s \in N_i$ passa ai suoi vicini $m_s(i)$.
- Sia N_{N_i} l'insieme dei vicini dei vicini di i.
- Ogni nodo $q \in N_{N_i}$ calcola
$$m_q(i) = \sum_{j \in (N_q \cap N_i)} m_j(i) \frac{1}{|(N_q \cap N_i)|}.$$
- Ogni nodo $q \in N_{N_i}$ aggiorna $S_q(i)$ con la voting rule stocastica usando $m_q(i)$

I vantaggi di questa soluzione sono che ogni nodo deve conoscere i valori stimati di *trust* solo dei suoi vicini e dei vicini dei vicini per poter effettuare l'aggiornamento. Questa affermazione si puó dimostrare in modo induttivo: se all'inizio di una iterazione tutti i nodi conoscono i valori stimati di *trust* dei vicini e dei vicini dei vicini, alla fine dell'iterazione tutti i nodi in N_i e in N_{N_i} avranno aggiornato la loro stima del nodo i-esimo, il che verifica l'affermazione. Inoltre la comunicazione avviene solo localmente: quando viene votato il nodo i, solo i nodi in N_i e in N_{N_i} ne vengono interessati, eliminando la necessità di distribuire a tutta la rete gli insiemi V_{N_i} e V_i, come avveniva col metodo precedente.
In analogia con quanto visto nel caso dei c_{ij} tempo-varianti nella sezione V, anche in questo caso si puó dimostrare

l'esistenza di una distribuzione di probabilità stazionaria per la catena di Markov associata al problema in esame. Se pensiamo ad uno stato S aumentato composto dai vettori di *trust* parziali S_q dei vari nodi q, otteniamo infatti di una catena di Markov a stati finiti con al più 2^{N^2} stati. L'uguaglianza vale se e solo se per ogni coppia di nodi i e j , $j \in N_i$ o $j \in N_{N_i}$, ovvero ogni nodo mantiene informazione sull'affidabilità di ogni altro nodo. Poichè per come è implementata la *voting rule* è possibile che ad ogni votazione di un certo nodo target i resti invariato $S_q(i)\ \forall q$, possiamo concludere che ogni stato della catena è dotato di *self-loop*. Inoltre è chiaro che il grafo della catena di Markov è fortemente connesso, poichè comunque si fissino due stati aumentati R e S, la probabilità di passare dall'uno all'altro è certamente positiva e non nulla, a patto di effettuare un numero sufficiente di votazioni. Queste due proprietà permettono di applicare il teorema di Perron-Frobenius alla matrice di transizione di probabilità, garantendo quindi l'esistenza di un autovettore destro che, una volta normalizzato, consiste proprio nella densità di probabilità a regime.
Sfortunatamente nemmeno in questo caso siamo riusciti a trovarne un'espressione esplicita in forma chiusa: questa volta gli N sottosistemi sono accoppiati e non è facile capire come influenzino l'uno l'energia dell'altro.
Naturalmente si potrebbe anche pensare ad una versione più evoluta in cui i nodi $q \in N_{N_i}$ calcolano $m_q(i)$ tenendo presente i valori di $S_q(j)$ per $j \in (N_q \cap N_i)$, ma questo complica ancora di più le cose dal punto di vista della trattabilità matematica.
L'unico problema di questo tipo di approccio è il fatto che ogni nodo $s \in N_i$ deve ottenere V_{N_i} e V_i affinchè l'algoritmo funzioni. Se i vicini di i comunicano tra loro, il problema non si pone, ma se per ricevere V_{N_i} devono passare attraverso i perchè non comunicano direttamente allora si capisce che se il nodo target i è cattivo potrebbe manomettere V_{N_i} prima di fare il *forwarding* ai suoi vicini, compromettendo i risultati della votazione. Chiaramente in questo caso si potrebbero utilizzare tecniche di crittografia per risolvere il problema, ammesso che siano disponibili nel contesto della particolare applicazione considerata. A questo proposito va precisato che non appare banale capire che tipo di manomissione di V_{N_i} sia vantaggiosa per un eventuale nodo i cattivo. Infatti i nodi $s \in N_i$ calcolano $m_s(i)$ come $\sum_{j \in N_i}(c_{ji}+c_{ij})S_s(i)$, dove $S_s(i)$ non è noto al nodo i, che quindi non potrebbe scegliere l'insieme V_{N_i} da forwardare al fine di ottenere l'$m_s(i)$ che desidera.

VII. Tecniche di reputation propagation

In questa sezione ci occuperemo di un problema strettamente connesso a quello trattato finora e noto in letteratura come *reputation propagation and agreement*. Avendo a che fare con sistemi di gestione del *trust management* distribuiti, risulta evidente che, poichè tutte le entità che compongono la rete sono uguali, ogni nodo deve avere un proprio vettore di *trust* stimato S, sia esso relativo solamente ai suoi vicini o all'intera rete. Questo tipo di approccio mette di fronte a due tipi di problemi: prima di tutto in generale i nodi che compongono la rete non concordano sull'opinione che hanno di un certo nodo i; in secondo luogo, molti nodi potrebbero non avere affatto a disposizione una stima dell'affidabilità di i. Risulta quindi interessante analizzare dei metodi che consentano di far ottenere a tutte le entità che compongono la rete il vettore S globale (con la stima di tutti i nodi) e possibilmente far sì che le opinioni che hanno i vari nodi finiscano per concordare. Queste tecniche, ad esempio, potrebbero risultare utili per propagare a tutti i nodi della rete il vettore S, assumendo ad esempio che esso venga aggiornato localmente con i metodi proposti precedentemente in questo lavoro.
Descriviamo prima alcuni risultati presenti in [5] e che per alcuni aspetti verranno usati anche per dimostrare qualche proprietà del nostro lavoro.
Si assume di trattare una rete di N nodi, ognuno dei quali possiede una valutazione del *trust* (o reputazione) di ogni altro nodo della rete.
Si definisce $r_{ji}(t)$ come la reputazione del nodo i sul nodo j all'istante t. Vedendo l'evoluzione del *trust* come un sistema dinamico a tempo discreto, definiamo lo stato del nodo i come un vettore $N-1$ dimensionale $\mathbf{s_i}(t) = [r_{1,i}(t), ..., r_{i-1,i}(t), r_{i+1,i}(t), ..., r_{N,i}(t)]$ evitando di mettere la reputazione di i su se stesso.
Sia $N_i(t)$ l'insieme dei vicini di i all'istante t. Si considera una classe generale di metodi di aggiornamento della reputazione di i su un qualsiasi nodo j, come una somma pesata delle reputazioni di i e dei suoi vicini su j secondo la formula

$$r_{ji}(t+1) = \sum_{k \in N_i(t)} w_{j,ik}(t) r_{jk}(t)$$

dove $w_{j,ik}(t) > 0$ è il peso dato da i al nodo k per il suo voto su j.
Questo modello di aggiornamento è molto generale. In [5] si considerano tre esempi per effettuare l'aggiornamento:

- *Esempio 1*: quando il nodo i riceve lo stato dal nodo k, aggiorna la reputazione su j ($j \neq i, k$) come meda fra la reputazione di k su j e della propria, sempre su j.
- *Esempio 2*: si assume che il nodo i disponga di memoria e salvi gli stati ricevuti in un determinato periodo di tempo. Quindi i aggiorna il proprio stato attraverso una media aritmetica di tutte le reputazioni ricevute.
- *Esempio 3*: molto simile all'esempio 2 a parte il fatto di eseguire una media pesata dei voti ricevuti con i pesi dati da i.

Si scrive quindi l'equazione di aggiornamento in forma matriciale. Definiamo $\mathbf{r_j}(t)$ il vettore colonna delle reputazioni

degli $N-1$ nodi della rete diversi da j sul nodo j:

$$\mathbf{r_j}(t) \triangleq \begin{pmatrix} r_{j,1}(t) \\ \vdots \\ r_{j,j-1}(t) \\ r_{j,j+1}(t) \\ \vdots \\ r_{j,N}(t) \end{pmatrix}$$

Definiamo anche $\mathbf{w_j}(t)$la matrice dei pesi usata per calcolare la reputazione su j. Abbiamo quindi:

$$\mathbf{w_j}(t) \triangleq \begin{pmatrix} w_{j,11} & \dots & w_{j,1k} & \dots & w_{j,1N} \\ \vdots & & \vdots & & \vdots \\ w_{j,i1} & \dots & w_{j,ik} & \dots & w_{j,iN} \\ \vdots & & \vdots & & \vdots \\ w_{j,N1} & \dots & w_{j,Nk} & \dots & w_{jNN} \end{pmatrix}$$

Scritto in forma matriciale l'equazione di aggiornamento risulta:

$$\mathbf{r_j}(t+1) = \mathbf{w_j}(t)\mathbf{r_j}(t) \tag{9}$$

Si mettono assieme tutti i nodi per scrivere un'unica equazione di aggiornamento in forma matriciale. Si definiscono quindi

$$\mathbf{r}(t) \triangleq \begin{pmatrix} \mathbf{r}_1(t) \\ \vdots \\ \mathbf{r}_j(t) \\ \vdots \\ \mathbf{r}_N(t) \end{pmatrix}$$

e

$$\mathbf{w}(t) \triangleq \begin{pmatrix} \mathbf{w}_1(t) & \dots & \mathbf{0} & \dots & \mathbf{0} \\ \vdots & & \vdots & & \vdots \\ \mathbf{0} & \dots & \mathbf{w}_j(t) & \dots & \mathbf{0} \\ \vdots & & \vdots & & \vdots \\ \mathbf{0} & \dots & \mathbf{0} & \dots & \mathbf{w}_N \end{pmatrix}$$

dove $\mathbf{0}$ è una matrice $(N-1)\times(N-1)$ di zeri. Dalle definizioni date sopra si può scrivere l'equazione di aggiornamento globale come

$$\mathbf{r}(t+1) = \mathbf{w}(t)\mathbf{r}(t). \tag{10}$$

L'obiettivo rimane quello di verificare la convergenza dello stato globale $\mathbf{r}(t)$. Come si vede dalle equazioni scritte sopra, l'aggiornamento globale può essere scomposto nell'aggiornamento di ogni singolo nodo separatamente e si vuole che per ogni nodo

$$\lim_{t\to\infty} \mathbf{r_j}(t) = r_j^{ss}\mathbf{1}$$

dove r_j^{ss} è il valore a cui converge la reputazione di j e $\mathbf{1} = [1,1,1,\dots,1]'$ è un vettore $N-1$ dimensionale. Si vuole cioè che tutti i nodi della rete abbiano la stessa opinione riguardo al nodo che si sta considerando, e questo deve valere per ogni nodo.

Per avere convergenza ad un valore significativo, ad ogni aggiornamento la somma dei pesi deve essere pari a 1, ovvero

$$\sum_{k=1}^{N} w_{j,ik}(t) = 1$$

altrimenti si converge al valore zero o infinito se la somma è rispettivamente minore o maggiore di 1. Da questa considerazione si può affermare quindi che la matrice $\mathbf{w_j}(t)$ è stocastica.

Per legare la matrice di aggiornamento all'effettivo scambio di informazioni fra i nodi si rappresenta il flusso di informazioni sul nodo j all'istante t con il grafo di diretto $G_j(t)$ di propagazione. In base al metodo di aggiornamento scelto e ai flussi di informazioni in vari istanti si mettono assieme i relativi grafi in istanti diversi per ottenere un supergrafo $G_j(t_1,t_2)$ come mostrato in Fig.23.

In base alle considerazioni fatte si può dedurre il seguente teorema.

Teorema 6: [*Reputation Agreement*] supponendo di considerare la valutazione del generico nodo j, se per ogni istante t_1 esiste un istante $t_2 > t_1$ tale che il supergrafo $G_j(t_1,t_2)$ sia fortemente connesso con *self-loop*, allora il sistema converge. Si ha cioè:

$$\lim_{t\to\infty} \mathbf{r_j}(t) = r_j^{ss}\mathbf{1} \tag{11}$$

La dimostrazione di questo teorema si basa sulla teoria del consenso descritta in [8].

A. Implementazione per WSN.

Implementiamo quindi una tecnica per il *trust management* di una rete che si basa sulle idee proposte in [5] e sopra accennate.

Per fare questo dobbiamo definire le caratteristiche dell'algoritmo in modo da soddisfare i vincoli di una *wireless sensor network*. Inoltre in [5] non viene dato un significato reale ai parametri che vengono usati lasciando libero spazio alle interpretazioni. Decidiamo quindi di definire anche questo lato della procedura.

Diamo nel seguito una descrizione di tutti i passaggi e delle risorse che deve usare ogni nodo al fine di implementare questa tecnica.

Modellizziamo la rete attraverso un grafo $G(V,C)$ non orientato in cui i nodi della rete sono i vertici del grafo e gli archi indicano la possibilità dei due nodi di comunicare fra loro. Si fa l'ipotesi che il numero di nodi sia $|V| = N$ e che il grafo sia fortemente connesso in modo che non ci siano blocchi isolati e l'informazione possa circolare attraverso la rete.

Per semplicità supponiamo che possa trasmettere un nodo alla volta in tutta la rete e che questo nodo venga scelto a caso. Quest'ipotesi è abbastanza realistica in quanto il canale trasmissivo è una risorsa comune e può essere usato

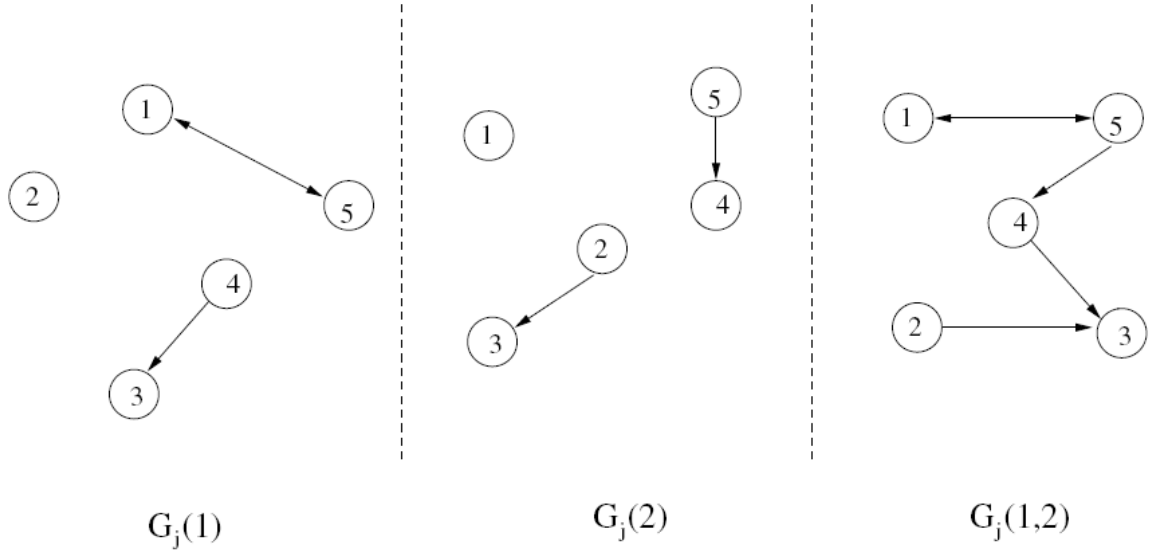

Figura 23. Illustrazione del grafo di propagazione e del supergrafo.

da un solo nodo alla volta e anche il metodo di accesso al mezzo è in genere abbastanza casuale.
Anche il ricevitore viene scelto a caso fra i vicini del nodo che trasmette. Un'alternativa a questa procedura potrebbe essere quella di scegliere il ricevitore tra i vicini a cui puó trasmettere il nodo, in maniera proporzionale alla reputazione che il nodo ha su di essi. Oppure si può scegliere di escludere dalla trasmissione i nodi vicini considerati cattivi. Quest'ultima potrebbe essere una procedura per escludere i nodi cattivi dalla rete.

I nodi della rete possono essere sia buoni che cattivi. Descriviamo lo stato di affidabilitá reale della rete attraverso il vettore $T = [t_1, \ldots, t_N]$ ponendo $t_i = 1$ se il nodo è buono e con $t_i = 0$ in caso contrario. Decidiamo di fornire inizialmente ogni nodo i con l'opinione r_{ji} dei propri vicini. Quest'opinione si può assimilare ai c_{ij}, cioè ai valori di confidenza usati nelle sezioni precedenti. Come già accennato, questa valutazione si può ottenere con varie tecniche che dipendono dalla particolare applicazione considerata, ma in generale i voti c_{ij} si ottengono osservando se i nodi j, vicini di i, si comportano correttamente.
Decidiamo quindi di dare un valore $r_{ji} \in [0.5, 1]$ se il nodo i valuta come *buono* il nodo j, mentre diamo un valore $r_{ji} \in [0, 0.5)$ se accade il contrario. Questa valutazione è quasi certamente affetta da errore. Definiamo quindi la probabilità di valutare in maniera errata il vicino come

$$p_e = \mathbb{P}[round(r_{ji}) \neq t_j] \quad \forall i$$

dove con $round(r_{ji})$ si intende l'arrotondamento al valore intero più vicino.
In principio avremo quindi una rete di N nodi ognuno dei quali avrà un vettore $\mathbf{s_i}$ contenente i voti dati ai vicini. Con l'utilizzo di questo algoritmo si vuole che ogni nodo venga valutato allo stesso modo da tutti i suoi vicini e che questa valutazione (comune a tutti) sia corretta. Per la correttezza della valutazione si vuole che il voto al nodo buono converga ad un valore $r_j \in [0.5, 1]$ e quello di un nodo cattivo ad un valore $r_j \in [0, 0.5)$ Scriviamo di seguito le operazioni effettuate ad ogni iterazione.

- Si sceglie il nodo che trasmette (n_{TX}) e quello che riceve (n_{RX}).
- Il $nodo_{TX}$ trasmette a $nodo_{RX}$ anche il proprio vettore $\mathbf{s}_{n_{TX}}$. Questo vettore contiene inizialmente solo il voto dato ai vicini, ma con il passare delle iterazioni contiene anche le opinioni di nodi che non sono a lui vicini.
- Il $nodo_{RX}$ ha a disposizione sia il proprio vettore di *trust* e quello ricevuto dal $nodo_{TX}$. Effettua quindi una media fra il proprio vettore e quello ricevuto, pesata con il valore 1 sul proprio vettore e con il valore $r_{n_{TX},n_{RX}}$ sul vettore ricevuto. Per il voto sul nodo j si ottiene quindi:

  $$r_{j,n_{RX}}(t+1) = \frac{r_{j,n_{RX}}(t) + r_{n_{TX},n_{RX}}(t) r_{j,n_{TX}}(t)}{1 + r_{n_{TX},n_{RX}}(t)}$$

  Se una componente del proprio vettore è nulla mentre quella del $nodo_{TX}$ è diversa da zero, $nodo_{RX}$ memorizza il valore ricevuto senza effettuare medie.
  Il nodo ricevente non aggiorna la componente del vettore *trust* relativa al nodo da cui ha ricevuto il vettore $r_{n_{TX}}$.
- Si ripetono queste operazioni.

Fra le operazioni appena elencate, per avere la convergenza risulta essenziale memorizzare anche le informazioni che non riguardano i propri vicini. Ciò deriva dal fatto che per avere la convergenza ad una stessa opinione, i vicini di un determinato nodo devono condividere fra loro le

informazioni sullo stesso. Perché questo avvenga, se non sono vicini fra loro, devono far transitare l'informazione attraverso altri nodi.

Sulla base delle ipotesi fatte precedentemente, possiamo ricreare le matrici $\mathbf{r}(t)$ e $\mathbf{w}(t)$. In quest'ultima matrice, quasi tutte le componenti saranno nulle, infatti ad ogni istante t trasmette solo un nodo. Solo la componente relativa al nodo trasmettitore e a quello ricevitore risulterà diversa da zero. Guardiamo uno specifico nodo. Eseguire varie iterazioni secondo l'equazione di aggiornamento (9) equivale ad effettuare una sequenza di moltiplicazioni per ottenere:

$$\mathbf{r_j}(t+1) = \mathbf{w_j}(t)\mathbf{w_j}(t-1)\dots\mathbf{w_j}(0)\mathbf{r_j}(0) = \mathbf{W_j}\mathbf{r_j}(0)$$

Come risulta dal teorema 6 e dalla teoria descritta in [5] e in [8], se alla matrice $\mathbf{W_j}$ è associabile un grafo fortemente connesso, si ha convergenza al valore $r_j^{ss}\mathbf{1}$ come visto in (11).
Per avere un riscontro pratico dei risultati teorici descritti effettuiamo varie simulazioni. Osserveremo la bontà della tecnica su diverse topologie di reti e per parametri che rendono il modello una buona approssimazione di una WSN. Per misurare le prestazioni definiamo alcune grandezze di interesse:

- *Scarto dalla media* per un determinato nodo secondo la formula

$$v_j(t) = \sum_{i \neq j} (r_{ji}(t) - m_j(t))^2 \qquad (12)$$

  dove $m_j(t)$ è la reputazione media del nodo j all'istante t. Con questa misura si osserva la velocità di convergenza ad una stessa opinione.
- *Probabilità che il vicino sia valutato correttamente*. Per fare questo calcolo consideriamo solo i nodi realmente vicini fra loro. Contiamo quindi le valutazioni errate dei vicini e dividiamo per il numero di opinioni totali di nodi fra loro vicini.

$$\begin{aligned} P_{correct}(t) &= 1 - \frac{\sum_i \sum_{j \in N_i} |round(r_{ij(t)}) - t_i|}{\sum_i |N_i|} \\ &= 1 - \frac{\text{numero di voti sbagliati}}{\text{numero totale di voti}} \end{aligned}$$

  dove N_i è l'insieme dei vicini di i e $|N_i|$ è la sua cardinalità. Le opinioni di nodi che non sono vicini fra loro non vengono contate in quanto non interessano l'effettiva valutazione data al nodo. La motivazione di questa scelta viene data con un esempio: un nodo può decidere a chi trasmettere in base alla valutazione che dà ad un'altro nodo; se però il nodo non è un suo vicino non potrà mai comunicare con lui anche se la valutazione è buona. Le valutazioni a nodi che non sono vicini non influiscono sulle comunicazioni, da cui la scelta di non considerarli nella misura di $P_{correct}$.
- *Numero di errori a regime*: é il numero di nodi valutati in maniera errata considerando l'avvenuta convergenza ad un valore comune.

$$\text{numero di errori} = \sum_j |(round(r_j^{ss}) - t_j|$$

L'evoluzione del sistema dipende dalle condizioni iniziali e dalla sequenza di aggiornamento, che sono generate in modo casuale. Per ottenere una misura media delle prestazioni decidiamo di mediare su varie realizzazioni del processo. Ad ogni prova variano le condizioni iniziali sui reciproci voti r_{ji} e la sequenza dei nodi che si vanno ad aggiornare. Vedremo quindi che otterremo i risultati desiderati, soprattutto per quanto riguarda la convergenza ad un valore comune della reputazione di un nodo, ma in buona parte anche per quanto riguarda la stima del *trust* globale della rete. I risultati dipenderanno comunque molto dalla topologia usata e dalla connettività della rete.

B. Simulazioni.

Abbiamo simulato con MATLAB l'evoluzione della rete per ottenere il *trust* dei vari nodi secondo la tecnica sopra descritta.
Innanzitutto bisogna decidere la topologia della rete che si vuole usare. Consideriamo reti in cui i nodi sono posizionati a scacchiera su una superficie e comunicano solo con gli immediati vicini (vedi Fig.24) e reti in cui i nodi sono distribuiti casualmente su una determinata area (1×1) e i collegamenti avvengono fra nodi la cui distanza è inferiore ad una certa soglia r (vedi Fig.25). Per semplicità chiameremo la prima *lattice 2-D*, essendo la stessa usata anche nelle sezioni precedenti, e la seconda *rete geografica*. I parametri

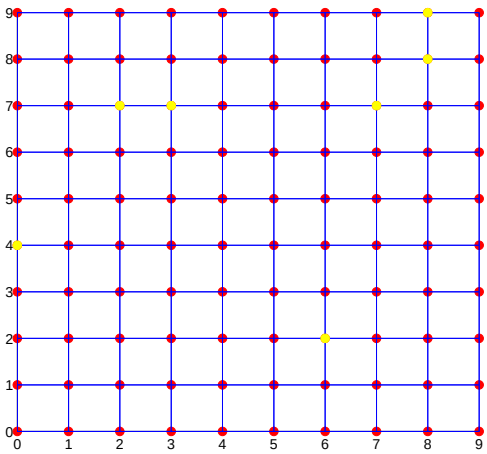

Figura 24. Lattice 2-D.

simulativi associati alla topologia sono il numero di nodi N, l'estensione della rete e il raggio r per i collegamenti.
Tra gli altri abbiamo il valore p_m, che indica la probabilità che un nodo sia *cattivo*, il valore p_e, che indica la probabilità che la valutazione iniziale del nodo sia sbagliata, e i vari parametri temporali per la simulazione e per il calcolo di

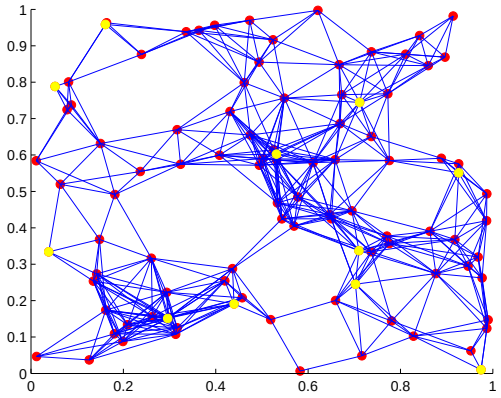

Figura 25. Rete geometrica dipendente dalla distanza.

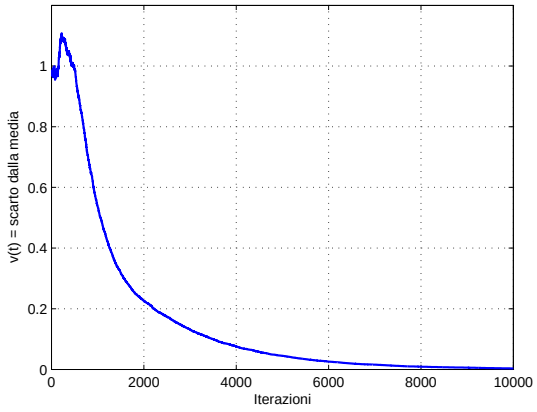

Figura 26. Scarto dalla media dei voti per uno specifico nodo.

medie.
Si assume che il comportamento dei nodi cattivi sia lo stesso dei nodi buoni, cioè votano i vicini in maniera corretta con la stessa probabilità di errore p_e. In alternativa si potrebbero avere nodi che votano appositamente in maniera errata, o che non svolgono la loro funzione. Su queste opzioni faremo alcune considerazioni nel seguito.
Per ottenere una media dei risultati verranno eseguite 50 prove, ciascuna delle quali consiste in 10000 iterazioni. Questi valori dei parametri si sono rivelati più che adatti alla nostra situazione perchè si è visto che, anche aumentando il numero di prove, i risultati non subiscono variazioni significative.
Di seguito verranno elencate alcune simulazioni che forniscono risultati interessanti che si prestano a varie considerazioni.

B.1) Rete di partenza.

Impostiamo $p_m = 0.1$ e $p_e = 0.1$ su una rete geografica di 100 nodi collegati per distanze inferiori a $r = 0.3$. Queste condizioni possono essere considerate di riferimento in quanto i valori dei parametri sono quelli che mediamente si trovano in realtà. Anche la topologia usata è abbastanza simile a quelle che si trovano in realtà per le reti di sensori. Verifichiamo quindi la convergenza ad un valore comune per quanto riguarda il voto dato ad un determinato nodo. In Fig.26 è mostrata l'evoluzione dello scarto dalla media definito dall'equazione (12) per uno specifico nodo scelto casualmente. Si nota come dopo 3000 iterazioni lo scarto sia molto piccolo. Ricordiamo che nel valutare lo scarto consideriamo tutti i nodi, anche quelli che non sono effettivamente vicini del nodo che stiamo valutando. Ciò significa che effettivamente gli $N - 1$ nodi che valutano il nodo scelto danno un voto comune. La speranza è che quest'ultimo sia quello esatto, cioè che se il nodo è buono venga valutato come buono, e viceversa. Dalla Fig.27 si nota come il valore della $P_{correct}$ si stabilizzi, dopo circa 3000 iterazioni, ad un valore più elevato di quello di partenza, che è legato alla p_e. Infatti inizialmente $P_{correct} = 1 - p_e$, ma applicando poi l'algoritmo che media i voti fra i vicini, si riesce a correggere qualche errore, soprattutto se è isolato. Per questa simulazione mostriamo

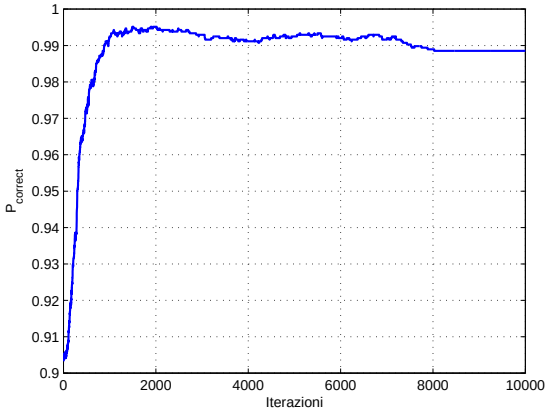

Figura 27. $P_{correct}$.

anche un grafico che rappresenta il numero di nodi valutati in maniera errata in ognuna della 50 prove effettuate. Nel caso peggiore si hanno 6 errori ma ci sono anche casi in cui non si hanno errori. Mediamente, comunque, si verificano circa 2 errori su 100 nodi da valutare. Il numero di errori è legato alla $P_{correct}$ ma non in maniera proporzionale, in quanto la $P_{correct}$ considera tutti gli archi di valutazione e dipende quindi dal numero di vicini del nodo che viene valutato in maniera errata, e non solo dal numero di errori di valutazione a regime.

B.2) Confronto fra varie topologie

La topologia della rete, e soprattutto il grado di connettività, legato al coefficiente di raggruppamento, influenzano le prestazioni dell'algoritmo per quanto riguarda la velocità di convergenza e in parte anche per la $P_{correct}$. In reti con un grado di connettività alto, in cui ogni nodo ha molti

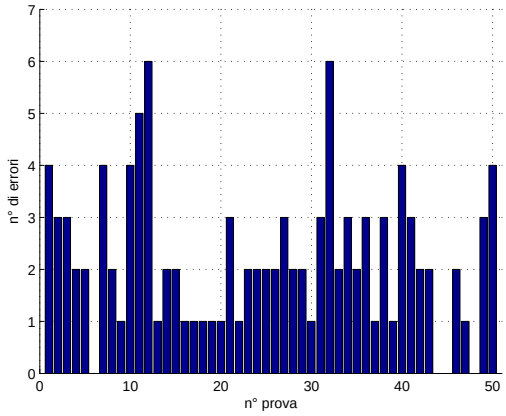

Figura 28. Numero di errori sui nodi a regime.

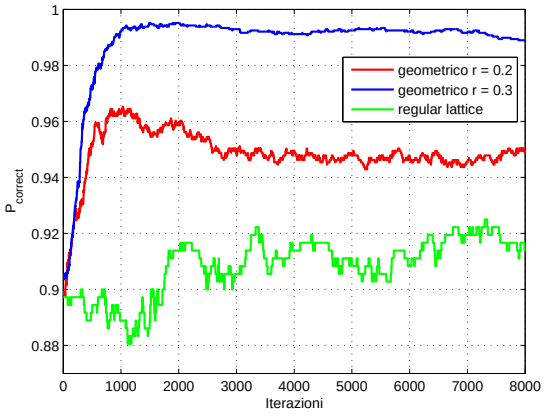

Figura 30. confronto della $P_{correct}$ per tre diverse reti.

vicini, pur avendo uno scarto iniziale piú grande dovuto al maggior numero di opinioni, si ha una convergenza più veloce rispetto a grafi in cui i collegamenti sono ridotti.
A conferma di ció osserviamo la Fig.29. Sono mostrati gli scarti in funzione dell'iterazione per tre diverse reti. Tutte e tre hanno lo stesso numero di nodi, due sono reti geografiche in cui varia il raggio per la connessione ($r = 0.3$ e $r = 0.2$), e una è il *lattice 2-D*. Per dare un'idea del grado di connettività, abbiamo circa 2200 archi diretti per la prima rete, circa 1000 per la seconda e 360 per quella a scacchiera. Si vede bene come la velocità di convergenza diminuisca al diminuire dei collegamenti. Avendo più collegamenti la distribuzione dell'informazione avviene in maniera più veloce e viceversa. Nel grafico si nota come in una prima

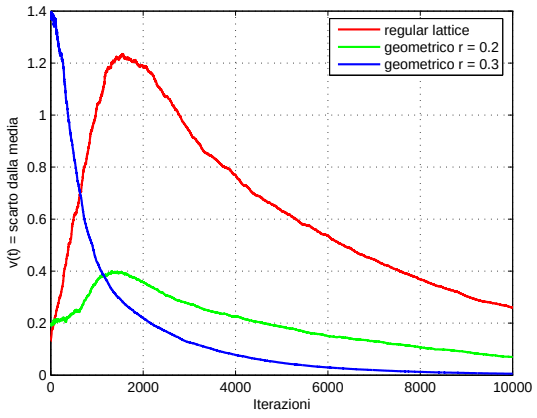

Figura 29. Confronto dello scarto dalla media dei voti per tre diverse reti.

fase lo scarto aumenti. E' giusto che ciò accada in quanto le opinioni su un determinato nodo si distribuiscono anche a nodi che non sono vicini di quello considerato e, come già detto, lo scarto considera tutti i nodi della rete.
Si ha una variazione anche nel valore di $P_{correct}$. Partendo dalle stesso condizioni iniziali su p_e si ha un peggioramento delle prestazioni per l'algoritmo applicato ad una rete meno connessa. Intuitivamente, essendo che l'algoritmo effettua medie fra le opinioni su uno stesso nodo, é chiaro che, avendo meno opinioni, un solo errore può portare ad una valutazione complessiva del nodo errata. Il grafo di Fig.30 mostra proprio la differenza di $P_{correct}$ per le tre topologie considerate in questo paragrafo.

B.3) Confronto con variazioni di p_e

Le prestazioni della tecnica che stiamo usando sono fortemente influenzate dal valore del parametro p_e. Ricordiamo che p_e rappresenta la probabiltà che nella condizione iniziale il voto che un nodo dá al vicino sia errato. All'aumentare di p_e si ha quindi un degrado delle prestazioni.
Come si può notare osservando la Fig.31, per probabilità di errore basse si arriva ad avere un errore complessivo nullo. All'aumentare di p_e il miglioramento della $P_{correct}$ che si ottiene con questa tecnica diminuisce fino a causare, per valori di $p_e \approx 0.5$, un peggioramento della stima a regime rispetto a quella di partenza.

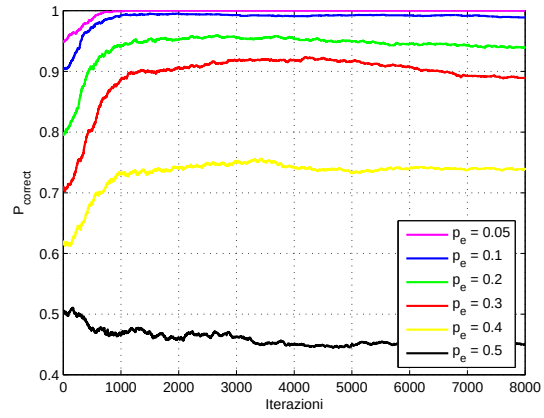

Figura 31. Confronto della $P_{correct}$ per diverse probabilità di errore p_e.

B.4) Confronto con variazioni di p_m

Come già accennato in precedenza, considerando il comportamento dei nodi cattivi uguale a quello dei nodi buoni, il valore p_m, che indica la probabilità che un nodo sia cattivo, non implica variazioni nelle prestazioni dell'algoritmo. Essendo il comportamento uguale per tutti i nodi, una variazione di p_m comporta solo una variazione delle condizioni iniziali per renderle congruenti con la rete. Il valore di $P_{correct}$ viene calcolato tenendo conto della vera condizione del nodo, e quindi una variazione dovuta a p_m è controintuitiva. La Fig.32, che mostra l'andamento di

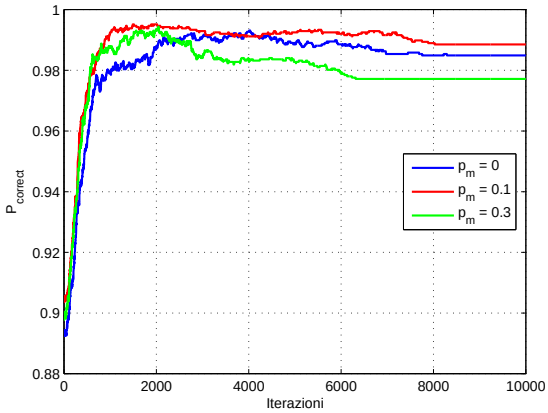

Figura 32. Confronto della $P_{correct}$ per diverse probabilità p_m di avere nodi cattivi.

$P_{correct}$ per diversi valori di p_m, conferma le affermazioni fatte.

B.5) Diverso comportamento dei nodi cattivi.

Consideriamo ora il caso in cui i nodi cattivi si comportano in maniera diversa da quelli buoni.
Le modalità in cui potrebbero operare i nodi cattivi sono numerosissime e, nella realtà, dipendono da tanti fattori. Il principale è il fatto che il nodo può essere *cattivo* perchè si è guastato oppure perchè programmato per creare disagi. Consideriamo il caso in cui il nodo cattivo dà una valutazione appositamente sbagliata al vicino. L'influenza si ha quindi solo sulla condizione iniziale in quanto, per quanto riguarda la procedura da eseguire, anche i nodi cattivi operano allo stesso modo di quelli buoni. La Fig.33 mostra un pesante degrado delle prestazioni all'aumentare di p_m. Le considerazioni per questo caso sono le stesse fatte per variazioni di p_e, in quanto il comportamento scelto influisce solo sulla condizione iniziale. Un aumento di p_m equivale ad un aumento di p_e.

Pur avendo inizialmente affrontato lo studio di queste tecniche di *repuatation propagation* con l'intento di integrarle con quanto visto nelle sezioni precedenti, alla fine si é rivelata una possibile alternativa per risolvere il problema di *trust management*.

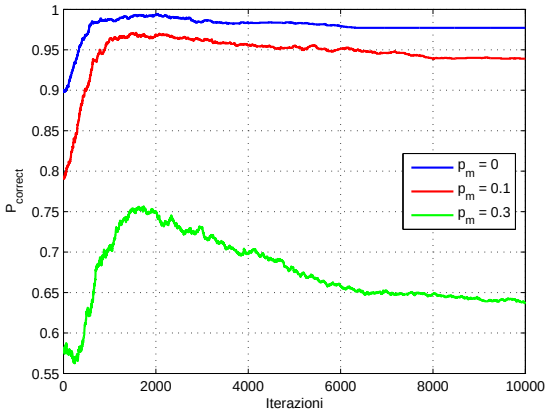

Figura 33. Confronto della $P_{correct}$ per diverse probabilità p_m di avere nodi cattivi.

VIII. Conclusioni e sviluppi futuri

Volendo affrontare il problema di *trust management* distribuito in reti wireless di sensori (WSN) anche da un punto di vista teorico, inizialmente abbiamo preso spunto dal lavoro proposto in [1], che sembra essere molto promettente in tal senso. In esso, infatti, si pongono delle fondamenta teoriche molto solide su cui viene poi costruito un algoritmo di *trust management*. Tale approccio è senza dubbio molto raro in un contesto in cui le soluzioni proposte sono solitamente testate solo per via simulativa.
A partire da uno studio approfondito dell'algoritmo proposto in [1], abbiamo ricavato dei risultati che ribadiscono l'importanza di un'analisi teorica nel giudicare le prestazioni di algoritmi atti a valutare l'affidabilità delle entità componenti una rete generica. In altre parole, il nostro lavoro conferma il fatto che, per quanto un approccio simulativo nell'affrontare il problema di *trust management* possa essere importante, bisogna sempre avere la massima cautela nell'analizzare i valori cosí ottenuti.
In primo luogo, avendo trovato risultati teorici importanti almeno per alcune classi di parametri ($\eta = 0$), siamo riusciti a capire più a fondo quali siano le prestazioni asintotiche dell'algoritmo. In questo modo ci siamo resi conto che i risultati simulativi proposti in [1] sono in completo disaccordo con quanto previsto dal teorema 2. Cercando di capire il motivo di questa contraddizione, abbiamo potuto apprezzare anche i limiti dell'analisi asintotica da noi condotta, rendendoci conto del fatto che in molti casi concreti sarebbero necessari tempi non realistici per ottenere risultati in accordo con quelli teorici. Un'indagine così approfondita mette in guardia sulla complessità della dinamiche che si instaurano nell'algoritmo proposto in [1], rendendo ancora più evidente la necessità di comprendere chiaramente qual è il contesto in cui esso verrà applicato, in modo tale da poter decidere di volta in volta i valori appropriati da assegnare ai parametri in gioco che, come abbiamo visto, influenzano molto le prestazioni del sistema.

A tal proposito, nella sezione III, ipotizzando che siano soddisfatte le condizioni per cui è possibile confrontarsi con i risultati teorici, ovvero supponendo che si abbia a che fare con reti piccole o si abbiano a disposizione tempi di vita molto lunghi, abbiamo notato come scelte opportune dei parametri garantiscano ottime prestazioni asintotiche.
Lo studio simulativo che abbiamo intrapreso ci ha poi aiutati a capire come può comportarsi l'algoritmo in applicazioni reali. A questo proposito abbiamo apportato anche alcune modifiche al modello simulativo, in modo da ottenerne uno che fosse più attinente alla realtà. Dapprima abbiamo approfondito osservazioni già fatte in [1], come l'introduzione di p_e e di p_m, le variazioni di p_{rw} e di k; in un secondo momento abbiamo apportato delle nostre modifiche al modello, considerando ad esempio la possibilità di avere ritardi aleatori o perdita di pacchetti nella rete di affidabilità. Ovviamente questi nuovi aspetti ci hanno spinti ad affiancare all'analisi statica, in parte già proposta in [1], anche un'analisi dinamica dell'algoritmo. In quest'ultimo scenario è quindi stato possibile studiare anche ciò che accade quando alcuni parametri, come i c_{ij} o il vettore di *trust* reale T, variano nel tempo. Nel complesso l'approccio simulativo da un lato ci ha permesso di confermare, per lo meno da un punto di vista intuitivo, i risultati teorici asintotici, e dall'altro ha messo in luce l'influenza di nuovi parametri sui risultati, come lo stato iniziale del vettore S e i tempi di simulazione n^* ed M. Anche in questo caso, tuttavia, abbiamo visto che scelte opportune di tali parametri garantiscono performance molto buone.
Studiando poi l'effettiva implementabilità dell'algoritmo sono sorte alcune difficoltà, principalmente imputabili alla grande generalità e all'eleganza matematica della trattazione di [1], ovvero proprio agli aspetti che, a nostro avviso, costituiscono i punti di forza di quell'approccio rispetto agli altri presenti in letteratura. Nella sezione VI abbiamo visto che, sebbene non sia immediato e banale integrare con reti realmente distribuite l'algoritmo di *trust management* proposto in [1], quest'ultimo ha una struttura portante che è effettivamente locale e quindi dovrebbe essere possibile implementarlo in un'applicazione reale. In questo contesto, nella sezione VII, abbiamo preso in considerazione tecniche di *reputation propagation and agreement*. Queste tecniche prevedono il raggiungimento di un valore comune a tutti i nodi della rete (*consensus*) per quanto riguarda la valutazione dell'affidabilità di ogni altra entità. Per integrare le due tecniche viste si potrebbe pensare di eseguire il primo algoritmo per un determinato periodo, dotando ogni nodo di un vettore di *trust* stimato S, e successivamente eseguire l'algoritmo proposto nella sezione VII per arrivare ad un'opinione comune fra i vari nodi, ottenendo così un solo vettore S comune a tutta la rete. La prima difficoltà che si trova nel cercare di integrare l'algoritmo proposto in [1] e questa tecnica è il fatto di considerare diversi possibili valori di *trust*: nella prima si usa un valore discreto $s_i \in \{1, -1\}$ mentre nell'altra gli r_{ij} assumono valori continui in $[0, 1]$. Già questo aspetto pone dei grandi limiti ad una possibile integrazione delle due tecniche, soprattutto perchè l'*agreement* consiste in una media dei valori di partenza, che mal si adatta al caso di valori discreti. Abbiamo deciso quindi di considerare la tecnica proposta in [5] come una tecnica di *trust management* a sè stante. Effettuando varie simulazioni si è visto che questa tecnica porta buoni risultati se la rete è molto connessa (diametro medio piccolo) e se l'errore iniziale è molto piccolo, come mostrato nelle figure 30 e 31; in caso contrario le prestazioni degradano abbastanza velocemente risultando così sicuramente peggiori di quelle viste nelle sezioni IV e V. D'altra parte, un indubbio vantaggio di questo approccio é sicuramente quello di essere piú facilmente implementabile.

Ovviamente, come si é giá potuto intuire leggendo le sezioni precedenti, c'é da lavorare ancora molto, anche nell'ambito della ricerca, per quel che riguarda il *trust management* distribuito.
A nostro parere, ad esempio, tra i possibili sviluppi futuri sarebbe molto utile cercare di trovare un'espressione esplicita per descrivere la distribuzione di probabilità stazionaria π_{SC}, ovvero quella che si ottiene quando i voti c_{ij} sono tempo-varianti, di cui abbiamo giá dimostrato l'esistenza nella sezione V. Tale risultato, infatti, permetterebbe di studiare le prestazioni teoriche asintotiche anche nel caso di c_{ij} tempo-varianti, rendendo possibile anche dei confronti con gli analoghi risultati trovati nel caso statico.
Altrettanto importante risulta, a nostro avviso, uno studio più approfondito delle tecniche proposte in VI, allo scopo di capire meglio il problema della reale implementabilità dell'algoritmo proposto in [1].
Un altro aspetto che merita sicuramente di essere approfondito in futuro riguarda l'utilizzo di topologie di reti di affidabilitá diverse da quella considerata in questa sede, ottenuta con il *WS model*. Si potrebbe pensare, ad esempio, di applicare il solito algoritmo di *trust management* su una rete implementata secondo il modello formulato da Klemm ed Eguíluz e descritto in [18], il quale, rispetto al *WS model*, é sicuramente piú attinente alla realtá ma nel contempo molto piú complesso. Uno degli scopi di questo studio sarebbe quello di capire quali siano le topologie di rete concretamente realizzabili cui corrispondono le migliori prestazioni dell'algoritmo.
Infine, potrebbe essere utile modificare il sistema di *trust management* in modo tale da escludere dalla rete i nodi sulla base dell'evoluzione temporale del loro stato di affidabilitá.

RINGRAZIAMENTI

Vogliamo ringraziare il prof. John Baras e Tao Jiang per la grande disponibilità che ci hanno dimostrato nel discutere le questioni da noi sollevate riguardanti il metodo proposto in [1].

RIFERIMENTI BIBLIOGRAFICI

[1] John Baras, Tao Jiang. *Trust Evaluation in Anarchy: A Case Study on Autonomous Networks*. Infocom ,2007.
[2] John Baras. *Security and Trust for Wireless Autonomic Networks*. European Journal of Control ,2007.
[3] Paul Coddington. *Montecarlo Simulations for Statistical Physics*. Lecture Notes
[4] Wolfhard Janke. *Statistical Analysis of Simulations: Data Correlations and Error Estimation* Quantum Simulations of Complex Many-Body Systems:From Theory to Algorithms, John von Neumann Institute for Computing
[5] Yanbin Liu, Yang Richard Yang. *Reputation Propagation and Agreement in Mobile Ad-Hoc Network*. Wireless Communications and Networking, WCNC 2003.
[6] James P. Sethna. *Introduction to the Ising Model* Cornell University, http://www.lassp.cornell.edu/sethna/sethna.html
[7] Barry A. Cipra. *The Ising Model is NP-Complete*. SIAM News , Volume 33, Number 6
[8] Luca Schenato. *Appunti delle lezioni*. http://www.dei.unipd.it/∼schenato
[9] Fabio Pasqualetti, Antonio Bicchi, Francesco Bullo. *Distributed Intrusion Detection for Secure Consensus Computations*. Proceedings of the 46th IEEE Conference on Decision and Control New Orleans, LA, USA, Dec. 12-14, 2007.
[10] Grant A. Jacoby, Nathaniel J. Davis.*Mobile Host-Based Intrusion Detection And Attack Identification*. IEEE Wireless Communications, August 2007.
[11] Sencun Zhu, Sanjeev Setia, Sushil Jajodia. *LEAP: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks*. CCS'03, Washington, DC, USA.
[12] Levente Buttyàn, Jean-Pierre Hubaux.*Security And Cooperation In Wireless Networks*. http://secowinet.epfl.ch
[13] Wenjia Li, Anupam Joshi.*Security Issues in Mobile Ad Hoc Networks*.
[14] F. L. Lewis.*Wireless Sensor Networks*.To appear in Smart Environments: Technologies, Protocols, and Applications ed. D.J. Cook and S.K. Das, John Wiley, New York, 2004.
[15] Duncan J. Watts, Steven H. Strogatz. *Collective dynamics of 'small-world' networks*.
[16] Yuumi Kawachi, Kenta Murata, Shinichiro Yoshii, Yukinori Kakazu. *The structural phase transition among fixed cardinal networks*.
[17] Réka Albert, Albert-Làszlò Barabàsi. *Statistical mechanics of complex networks*.
[18] Konstantin Klemm, Vìctor M. Eguìluz. *Growing scale-free networks with small-world behavior*.
[19] Jilei Liu, Baochun Li. D*istributed Topology Control in Wireless Sensor Networks with Asymmetric Links*.
[20] Fernandez-Gago, Roman, Lopez. *A Survey on the Applicability of Trust Management Systems forWireless Sensor Networks*. Third International Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing (SecPerU 2007), 2007
[21] Mohammad Momani, Subhash Challa. *Trust Management in Wireless Sensor Networks*.
[22] Li Xiong, Ling Liu. *PeerTrust: Supporting Reputation-Based Trust for Peer-to-Peer Electronic Communities*. IEEE Transactions on Knowledge and Data Engineering, vol. 16, no. 7, pp. 843-857, Jul., 2004
[23] Yacine Atif. *Building Trust in E-Commerce*. IEEE Internet Computing, vol. 06, no. 1, pp. 18-24, Jan/Feb, 2002
[24] Sergio Marti, A T. J. Giuli, A Kevin Lai, A Mary Baker. *Mitigating routing misbehavior in mobile ad hoc networks*, Proceedings of the 6th annual international conference on Mobile computing and networking , 2000
[25] Y Rebahi, V Mujica, D Sisalem, F Fokus. *A reputation-based trust mechanism for ad hoc networks* . Computers and Communications, 2005. ISCC 2005. Proceedings. 10th IEEE Symposium on , 2005