Leader (and sub Leader) Election per uniformare e minimizzare i consumi dei nodi di una rete

Ticozzi Paolo,Matr:570067-IAM, Trinca Claudio,Matr:567105-IAM

1 Introduzione

Si verifica oggigiorno con una frequenza sempre crescente visto anche la veloce diffusione dei sensori wireless, il caso di dover operare con una rete di sensori connessi, la quale deve comunicare all'esterno delle misure o elaborazioni di queste (che sono valori estratti dai singoli nodi). Spesso si vuole per semplicita' di realizzazione che soltanto un singolo nodo possa comunicare con l'esterno, ma si accetta di buon grado che questo ruolo possa nel tempo essere attribuito a nodi diversi girando quindi per la rete (usualmente un nodo mentre ricopre questa funzione viene chiamato leader oppure radice) .
Il funzionamento si basa sul fatto che i nodi non-leader convoglieranno le proprie informazioni al nodo leader che le elaborera' e le mandera' (se necessario o richiesto) e le trasmettera' all'esterno.
In queste reti di sensori in generale il carico di lavoro a cui vengono soggetti i vari nodi rappresenta un problema di fondamentale importanza perche' sussiste una relazione diretta tra il carico di lavoro dei vari nodi e l'energia richiesta dagli stessi per compiere le funzioni a cui vengono preposti (ricezione, compattamento, invio dei dati ed elaborazione nel caso della radice).
Si ha quindi un'assegnazione di compiti differenti o quantomeno di carichi di lavoro differenti tra i vari nodi della rete; questa diversificazione dei compiti causa il fatto che ad alcuni nodi venga richiesto uno 'sforzo' e quindi un dispendio energetico maggiore di quanto richiesto ad altri.
Ci si trova di fronte a casi di reti con consumi energetici diversificati, in cui mantenere il ruolo di radice su uno stesso nodo per tutta la durata del processo comporta che ci siano nodi che si scarichino molto prima di altri appunto per via del consumo non uniforme cui si accennava poc'anzi.
Cio' potrebbe addirittura compromettere l'intero funzionamento della rete (visto lo spegnimento di alcuni nodi), cosa che nel caso si fosse fatto girare il nodo radice sarebbe potuta accadere ugualmente ma in un tempo successivo, si avrebbe dunque potuto avere la rete con tutti i suoi nodi funzionanti per un tempo maggiore.
L'intento per migliorare le prestazioni e la durata di funzionamento corretto della rete sara' quindi quello di far cambiare lo stato leader facendo mutare il nodo con questa funzione di modo che il consumo del livello di batteria dei singoli nodi sia abbastanza omogeneo.
Altro caso cui si possono applicare tecniche

- *E-mail: paolo.ticozzi@gmail.com*
- *E-mail: claudio.trinca@studenti.unipd.it*

Manuscript received March 20, 2008;

e concetti che saranno illustrati in seguito, è quello di una rete di calcolatori che devono comunicare con l'esterno, tramite un unico calcolatore (server), in questo caso bisognera' tentare di uniformare non il livello delle batterie bensi' il carico di lavoro delle cpu; si noti comunque come il problema sia alquanto simile a quello dei nodi con batteria limitata che devono fare delle misure mandarle ad un nodo che le raccolga, le elebori e le mandi all'esterno.

1.1 Funzionamento generale

Verranno ora esposte le fasi principali del funzionamento di una rete atta al compito precedente, cioe' l'elezione del leader, il recupero di dati da parte dei sensori, l'invio di queste ultime al leader, la comunicazione di questo con l'esterno e lávvio di una nuova L.E. (Leader Election)

- Si indaga la morfologia della rete (se necessaria alla comunicazione, agli algoritmi di L.E. o per verificare che la rete non si sconnetta nel caso un nodo smetta di funzionare).
- Si sincronizzano i nodi per la L.E. (tutti gli algoritmi e gli argomenti che sono stati trattati in questo lavoro utilizzano come ipotesi quella che la rete di cui si sta trattando sia sincrona, sono noti in letteratura algoritmi per la sincronizzazione di clock di sensori)
- Inizia la Leader Election (secondo uno dei vari algoritmi illlustrati successivamente).
- Individuati Leader e Sotto-leader si comunica chi sono agli altri nodi (nel caso l'algoritmo non dia informazione completa a tutti i nodi, cioe' non tutti i nodi sappiano chi siano leader e sottoleader).
- Il leader precedente (o il sottoleader precedente) comunicano all'esterno i nuovi leader oppure sono proprio questi ultimi a contattare l'esterno della rete.
- I nodi convogliano l'informazione verso di loro le informazioni raccolte e compattate.
- L'esterno comunica con il Leader (o il sotto-leader in caso questo non risponda(puo' essere rotto completamente o rotto solo il canale di comunicazione))
- Si avvia una nuova L.E.

1.2 Inquadramento delle varie probematiche concernenti

Prima di addentrarsi in metodologie e algoritmi per trovare leader e sub-leader, è opportuno dare uno sguardo piu' generale alle problematiche concernenti al problema piu' generale e complesso del funzionamento globale della rete.

1.2.1 Temporizzazione L.E.

E' un problema complesso perche' da una parte e' molto comodo avere un tempo $T_{leadership}$ prefissato costante che si intervalla tra le elezioni in quanto cio' consente il risparmio della comunicazione a tutti i nodi dell'avvio della fase di L.E. (si apre pero' il problema di fissare questo $T_{leadership}$ ponendo $T_{leadership} > T_{election}$) dall'altra spesso ci sono esigenze molto concrete dipendenti dalle applicazioni che $T_{leadership}$ non sia prefissato e possa variare, ad esempio spesso si richiede che una nuova fase di leader election possa scattare se un valore del nodo leader (livello di batteria o carico cpu) oltrepassa una determinata soglia.
In molti casi reali non e' un'ipostesi pesante quella di ipotizzare che la rete sia sincrona, sono noti e ben collaudati algoritmi di convergenza e sincronizzazione di clock.
Nel caso in analisi un'idea valida potrebbe essere quella di fare in modo che $T_{leadership}$ vari in funzione della differenza di livello di batteria tra il nodo leader e il sotto-leader, o la media dei livelli di batteria di tutti i nodi, o il livello id batteria del nodo più scarico.

1.2.2 Comunicazione, duplicazione e compattamento informazione

La comunicazione di informazioni in una rete, il relativo miglior instradamento dei messaggi e la minimizzazione della ridondanza e' un problema aperto che e' gia' stato in parte affrontato con buoni risultati in un progetto dello scorso anno, i nuovi problemi che si pongono sono di come riuscire a mettere insieme i messaggi delle varie fasi per minimizzare i

consumi.
Inoltre spesso si vuole che ci sia una certa ridondanza dell'informazione, che tutta l'informazione che arrivi al leader arrivi anche al sotto-leader, quindi qui si pone il problema di minimizzare il costo dell'invio di queste informazioni pur mantenendo un certo grado di robustezza alla rottura dei nodi durante l'invio delle informazioni (si puo' pensare che sia il leader ad inoltrare tutte le informazioni al sottoleader, ma e' un approccio robusto? non e' forse meglio diramare prima le informazioni tra i due).
Spesso si vuole inoltre avere nella rete un'archivio delle informazioni raccolte, qui si pone un altro problema che e' se inoltrare le informazioni al nuovo leader, e chiaramente se si opta per il si la distanza tra il nuovo leader e il vecchio influira' sui costi e sui livelli di batteria dei nodi della rete (un approccio migliore potrebbe essere mantenere soltanto la cronologia di quali nodi sono stati leader e sottoleader e in che fase ed inoltrare le informazioni solo nel caso si rompa o il leader o il sottoleader di una determinata fase).
Inoltre se si vuole tentare di rendere simili i livelli di batteria in modo molto preciso sarebbe necessario sapere esattamente come avviene la comunicazione e tenerne conto nella scelta di leader e sottoleader.

1.2.3 Controllo stato nodi

Bisogna prevedere un controllo dello stato vitale dei nodi in ogni momento, sia durante la fase di L.E. che in quelle di comunicazione e raccolta dati. Questa andra' parzialmente ad influire sui messaggi trasmessi nella rete, e potrebbe influire sulla scelta di elezione. In questo luogo non se ne terra' pero' conto, si fara' soltanto notare quando negli algoritmi di L.E. si potra' avere un controllo gratuito.
Altra cosa importante sarebbe che quando la rottura di un nodo e' tale da rendere sconnessa la rete la rete se ne renda conto e provveda a far scattare una nuova L.E. nella parte di rete orfana di leader se in questa manca il sottoleader o in questo caso usi quest'ultimo come leader di quella porzione di rete.

1.2.4 Criteri di elezione

Mentre nel caso si voglia eleggere soltanto un leader la scelta del criterio di elezione e'abbastanza chiara (essendo che si vogliono minimizzare esclusivamente i consumi) quando si vogliono eleggere sia un leader che un sotto-leader la scelta di criteri congiunti per l'elezione da' luogo a problematiche maggiori che comportano varie possibili scelte:

- ricerca leader e sottoleader entrambi separatamente ottimi (e' quella che verra'trattata principalmente in questo lavoro)
- ricerca congiunta di leader e sottoleader (si puo' pero' riportare questo al caso precedente utilizzando opportuni funzionali dipendenti dai valori di tutti i nodi che tengano conto della scelta congiunta)
- ricerca leader ottimo e poi scelta del sottoleader in modo euristico (ad esempio per risparmiare sulle comunicazioni si puo' scegliere per questo ruolo uno dei vicino del nodo leader)

2 DEFINIZIONE DEL PROBLEMA E RAPPRESENTAZIONE FORMALE DELLA RETE.

Nei prossimi capitoli ci proporremo l'obbiettivo di trovare la tecnica migliore per uniformare il carico di lavoro tra i vari nodi. Nel caso in cui non esista una tecnica ottima cercheremo allora di studiare una serie di tecniche sostitutive valutandone i pregi ed i difetti. Notiamo che se non esiste una tecnica ottima secondo un certo criterio per uniformare il carico di lavoro della rete wireless é intuibile che l'utilizzo di altre tecniche subottime puó dare risultati diversi al variare di quale tecnica venga impiegata ed al variare della struttura della rete nel caso specifico.

2.1 Morfologia della rete.

Per affrontare il percorso di studio che verrá fatto in tale sede riguardo agli argomenti precedentemente menzionati non verranno avanzate richieste troppo pretenziose nei confronti della struttura della rete o dei suoi nodi. L'obbiettivo primario rimane quello di uniformare il carico di lavoro tra i nodi della rete. A tale scopo chiediamo che i vari nodi, quindi i vari sensori costituenti la rete stessa, abbiano le **stesse potenzialitá di calcolo**. Questa caratteristica viene richiesta per garantire che tutti i nodi siano in grado di adempiere a tutte le richieste possibili. In sostanza si vuole poter decidere liberamente a quale nodo far compiere una certa operazione indipendentemente dalla sua complessitá od onerositá dal punto di vista energetico.
Viene anche richiesto che i vari sensori abbiano lo **stesso livello energetico iniziale**. Questa ipotesi risulta comoda perché tramite essa siamo autorizzati a stabilire in modo unico per tutti i componenti della rete una relazione di equivalenza tra livello energetico di un sensore ed il lavoro svolto dallo stesso fino ad un certo istante. Tale scelta inoltre non risulta essere una pura formalita teorica perché intuitivamente risulta piú comodo l'acquisto in gruppo dei componenti che dovranno costituire la rete. Notiamo che se volessimo evitare di effettuare una legge di controllo sulla rete evidentemente bisognerebbe dimensionare le batterie dei singoli sensori in base alla quantitá di lavoro che si prevede per il singolo sensore. Tale scelta non puó essere soddisfacente in generale in quanto é ragionevole pensare che in commercio non esistano batterie che soddisfino a una qualsiasi configurazione ipotetica della rete. Evidentemente si potrebbe pensare di attuare una legge di controllo su una rete di sensori con batterie di diverse capacitá. Per il momento tale scelta non viene considerata per semplicitá.
Per quanto riguarda la metodologia di comunicazione tra i vari nodi supporremo che la **comunicazione** sia **bidirezionale**. Ció significa che se il nodo A puó mandare messaggi al nodo B allora il nodo B puó mandare messaggi al nodo A. Risulta inoltre fondamentale ipotizzare una comunicazione tra i vari nodi caratterizzata dalla **assenza di perdite di messaggio**. La prima delle ultime due ipotesi introdotte non é inverosimile, infatti a meno di malfunzionamenti poco probabili essa risulta verificata. Inoltre se tutta la rete é costituita da una unica tipologia di sensori se un sensore riesce a comunicare con un altro sensore per simmetria risulta logico pensare che quest'ultimo riesca a comunicare con il primo. L'ultima ipotesi riguardante invece le perdite di messaggio potrebbe non essere verificata. Tale problema non verrá approfondito in tale sede e dovrá essere gestito da una opportuna architettura di comunicazione.
Un altro fenomeno di fondamentale importanza che caratterizza fortemente il funzionamento della rete é costituito dai ritardi di arrivo dei messaggi. Sostanzialmente ipotizzeremo l'**assenza di ritardi** di arrivo dei messaggi. Per chiarire questo punto consideriamo un esempio:
detto γ il tempo tra due istanti di aggiorna-

mento della rete se in un qualche istante temporale continuo (τ) compreso tra due istanti di aggiornamento della rete discreti e contigui (K_1 e $K_2 = K_1 + \gamma$) il nodo A invia un messaggio al nodo B allora si vuole che il messaggio arrivi al nodo B entro un tempo $\alpha < \gamma$. Si vede con facilitá che il problema dei ritardi di arrivo dei messaggi puó essere affrontato con una scelta opportuna di γ.
Un'ipotesi fondamentale sulla struttura della rete consiste nel considerare durante la trattazione solamente **reti connesse**. Vogliamo cioé avere la possibilitá di essere in grado di raggiungere un qualsiasi nodo a partire dall'invio di un messaggio da un qualsiasi altro nodo della rete grazie al passaggio attraverso dei nodi intermedi che provvedono al reinvio del messaggio stesso.
L'ultima ipotesi che prenderemo riguarda il flusso di messaggi attraverso la rete. Si vuole che tutti i nodi della rete inviino il proprio messaggio in direzione dello stesso nodo detto **radice**. Si vuole cioé che se i nodi della rete sono A, B, C e D dove D é radice ed il percorso piú breve per arrivare a D partendo dal generico sensore A é $A \to B \to C \to D$ allora A invia il messaggio a B che poi dovrá reindizizzarlo verso D. La radice sostanzialmente é il nodo al quale deve giungere l'informazione che dovrá essere elaborata.

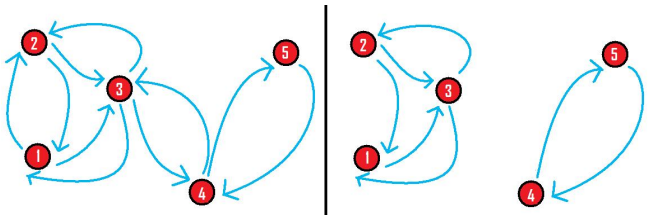

Figura 1. La rete a sinistra soddisfa le ipotesi mentre quella a destra non le soddisfa.

Riportiamo di seguito le varie ipotesi sulla morfologia della rete:

1) Uguali potenzialitá di calcolo per tutti i sensori.
2) Uguali livelli energetici iniziali per tutti i sensori.
3) Comunicazione bidirezionale tra i vari sensori.
4) Assenza di perdite di messaggio durante la comunicazione.
5) Scelta opportuna dei parametri per avere l'assenza di ritardi di comunicazione.
6) Presenza di struttura connessa della rete in esame.
7) Invio dei messaggi verso la radice.

2.2 Idea base sulla metodologia di controllo.

Riteniamo indispensabile al fine di una comprensione approfondita del problema chiarire alcuni importanti concetti che introdurremo nel seguito. Abbiamo caratterizzato il flusso dei messaggi nella rete come un flusso direzionale. Tutti i sensori inviano i loro messaggi in direzione del nodo detto radice. Risulta abbastanza semplice capire che il flusso di messaggi attraverso i singoli sensori sará piú intenso in corrispondenza ai sensori piú vicini alla radice. Tale flusso di messaggi sará invece meno intenso in corrispondenza ai sensori piú distanti da quest'ultima.
Nel caso in cui venga ad instaurarsi una proporzionalitá diretta tra il numero di messaggi ricevuti ed il dispendio energetico, ovvero se ogni arrivo di un messagio comporta una spesa energetica di una unitá fissa, allora si puó dire con certezza che il livello energetico dei sensori piú vicini alla radice (radice inclusa) cala piú velocemente rispetto ai livelli energetici dei sensori che risiedono nelle zone piú periferiche della rete. Questo fatto é sentinella squillante di una possibile tecnica di risoluzione del problema stesso. Tale tecnica si basa sulla mobilitá della radice attraverso la rete. Ció significa che per risolvere il problema della equidistribuzione del carico di lavoro dei vari sensori vogliamo trovare un criterio opportuno di assegnazione del ruolo di radice. Tale ruolo verrá aggiornato nel corso della evoluzione della rete secondo una legge specifica che ricaveremo nel seguito. Il motivo basilare che induce a tentare tale metodologia di risoluzione del problema che ci siamo posti risiede nel fatto che se nel corso della evoluzione della rete assegnamo il ruolo di radice ad un no-

do che risiede in una delle zone piú cariche della rete stessa allora i sensori in tale zona molto probabilmente inizieranno a scaricarsi piú velocemente rispetto ai sensori nelle zone periferiche. Inoltre se i sensori appena citati che risiedono nelle zone piú periferiche della rete prima dell'aggiornamento della radice costituivano la porzione della rete piú scarica allora possiamo pensare, senza purtroppo averne la certezza assoluta, che nella evoluzione prossima della rete i vari livelli energetici inizino a tendere ad un livello energetico comune. Infatti é ragionevole che i nodi che erano piú scarichi prima dell'aggiornamento della radice ora si scarichino piú lentamente dei nodi che erano meno scarichi prima dell'aggiornamento della radice. Il livello di quest'ultimi allora inizierá a calare fino al raggiungimento del livello dei nodi che erano piú scarichi. Raggiunta tale situazione si dovrá valutare tra quanto fare un nuovo aggiornamento della radice.
Notiamo che sono essenziali le ipotesi di rete connessa ed uguale capacitá di calcolo dei vari sensori. Queste due ipotesi precedentemente introdotte ci garantiscono di poter prendere come radice della rete uno qualsiasi dei sensori in uno qualsiasi degli istanti di evoluzione. Infatti senza l'ipotesi di rete connessa alcuni sensori potrebbero essere esclusi dalla lista dei candidati ad essere radice perché non raggiungibili da altri sensori, mentre senza l'altra ipotesi alcuni sensori potrebbero non essere dotati delle capacitá di calcolo indispensabili per ricoprire il ruolo di radice.

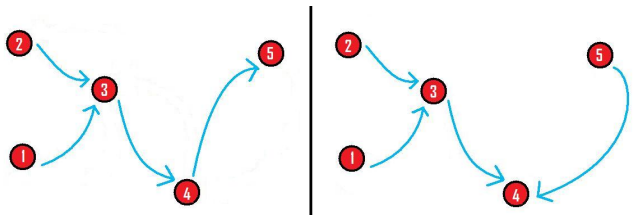

Figura 2. A sinistra flusso di messaggi con sensore 5 radice, a destra flusso di messaggi con sensore 4 radice. É stata usata la rete di FIGURA 1 parte sinistra.

2.3 Impiego di grafi ed utilizzo della matrice M_g.

Ci accingiamo ora ad introdurre una rappresentazione formale della rete di sensori wireless. Ricordiamo che l'obbiettivo principale che ci siamo posti é trovare una legge di controllo che indichi dove posizionare la radice all'interno della rete di sensori nel corso dell'evoluzione della stessa. Ricordiamo inoltre che la ricerca di tale legge di controllo nasce dalla volontá di uniformare il dispendio energetico tra i vari sensori della rete.
Evidentemente per trovare una legge di controllo che soddisfi ad un certo criterio bisogna utilizzare una buona rappresentazione formale del problema in questione. Nella trattazione precedente sono stati volutamente trattati come sinonimi i termini "sensore" e "nodo". Questo fatto dovrebbe far intuire come la scelta qui presa sia quella di schematizzare la rete di sensori wireless attraverso l'utilizzo di una struttura a grafo.

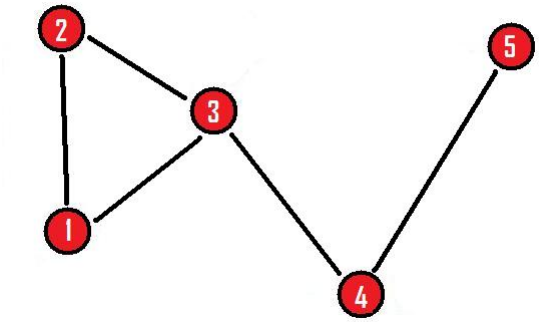

Figura 3. Rappresentazione del grafo della rete di FIGURA 1 parte sinistra. Gli arci sono bidirezionali.

La rappresentazione tramite un grafo se presa da sola purtroppo non riesce a dare una struttura formale abbastanza potente da permettere di lavorare agevolmente alla ricerca della soluzione al problema che ci siamo posti. A tale scopo risulta inevitabile l'introduzione della matrice rappresentativa del grafo in questione. Tale matrice che rappresenta in modo equivalente al grafo la rete di sensori wireless verrá in tutto il seguito chiamata M_g.

2.3.1 *La matrice M_g.*

Gli elementi costituenti la matrice $M_g = [m_{i,j}]$ rappresentante della rete in esame vengono definiti nel seguente modo:

$$(m_{i,j} = 1 \vee m_{i,j} = 0)$$
$$\wedge$$
$$(m_{i,j} = 1 \iff \text{i puó mandare messaggi a j})$$

Notiamo che tale matrice gode di alcune proprietá:

1) É costituita da elementi che possono essere solo zero o uno.
2) É simmetrica
3) Sulla diagonale principale puó avere solo elementi nulli (esclusa la possibilitá di autoinviarsi messaggi).
4) É in generale NON definita.

Le prime tre proprietá sono banali, la quarta seppur piuttosto semplice viene chiarita meglio dal seguente esempio.

ESEMPIO: Consideriamo sempre la rete di FIGURA 1 parte sinistra. La matrice M_g relativa a tale rete si trova facilmente.

$$M_g = \begin{bmatrix} 0&1&1&0&0 \\ 1&0&1&0&0 \\ 1&1&0&1&0 \\ 0&0&1&0&1 \\ 0&0&0&1&0 \end{bmatrix}$$

Per convincersi che tale matrice é non definita basta trovare i seguenti due risultati:

$$x^T M_g x = -34 \text{ se } x^T = \begin{bmatrix} -10&1&1&1&1 \end{bmatrix}$$
$$x^T M_g x = 10 \text{ se } x^T = \begin{bmatrix} 1&1&1&1&1 \end{bmatrix}$$

Siccome i due risultati sono discordi in segno M_g é in generale non definita.

2.4 Impiego di alberi ed utilizzo della matrice M_a.

Durante l'evoluzione della rete il flusso di messaggi viene continuamente indirizzato verso il nodo del grafo definito radice. Tale nodo radice non é vincolato ad una posizione invariante per tutta la durata della evoluzione della rete ma puó essere spostato senza alcuna tipologia di vincoli. Si rende quindi necessaria l'introduzione di una struttura capace di racchiudere in se la configurazione della rete e l'evoluzione del flusso di messaggi attraverso la stessa.
Lo strumento utilizzato per risolvere il problema di tale formalismo deriva dalla Teoria dei Grafi ed é rappresentato dagli Alberi. La struttura ad albero si presta perfettamente allo scopo prefissato. Infatti la radice dell'albero rappresenta perfettamente il nodo del grafo che é stato definito nodo radice, i nodi dell'albero che sono figli della radice rappresentano i nodi del grafo che inviano messaggi alla radice, i nodi dell'albero che sono figli dei figli della radice rappresentano i sensori che non possono comunicare direttamente con la radice e sono quindi costretti ad inviare i loro messaggi ai figli della radice che reinvieranno i messaggi ricevuti, si procede fino ad arrivare alle foglie dell'albero le quali sono i nodi del grafo che inviano solamente messaggi senza riceverne.

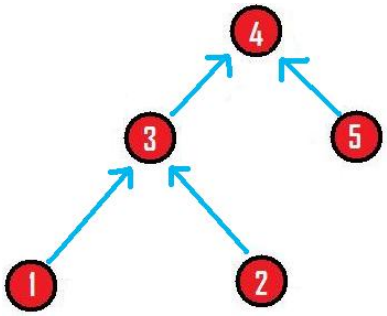

Figura 4. Rappresentazione dell'albero estratto dal grafo di FIGURA 3 con nodo radice 4. Gli arci sono monodirezionali.

Dalla Teoria dei Grafi si sa che dato un grafo e preso un suo nodo la scelta dell'albero avente il nodo considerato in posizione di radice non é unica. Ci si chiede allora secondo quale criterio bisogna definire la struttura ad albero a partire dal grafo rappresentativo della rete ed a partire dalla conoscenza di quale sará il nodo radice dell'albero. La scelta migliore risulta senza dubbio quella di prendere tra tutte le possibili configurazioni ad albero quella con il minor numero di livelli. Infatti prendendo l'albero con

il minor numero di livelli e detto T il tempo entro il quale tutti i messaggi arrivano alla radice si garantisce che se ad un certo istante tutti i nodi della rete inviano un messaggio verso la radice allora T é il minore possibile. Inoltre un altro motivo che fortifica questa scelta risiede nel fatto che avendo il minor numero di livelli possibili ogni messaggio prima di raggiungere la radice verrá reinviato il minor numero di volte. Visto che tale tecnica cerca di ridurre il numero di messaggi in transito nella rete e visto che il numero di messaggi che vengono inviati é in un certo senso proporzionale alla energia spesa dalla rete allora possiamo dire che tra tutte le configurazioni ad albero possibili quella a minimo numero di livelli é quella che minimizza il dispendio energetico.

2.4.1 Matrice M_a e sua estrazione a partire da M_g.

Come fatto nella sezione precedente per M_g ci accingiamo ora a definire una matrice rappresentativa dell'albero che definiamo come M_a. In tale sede peró introdurremo anche un primo metodo per riuscire a ricavare M_a a partire da M_g e dalla conoscenza di quale vogliamo che sia il nodo radice. La matrice $M_a = [m_{i,j}]$ in questione é definita nel modo seguente:

$$m_{i,j} = 1 \vee m_{i,j} = 0$$
$$\wedge$$
$m_{i,j} = 1 \iff$ i riceve messaggi da j nella particolare configurazione ad albero

Tale definizione di M_a richiede la conoscenza dell'albero a minimo numero di livelli che essa deve rappresentare. In veritá vedremo nel seguito un algoritmo che ricava direttamente M_a sfruttando la matrice M_g. Prima di iniziare tale tecnica che verrá realizzata da un algoritmo per facilitare la comprensione vediamo un esempio.

ESEMPIO: Considerare la matrice M_g dell'esempio in sezione 2.1.1 . Si vede con facilitá che la matrice M_a ottenuta dalla M_g con nodo radice 4 risulta essere:

$$M_a = \begin{bmatrix} 0&0&0&0&0 \\ 0&0&0&0&0 \\ 1&1&0&0&0 \\ 0&0&1&0&1 \\ 0&0&0&0&0 \end{bmatrix}$$

Riportiamo di seguito un primo algoritmo che ricava direttamente M_a sfruttando la matrice M_g.

1) Prendo una matrice "vuota" A delle stesse dimensioni di M_g e una matrice
$$B = M_g$$
2) Metto a zero la riga di B relativa alla radice.
$$B = \begin{bmatrix} 0&1&1&0&0 \\ 1&0&1&0&0 \\ 1&1&0&1&0 \\ 0&0&0&0&0 \\ 0&0&0&1&0 \end{bmatrix}$$
3) Come colonna di A in posizione j relativa al nodo radice j metto la colonna j-esima di B relativa alla radice
$$A = \begin{bmatrix} ?&?&?&0&? \\ ?&?&?&0&? \\ ?&?&?&1&? \\ ?&?&?&0&? \\ ?&?&?&1&? \end{bmatrix}$$
4) Se in una delle righe di A c'é almeno un uno allora metto a zero la rispettiva riga di B. Faccio cosí per tutte le righe di A.
$$B = \begin{bmatrix} 0&1&1&0&0 \\ 1&0&1&0&0 \\ 0&0&0&0&0 \\ 0&0&0&0&0 \\ 0&0&0&0&0 \end{bmatrix}$$
5) Prendo una matrice C delle dimensioni di A con colonna in posizione della radice la colonna di M_g relativa alla radice. Se nella riga j-esima di A c'é almeno un uno allora prendo come colonna j-esima di C la colonna j-esima di B. Faccio cosí per tutte le righe di A.
$$C = \begin{bmatrix} ?&?&1&0&0 \\ ?&?&1&0&0 \\ ?&?&0&1&0 \\ ?&?&0&0&0 \\ ?&?&0&1&0 \end{bmatrix}$$
6) Impongo $A = C$.
7) Se B é diversa dalla matrice nulla allora torno al punto 4). Finite le iterazioni la A dell'esempio diventa:
$$A = \begin{bmatrix} 0&0&1&0&0 \\ 0&0&1&0&0 \\ 0&0&0&1&0 \\ 0&0&0&0&0 \\ 0&0&0&1&0 \end{bmatrix}$$
8) In generale la matrice A potrebbe avere almeno in una riga piú di un uno. Fino a questo punto dell'algoritmo si sono

sostanzialmente fissati i vari livelli dell'albero. Ció che succede é che in un livello dell'albero potrebbe esserci un nodo che almeno potenzialmente potrebbe avere piú genitori, bisogna decidere quale sará il suo genitore effettivo. Un possibile criterio é quello di equidistribuire i figli tra i genitori del livello precedente al fine di uniformare il carico di lavoro.

9) Alla fine si pone

$$M_a = A^T$$

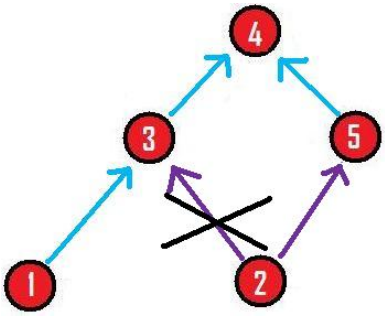

Figura 5. Nell'ipotetica rete considerata in figura il nodo 2 potenzialmente puó avere piú genitori. Viene scelto 5 come genitore per uniformare il carico di lavoro tra 5 e 3.

Tale algoritmo seppur esaminato nei minimi dettagli non é stato dimostrato in modo matematicamente rigoroso, si lascia al lettore il tempo di meditare su tale procedura di cui se ne crede fermamente la correttezza. La tecnica di uniformare i figli non é l'unica possibile in seguito tratteremo altre tecniche ed altri algoritmi. É di fondamentale importanza notare che il numero di iterazioni di tale algoritmo é non superiore al numero di livelli dell'albero che si vuole rappresentare. Si puó quindi affermare con certezza che **l'algoritmo termina in un numero finito di passi**.

2.5 La funzione $\Phi(\bullet)$.

Apriamo ora un paragrafo il cui argomento potrebbe sembrare estraneo a quanto trattato fino ad ora. Premettiamo che quanto verrá spiegato in tale sede sará indispensabile alla rappresentazione sistemistica della rete che faremo in seguito.

Introdurremo ora una particolare funzione detta funzione $\Phi(\bullet)$.

$$\text{DEF:} \quad \Phi(\bullet): \quad R_+^{N\times M} \longrightarrow \{0,1\}^{N\times N}$$

$$M \in R_+^{N\times M} \longrightarrow \begin{bmatrix} \Phi_{1,1} & & \oslash \\ & \ddots & \\ \oslash & & \Phi_{N,N} \end{bmatrix}$$

Dove gli elementi $\Phi_{j,j}$ sono definiti dalla:

$$\begin{cases} \Phi_{j,j} = 0 \text{ se } M_{j,i} = 0 \forall i \\ \Phi_{j,j} = 1 \text{ altrimenti} \end{cases}$$

Per evitare incomprensioni viene di seguito specificato il significato dei vari simboli utilizzati nella definifione di tale funzione:

- $R_+^{N\times M}$ é lo spazio delle matrici con N righe ed M colonne ad elementi maggiori od uguali a zero.
- $\{0,1\}^{N\times N}$ é lo spazio delle matrici quadrate di ordine N con elementi che possono assumere solamente valore 0 oppure 1.

La funzione $\Phi(\bullet)$ in sostanza non fa altro che prendere una matriche M ad elementi non negativi che potrebbe essere rettangolare o quadrata oppure potrebbe essere un semplice vettore e restituire una matrice quadrata $\Phi(M)$ dell'ordine del numero di righe di M. Tale matrice $\Phi(M)$ presenta elementi tutti nulli ad eccezione di alcuni elementi della diagonale principale. Il criterio per decidere se mettere zero od uno sulla diagonale principale di $\Phi(M)$ é il seguente: se la riga i-esima di M é costituita da elementi tutti nulli allora in posizione (i,i) di $\Phi(M)$ metto zero, altrimenti metto uno.

ESEMPI:

$$\Phi\left(\begin{bmatrix} 1 \\ 8.5 \\ 0 \end{bmatrix}\right) = \begin{bmatrix} 1&0&0 \\ 0&1&0 \\ 0&0&0 \end{bmatrix}; \quad \Phi\left(\begin{bmatrix} 1&7.9 \\ 0&0 \\ 0&8.1 \end{bmatrix}\right) = \begin{bmatrix} 1&0&0 \\ 0&0&0 \\ 0&0&1 \end{bmatrix}$$

$$\Phi\left(\begin{bmatrix} 0&1 \\ 1&0 \end{bmatrix}\right) = \begin{bmatrix} 1&0 \\ 0&1 \end{bmatrix}$$

Premettiamo che questa funzione comparirá nella rappresentazione sistemistica della rete che verrá fatta nel seguito. La conoscenza di alcune proprietá del sistema che rappresenterá

la rete sará quindi strettamente correlata alla conoscenza della proprietá di questa funzione. Diventa allora indispensabile procedere ad un studio approfondito della funzione $\Phi(\bullet)$. Tale approfondimento sulla funzione in esame ha portato ad individuare una serie di proprietá che vengono di seguito elencate.

PROPRIETÁ DI $\Phi(\bullet)$:

1) $\Phi(M+M) = \Phi(M)$
   dove M é una matrice di dimendioni arbitrarie

   $\Longrightarrow \Phi(\bullet)$ é NON LINEARE.

2) $\Phi(\Phi(M)) = \Phi(M)$.

3) $\Phi\left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right) = \Phi\left(\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\right)$

   $\Longrightarrow \Phi(\bullet)$ NON INIETTIVA

   $\Longrightarrow \Phi(\bullet)$ NON INVERTIBILE.

4) $\Phi((I-\Phi(M))*A) = (I-\Phi(M))*\Phi(A)$.
   dove I é la matrice identitá e M ed A sono matrici di dimensioni arbitrarie

5) $\Phi(A)*\Phi(A) = \Phi(A)$.

6) $\Phi(A)*\Phi(B) = \Phi(B)*\Phi(A)$
   dove B ha dimensioni arbitrarie.

7) $\Phi(A+B) = \Phi(\Phi(A)+\Phi(B))$
   dove B ha le stesse dimensioni di A.

8) $\Phi(A)*\Phi(M) = \Phi(\Phi(M)*A)$.

La dimostrazione di tali proprietá é banale se si tiene presente che presa la generica matrice M appartenente al dominio di $\Phi(\bullet)$ si ha che $\Phi(M)$ é una matrice diagonale, inoltre si ha che le operazioni $M*\Phi(\bullet)$ e $(I-\Phi(\bullet))*M$ rappresentano rispettivamente l'operazione di mettere a zero tutti gli elementi di alcune colonne di M e l'operazione di mettere a zero tutti gli elementi di alcune righe di M.

2.6 Relazione formale tra M_a ed M_g ottenuta tramite la funzione $\Phi(\bullet)$.

Anche se già accennato in precedenza ribadiamo che tra non molto introdurremo una rappresentazione sistemistica della rete di sensori wireless. Risultera quindi molto più comodo ai fini della trattazione teorica anzichè avere una rappresentazione algoritmica del legame tra M_a ed M_g avere una vera e propria formula matematica per rappresentare tale relazione. Premettiamo che la ricerca di un tale legame sarà caratterizzata da un forte utilizzo della funzione $\Phi(\bullet)$). La formula che riporteremo di seguito non sarà altro che una rappresentazione matematica dell'algoritmo già introdotto che lega M_a ad M_g. Prima di procedere a tale scopo definiamo il seguente vettore:

$$X_r = \begin{bmatrix} X_r^1 \\ \vdots \\ X_r^N \end{bmatrix}$$

dove $X_r^i = 1$ se il nodo i è radice altrimenti vale 0.
Grazie all'utilizzo di tale vettore possiamo introdurre la formula che lega M_a ad M_g:

$$\begin{aligned}
(M_a)^T = {} & \overbrace{(I-\Phi(X_r))*M_g*\Phi(X_r)}^{P_0} + \\
& + \overbrace{(I-\Phi(\Phi(X_r)+P_0))*M_g*\Phi(P_0)}^{P_1} + \\
& + \overbrace{(I-\Phi(\Phi(X_r)+P_0+P_1))*M_g*\Phi(P_1)}^{P_2} + \\
& + \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots + \\
& + \overbrace{(I-\Phi(\Phi(X_r)+\sum_{i=0}^{N-1} P_i))*M_g*\Phi(P_{N-1})}^{P_N}
\end{aligned}$$

Per facilitare la comprensione di questa formula ricordiamo che l'operazione $M_g*\Phi(X_r)$ ha il significato di mettere a zero gli elementi delle colonne di M_g in corrispondenza agli elementi nulli di X_r, invece l'operazione $(I-\Phi(X_r))*M_g$ corrisponde a mettere a zero gli elementi delle righe di M_g in corrispondenza agli elementi di valor 1 del vettore X_r. Risulta immediato notare che le

varie matrici P_i sono tutte matrici quadrate delle stesse dimensioni della matrice M_a e quindi anche delle stesse dimensioni della matrice M_g. Inoltre tali matrici P_i sono costituite da elementi che possono assumere solamente valore 1 oppure valore zero. Questo fatto deriva direttamente dalle operazioni che portano ad ottenere le matrici in questione a partire dalla M_g. Per ottenere una particolare P_i infatti non si fa altro che annullare alcune righe di M_g e selezionare dalla matrice così ottenuta alcune delle sue colonne. Premettiamo che la somma della generica P_i corrisponde alla creazione del livello i-esimo dell'albero che sarà rappresentato dalla matrice M_a. Notiamo inoltre che le varie P_i sono in un certo senso disgiunte, ovvero se in P_i la colonna k-esima presenta almeno un elemento pari ad 1 allora tutte le altre P_j con j diverso da i presentano colonna k-esima con tutti elementi nulli. Il numero di P_i, le quali rappresentano i vari livelli dell'albero in questione, sarà esattamente uguale al numero di livelli di tale albero. Possiamo quindi affermare che tale somma di matrici è una **somma di un numero finito di termini**. Tale procedimento risulterà molto più chiaro una volta visto il seguente esempio.

ESEMPIO: Consideriamo la matrice M_g ed il nodo radice dell'esempio fatto in sezione 2.2.1 . Notiamo che risulta:

$$M_g = \begin{bmatrix} 0&1&1&0&0\\ 1&0&1&0&0\\ 1&1&0&1&0\\ 0&0&1&0&1\\ 0&0&0&1&0 \end{bmatrix} \quad X_r = \begin{bmatrix} 0\\0\\0\\1\\0 \end{bmatrix}$$

Procediamo ora al calcolo delle varie P_i:

$$P_0 = \left(I - \begin{bmatrix} 0&0&0&0&0\\ 0&0&0&0&0\\ 0&0&0&0&0\\ 0&0&0&1&0\\ 0&0&0&0&0 \end{bmatrix} \right) M_g \begin{bmatrix} 0&0&0&0&0\\ 0&0&0&0&0\\ 0&0&0&0&0\\ 0&0&0&1&0\\ 0&0&0&0&0 \end{bmatrix}$$

$$= \begin{bmatrix} 1&0&0&0&0\\ 0&1&0&0&0\\ 0&0&1&0&0\\ 0&0&0&0&0\\ 0&0&0&0&1 \end{bmatrix} M_g \begin{bmatrix} 0&0&0&0&0\\ 0&0&0&0&0\\ 0&0&0&0&0\\ 0&0&0&1&0\\ 0&0&0&0&0 \end{bmatrix} = \begin{bmatrix} 0&0&0&0&0\\ 0&0&0&0&0\\ 0&0&0&1&0\\ 0&0&0&0&0\\ 0&0&0&1&0 \end{bmatrix}$$

$$P_1 = \left(I - \begin{bmatrix} 0&0&0&0&0\\ 0&0&0&0&0\\ 0&0&1&0&0\\ 0&0&0&1&0\\ 0&0&0&0&1 \end{bmatrix} \right) M_g \begin{bmatrix} 0&0&0&0&0\\ 0&0&0&0&0\\ 0&0&1&0&0\\ 0&0&0&0&0\\ 0&0&0&0&1 \end{bmatrix}$$

$$= \begin{bmatrix} 1&0&0&0&0\\ 0&1&0&0&0\\ 0&0&0&0&0\\ 0&0&0&0&0\\ 0&0&0&0&0 \end{bmatrix} M_g \begin{bmatrix} 0&0&0&0&0\\ 0&0&0&0&0\\ 0&0&1&0&0\\ 0&0&0&0&0\\ 0&0&0&0&1 \end{bmatrix} = \begin{bmatrix} 0&0&1&0&0\\ 0&0&1&0&0\\ 0&0&0&0&0\\ 0&0&0&0&0\\ 0&0&0&0&0 \end{bmatrix}$$

$$P_2 = \left(I - \begin{bmatrix} 1&0&0&0&0\\ 0&1&0&0&0\\ 0&0&1&0&0\\ 0&0&0&1&0\\ 0&0&0&0&1 \end{bmatrix} \right) M_g \begin{bmatrix} 1&0&0&0&0\\ 0&1&0&0&0\\ 0&0&0&0&0\\ 0&0&0&0&0\\ 0&0&0&0&0 \end{bmatrix} = \begin{bmatrix} 0&0&0&0&0\\ 0&0&0&0&0\\ 0&0&0&0&0\\ 0&0&0&0&0\\ 0&0&0&0&0 \end{bmatrix}$$

Visto che $P_2 = \oslash$ ciò implica direttamente che P_j per qualsiasi $j > 2$ sarà anche essa una matrice nulla. Notiamo che per ottenere la M_a dobbiamo sommare un numero di matrici pari al numero di livelli dell'albero che vogliamo rappresentare con tale matrice. Sommando le matrici appena calcolate otteniamo:

$$(M_a)^T = \begin{bmatrix} 0&0&1&0&0\\ 0&0&1&0&0\\ 0&0&0&1&0\\ 0&0&0&0&0\\ 0&0&0&1&0 \end{bmatrix}.$$

Tale risultato è lo stesso che abbiamo ottenuto nell'esempio di sezione 2.2.1. Notiamo che con tale procedura rimane il problema che alcuni nodi di un livello potrebbero avere più candidati ad essere genitori. Bisognerà quindi usare un criterio di scelta dei genitori. Il grande vantaggio che ci da l'utilizzo della funzione $\Phi(\bullet)$ sta nella immediata conoscenza della non linearità della relazione tra M_a, M_g e X_r. Inoltre la formula trovata si presta meglio ad una trattazione formale rispetto all'algoritmo introdotto in precedenza.

2.7 Rappresentazione sistemistica della rete.

Lo scopo che ci prefiggiamo ora è quello di dare una rappresentazione sistemistica della rete di sensori wireless. Una tale tipologia di rappresentazione è sembrata necessaria per cercare di intraprendere una qualche stada per arrivare ad una legge di controllo della rete. Tale legge di controllo sarà cercata in modo da riuscire ad adempiere allo scopo iniziale di equidistribuire il carico di lavoro tra i vari nodi dell'albero.
Visto che stiamo cercando di delineare la struttura di una rappresentazione sistemistica come prima cosa bisogna definire lo stato della rete, quindi del sistema.

2.7.1 Definizione dello stato della rete.

Si definisce stato del sistema rete il vettore seguente:

$$X(k) = \begin{bmatrix} X_1(k) \\ \vdots \\ X_N(k) \end{bmatrix}$$

dove $X_i(k)$ rappresenta il numero di messaggi ARRIVATI al nodo i-esimo INVIATI dai figli del nodo i-esimo stesso.
Notiamo che la dimensione dello stato del sistema rete è pari alla dimensione effettiva della rete di sensori, quindi è esattamente pari al numero di sensori componenti la rete. Ovviamente le componenti del vettore di stato possono assumere solamente valori naturali e non negativi. Notiamo inoltre che se il nodo i-esimo rappresenta una delle foglie dell'albero nella particolare configurazione della rete allora la componente i-esima del vettore di stato sarà nulla perchè le foglie della rete non hanno figli e quindi non ricevono messaggi da altri nodi. Dato che per ipotesi abbiamo una comunicazione sicura possiamo affermare che le uniche componenti nulle del vettore di stato sono quelle relative alle foglie della rete.

2.7.2 Tipologie di messaggi.

In tale sede divideremo i messaggi ricevuti dai vari nodi della rete in due categorie. Distinguiamo i vari messaggi tra i messaggi arrivati ad un certo nodo dopo essere stati inviati da un altro nodo e tra i messaggi di misura. Ricordiamo che lo scopo di una rete di sensori è quello di fare delle misure di alcune particolari grandezze quindi possiamo modellare l'evento di misura eseguita come un vero e proprio arrivo di un messaggio contenente l'informaziuone della misura stessa. Tale informazione servirà al nodo radice per eseguire su essa dei calcoli. A tale scopo il nodo che ha eseguito la misura invierà un messaggio contenente l'informazione della misura verso la radice. L'ultimo messaggio menzionato appartiene alla prima categoria di messaggi introdotta.

2.7.3 Procedura di invio di messaggi.

Definiamo ora con precisione come vengono inviati i messaggi. Ricordiamo che k è un istante temporale discreto e come tale assume

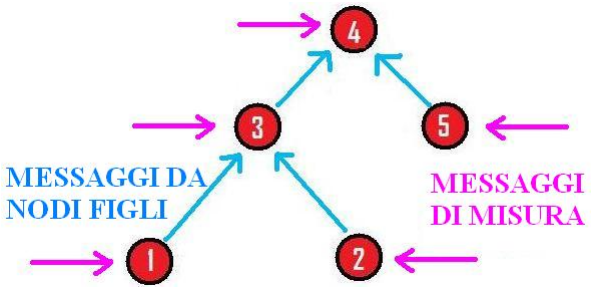

Figura 6. Albero con distinzione tra le diverse tipologie di messaggi.

i valori naturali a partire da 0. Definiamo T come una quantità temporale continua fissata e caratteristica della rete. Supponiamo di trovarci nell'istante continuo $T * k$. In tale istante si suppone che tutti i nodi della rete ricevano il proprio messaggio di misura e ricevano i vari messaggi che gli sono stati inviati da altri nodi della rete. Una volta ricevuti i vari messaggi tutti i nodi portano a termine le operazioni alle quali sono preposti entro il tempo T e inviano i loro messaggi che arriveranno al destinatario al tempo $(T + 1) * k$.
Nel seguito della trattazione sono state considerate due diverse situazioni di reinvio dei messaggi. Un primo metodo di reinvio dei messaggi consiste nel reinviare tutti i messaggi ricevuti al prorio nodo genitore. Quindi se all'istante k un nodo riceve n messaggi dai suoi nodi figli ed un messaggio di misura allora tale nodo reinvierá $(n + 1)$ messaggi al suo nodo genitore. Volendo si puó definire tale metodo come metodo con reinvio totale dei messaggi. Il secondo metodo di reinvio dei messaggi che é stato considerato prende il nome di Sensor-Fusion. Come si puó intuire dal nome in questa metodologia di trasmissione di messaggi se all'istante k un nodo riceve n messaggi dai suoi nodi figli e riceve un messaggio di misura allora i dati ricevuti verranno elaborati dal nodo in questione che andrá a creare un messaggio riassuntivo contenete l'informazione complessiva che gli é arrivata. Tale messaggio riassuntivo verrá reinviato al proprio nodo genitore. Notiamo che con la tecnica Sensor-Fusion ad ogni istante k il numero di messaggi ricevuti da ogni nodo sará pari al numero di figli dello stesso che hanno inviato un messaggio piú il messaggio di misura che viene sempre ricevuto

da ogni nodo. L'utilizzo della tecnica Sensor-Fusion non é scontato infatti non é detto che sia sempre meglio usare tale tecnica rispetto a quella di reinvio totale dei messaggi. Un punto a favore della tecnica Sensor-Fusion consiste nella riduzione del numero di messaggi che transitano nella rete e quindi almeno da questo punto di vista si ha una riduzione del consumo energetico. Purtroppo peró con la tecnica Sensor-Fusion si ha una maggior quantitá di calcoli da dover fare per ogni sensore e quindi ció comporta un maggior dispendio energetico. La convenienza o meno di una delle due tecniche rispetto all'altra dovrá essere valutata caso per caso andando a prendere la soluzione che minimizza il dispendio energetico tenendo conto di tutti i fattori possibili.

2.7.4 Equazione di aggiornamento dello stato per reti con reinvio totale dei messaggi.

Procediamo ora a ricavare l'equazione di aggiornamento dello stato per reti con reinvio totale dei messaggi. A tale scopo introduciamo il vettore dei messaggi di misura all'istante discreto k:

$$U(k) = \begin{bmatrix} u_1(k) \\ \vdots \\ u_N(k) \end{bmatrix}$$

dove $u_i(k) = 1$ se all'istante k é arrivata una misura al nodo i altrimenti $u_i(k) = 0$.
Grazie all'introduzione di tale vettore l'equazione di aggiornamento dello stato risulta essere:

$$X(k+1) = M_a(M_g, X_r(k+1)) * (X(k) + U(k))$$

In tale equazione $(X(k) + U(k))$ rappresenta il vettore le cui componenti dicono quanti messaggi in generale sono arrivati ai vari nodi della rete all'istante k-esimo e $M_a(M_g, X_r(k+1))$ é la matrice che rappresenta la configurazione della rete all'istante successivo, essa é funzione di M_g e di $X_r(k)$ cioé del vettore che dice quale nodo é la radice all'istante k. Notiamo che $M_a(M_g, X_r(k+1)) * (X(k) + U(k))$ corrisponde alla operazione di reinvio totale dei messaggi e siccome per ipotesi abbiamo una comunicazione sicura allora i messaggi inviati all'istante k sono i messaggi che vengono ricevuti all'istante $(k+1)$. Per ipotesi prenderemo sempre **stato iniziale nullo**. Notiamo che se teniamo fissata la posizione della radice per ogni k allora tale sistema diventa un semplicissimo sistema lineare tempo invariante. Inoltre data la particolare struttura dell'ingresso che puo avere come elementi solo 0 oppure 1 si vede che lo stato di ogni nodo sará sempre inferiore a $2*N$ dove N é il numero di nodi della rete. Per ipotesi abbiamo supposto che ad ogni istante k ogni nodo riceva un messaggio di misura e quindi il vettore dei messaggi di misura diventa un vettore fatto da tutti elementi pari ad uno.

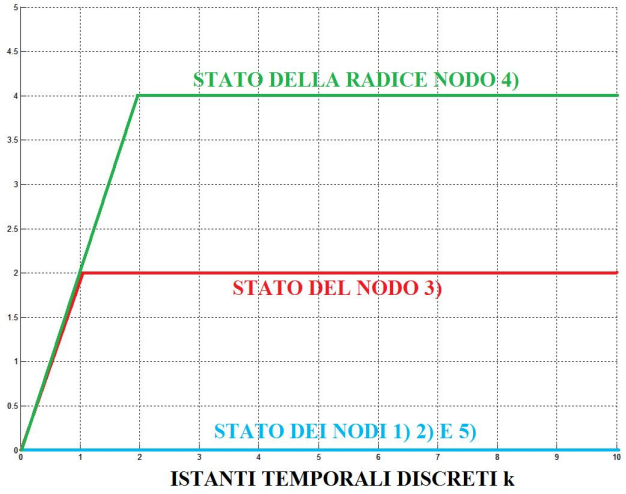

Figura 7. Evoluzione dello stato della rete con configurazione rappresentata in FIGURA 6, radice in posizione fissa e tecnica di reinvio totale dei messaggi.

Se invece la radice non viene tenuta fissa dobbiamo considerare come ingresso anche $X_r(k)$. Diventa ora molto interessante notare che $M_a(\bullet)$ é una funzione NON LINEARE dei suoi argomenti. Tale non linearita deriva dalla non linearitá della funzione $\Phi(\bullet)$ ed implica la non linearitá del sistema in considerazione. Inoltre tale sistema non lineare presenta anche una restrizione sulla gamma dei possibili valori che possono essere assunti dall'ingresso. Infatti se prendiamo per ipotesi che ad ogni istante k ogni nodo riceva sempre un messaggio di misura allora l'unico ingresso diventa in vettore $X_r(k)$ il quale ha tutte componenti nulle eccetto un elemento che vale uno.

Prima di poter fare un esempio con radice mobile dobbiamo notare che l'equazione di aggiornamento dello stato deve essere modificata. Tale equazione é senza dubbio corretta nel caso di radice fissa peró non da una rappresentazione appropriata della rete nel momento in cui consideriamo la radice libera di muoversi attraverso la rete stessa. Per comprendere questo fatto notiamo che se un nodo A é radice all'istante k-esimo, riceve n messaggi e all'istante $(k+1)$ tale nodo non sará piú radice allora seguendo la formula introdotta precedentemente il nodo A reinvierá i messaggi ricevuti piú il suo messaggio di misura verso quello che sará il nuovo nodo radice e quindi A reinvierá $(n+1)$ messaggi. Ció non é corretto perché se all'istante k il nodo A era radice allora tale nodo avrá eseguito delle operazioni sui dati che gli sono arrivati e quindi sará sufficiente che esso invii solamente un messaggio verso la nuova radice. Tale messaggio dovrá contenere l'informazione necessaria alla nuova radice per continuare i vari calcoli preposti alle radici. La formula di aggiornamento dello stato viene cosí modificata:

$$X(k+1) = M_a(k+1) * ((I - \Phi(X_r(k))) * X(k) + U(k))$$

Notiamo che se consideriamo il caso di radice fissa la ultima formula introdotta si riconduce alla prima. Notiamo inoltre che se un nodo é radice ad un certo istante ed all'istante successivo non é piú radice allora manda un solo messaggio e che se un nodo ad un certo istante non é radice ma all'istante successivo esso diventa radice allora non manda messaggi ma si tiene l'informazione che gli servirá per fare i calcoli all'istante successivo. Si é voluto in questa sede introdurre per gradi la formula di evoluzione di stato al fine di cercare di renderne meno pesante la comprensione. In definitiva l'ultima formula é quella corretta. Riportiamo in FIGURA 8 un esempio di evoluzione del sistema. In tale esempio durante l'evoluzione del sistema stesso il ruolo di radice é stato assegnato in sequenza ai nodi 4) 5) 3) e poi é rimasto al nodo 1). Si nota come la natura di tale sistema sia piuttosto caotica ed accenniamo che in un certo senso ci siamo messi in condizioni di una certa stabilitá per fare l'esempio in questione.

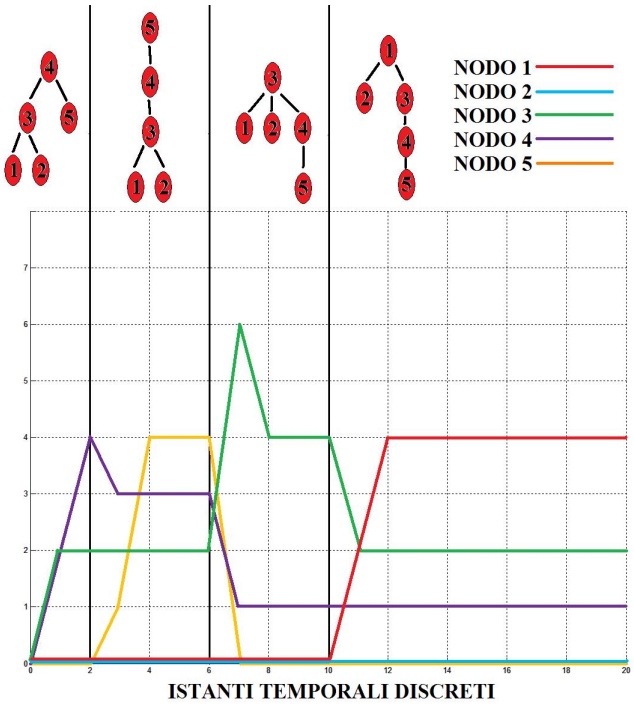

Figura 8. Evoluzione dello stato della rete con radice in posizione mobile e tecnica di reinvio totale dei messaggi.

2.7.5 Equazione di aggiornamento dello stato per reti con reinvio dei messaggi di tipo Sensor-Fusion.

In tale sede introdurremo la formula di aggiornamento dello stato per reti con reinvio dei messaggi di tipo Sensor-Fusion. L'idea base di tale tecnica é giá stata spiegata. Inoltre dopo quanto detto nella sezione appena precedente discende direttamente la seguente formula:

$$X(k+1) = M_a(M_g, X_r(k+1)) * \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}$$

Questa formula deriva direttamente dal fatto che nella tecnica Sensor-Fusion ogni nodo manda un solo messaggio ad ogni istante di invio. Notiamo che il sistema rimane non lineare e che grazie al prodotto per $M_a(\bullet)$ se un nodo all'istante $(k+1)$ sará radice allora tale nodo all'istante k non invia messaggi. Notiamo inoltre che lo stato all'istante $(k+1)$ non dipende dallo stato agli istanti precedenti.

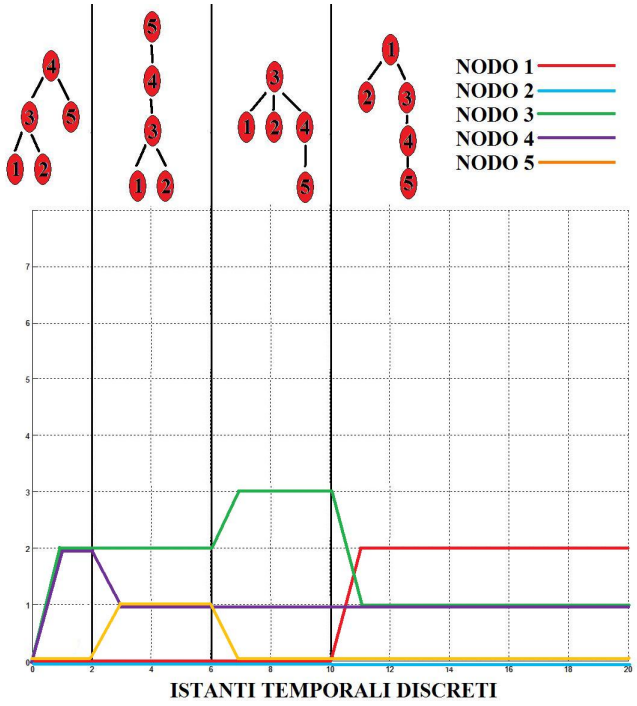

Figura 9. Evoluzione dello stato della rete con radice in posizione mobile e tecnica di reinvio dei messaggi Sensor-Fusion.

2.8 Considerazioni finali della sessione.

Come prima cosa notiamo che il problema di equidistribuire il carico di lavoro tra i vari nodi dell'albero si riduce alla ricerca di un opportuno segnale $X_r(k)$. Una possibile obbiezione a tutto il procedimento che stiamo descrivendo potrebbe riguardare il fatto che apparentemente i vari nodi non conoscono lo stato della rete, infatti potrebbe sembrare che ogni nodo conosca solamente il numero di messaggi da esso ricevuti e non sappia niente di quanto accade nel resto della rete. Ricordiamo però che lo stato iniziale della rete e le leggi di evoluzione dello stato stesso sono state ben definite quindi a meno di eventi aleatori ogni sensore può ricostruire almeno il linea di principio lo stato della rete e quindi in realtà tutti i sensori sanno quanti messaggi vengono ricevuti da ogni altro sensore. Ricordiamo che gli eventi aleatori sono stati da noi esclusi e quindi dovranno essere trattati in sede staccata attraverso tecniche complementari alla legge di controllo che troveremo. Tale legge seppur incognita per il momento appena essa sarà trovata verrà resa nota ai vari sensori. Possiamo in sostanza affermare che tutti gli ingredienti necessari ad una simulazione sono noti. Se tale calcolo può essere eseguito da un elaboratore e se tralasciamo il fatto che i sensori hanno diverse capacità di calcolo rispetto ad un computer allora gli stessi calcoli possono essere eseguiti dai sensori che così facendo si ricavano lo stato della rete.

Fino ad ora è solo stato accennato il fatto che nella costruzione della matrice M_a si devono assegnare i nodi di un certo livello come figli dei nodi del livello superiore secondo un certo criterio. Le simulazioni sono state eseguite con tre diversi criteri. Il primo consiste nel cercare di uniformare il numero di figli dei vari nodi di ogni livello dell'albero, tale criterio è stato indicato con il nome di FigliEquispaziati. Il secondo criterio che è stato indicato con il nome di PrimoUno invece per assegnare i figli ai loro genitori considera la generica riga di $(M_a)^T$ tiene il primo uno e mette a zero tutti gli altri. Questo criterio è stato preso in considerazione per la sua facilità di implementazione. Il terzo criterio è stato considerato solo per approfondire la conoscenza del significato delle varie simulazioni. Tale scelta assegna i figli ai vari candidati ad essere genitori in modo casuale. L'ultima tecnica prende il nome di FigliCasuali.

3 CONTROLLO DELLA RETE CON REINVIO TOTALE DEI MESSAGGI TRAMITE TECNICA UNIFORMITÁFLUSSO.

3.1 Introduzione all'approccio di controllo.

In tale sede ci prepariamo ad affrontare la ricerca della legge di controllo che impone quale nodo debba essere la radice al variare degli istanti temporali di evoluzione della rete. Quanto si é fatto cerca almeno in linea di principio di ricalcare il procedimento giá visto per il controllo ottimo. Nella fattispecie peró siamo in presenza di una tipologia differente di sistemi rispetto a quella vista in tale metodologia di controllo. Ricordiamo a tale scopo che il sistema in questione é un sistema fortemente non lineare nel quale lo stato e l'ingresso sono vincolati a poter assumenre solamente una limitata gamma di valori che appartengono quindi ad insiemi dotati di limitate proprietá geometriche. Senza entrare troppo nel dettaglio notiamo che l'insieme dei valori assumibili dall'ingresso, ovvero l'insieme dei vettori a componenti nulle ad eccezione di una che vale uno, non é nemmeno un gruppo abelliano secondo le canoniche operazioni algebriche.
Il procedimento seguito ha come punto di partenza la definizione di un funzionale di costo appropriato e da tale funzionale di costo attraverso un procedimento di minimizzazione si ricaverá la legge di controllo cercata. Chiameremo la tecnica di controllo che definiremo nel seguito con il nome di tecnica Uniformitá flusso.

3.2 Definizione del funzionale di costo J(•).

Ricordiamo ora che il nostro obbiettivo é far consumare alla rete la minor energia possibile e fare in modo che il consumo di energia sia il maggiormente possibile equidistribuito tra i vari nodi. Introduciamo a tale scopo due funzionali che serviranno per definire il funzionale da minimizzare.

3.2.1 Il funzionale di costo $J_0(\bullet)$.

Nel caso di evoluzione della rete con reinvio totale di messaggi si puó pensare che la energia spesa da un singolo nodo all'istante T sia proporzionale al numero di messaggi ricevuti ed inviati in tale istante. Notiamo quindi che il vettore la cui componente j-esima rappresenta la energia spesa dal nodo j-esimo a partire dall'istante 0 fino all'istante T si puó scrivere a meno di una costante moltiplicativa come:

$$V_T = \sum_{k=0}^{T}[X_R(k) + X_i(k) + U(k)] + X_R(T+1)$$

dove $X_R(k)$ é il vettore di stato che indica il numero di messaggi ricevuti all'istante k inviati da altri nodi, $U(k)$ invece é il vettore dei messaggi di misura che per ipotesi viene preso con componenti tutte pari ad uno mentre $X_i(k)$ é il vettore la cui componente j-esima indica il numero di messaggi inviati dal nodo j-esimo all'istante k. L'ultimo vettore si puó scrivere come:

$$X_i(k) = [I - \Phi\left(X_r(k+1)\right)] * \\ \{[I - \Phi\left(X_r(k)\right)] * X_R(k) + U(k)\}$$

dove $X_r(k)$ é il vettore a componenti nulle ad eccezione della componente in corrispondenza al nodo che é radice all'istante k la quale assume valore uno. Marchiamo ora la formula appena scritta perché visto che abbiamo una comunicazione sicura si potrebbe pensare ERRONEAMENTE che i messaggi che invio ad un istante arrivano tutti a destinazione all'istante successivo e da questo fatto trarre la conclusione SBAGLIATA che $X_i(k) = X_R(k+1)$. La formula corretta precedentemente scritta seppur piuttosto semplice potrebbe non venire compresa perfettamente se non risulta chiaro il **procedimento di invio dei messaggi**.

Ribadiamo ora tale procedimento introdotto in sezione 2.5.3 :

- Scatta l'istante discreto k
- Tutti i nodi ricevono i messaggi $X_R(k)$ all'istante discreto k ed anche i loro messaggi di misura $U(k)$
- I messaggi ricevuti all'istante k vengono preparati per il prossimo invio. Il nodo j-esimo che é radice all'istante k elabora i messaggi ricevuti $X_R^{(j)}(k)$ ed il messaggio di misura che ha anch'esso ricevuto.
- Quindi all'istante k i nodi inviano i messaggi verso la radice all'istante $(k+1)$ definita da $(X_r(k+1))$. Il nodo che sará radice all'istante $(k+1)$ invia zero messaggi all'istante k. Se il nodo che é radice all'istante k non é radice all'istante $(k+1)$ allora tale nodo invia un unico messaggio risultato dalle sue operzioni.

Commentiamo ancora la scelta fatta per il vettore V_T. Ammettiamo che potrebbe sembrare poco chiaro il motivo della presenza di $X_R(T+1)$ all'interno della formula che definisce V_T. Tale scelta é stata fatta al fine di includere nel vettore V_T tutti i termini che rappresentano un dispendio energetico i quali dipendono da $X_r(k)$ con k che arriva fino a $(T+1)$. Se noi avessimo preso $V_T = \sum_{k=0}^{T}[X_R(k) + X_i(k) + U(k)]$ avremmo avuto in tale somma la presenza di termini dipendenti da $X_r(k)$ con k che arriva fino a $(T+1)$, peró anche $X_R(k+1)$ dipende da $X_r(k+1)$ quindi prendiamo anch'esso al fine di inglobare in V_T tutti e soli i termini definiti completamente da $X_r(k)$ con k che va da 0 a $(T+1)$. Per dare ridondanza all'informazione rispieghiamo quanto appena detto in termini differenti. Quello che vogliamo fare in sostanza é prendere una successione di $X_r(k)$, avere un funzionale che dica quanta energia totalmente spendo in funzione della scelta dei $X_r(k)$ (quindi devo avere anche $X_R(T+1)$) e trovare quale successione di $X_r(k)$ minimizza il funzionale. **Notiamo che non si sa a priori se il minimo esiste**. Per definizione prendiamo come funzionale di costo J_0 il seguente:

$$J_0(T) = (V_T)^T * V_T$$

Il quale non é altro che il quadrato della norma del vettore la cui componente j-esima rappresenta la energia spesa dal nodo j-esimo a partire dall'istante 0 fino all'istante T.

3.2.2 Il funzionale di costo $J_1(\bullet)$.

Notiamo che la richiesta di equidistribuzione del dispendio energetico nel corso dell'evoluzione della rete equivale a richiedere che nel corso della evoluzione della rete stessa la quantitá di energia spesa dal singolo nodo vari nel tempo ma sia una costante tra i vari sensori. Idealmente si vorrebbe quindi che tutti i nodi abbiano sempre lo stesso livello energetico o equivalentemente che tutti i nodi consumino sempre la stessa energia. Ricordiamo che V_T é il vettore rappresentante il livello energetico dei vari nodi. Quanto appena detto si traduce nella richiesta di voler minimizzare in un certo senso la varianza delle componenti del vettore V_T. Definiamo il seguente vettore:

$$V_T^{Media} = \frac{[1\cdots1]}{N} * V_T * \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}$$

Dove N é il numero di stati del sistema. Notiamo che tale vettore ha ogni elemento pari alla media aritmetica di tutti gli elementi del vettore V_T. Notiamo inoltre che il vettore $(V_T - V_T^{Media})$ non é a componenti tutte positive.
Dopo quanto detto fino ad ora risulta immediata l'introduzione del funzionale $J_1(\bullet)$:

$$J_1(T) = (V_T - V_T^{Media})^T * (V_T - V_T^{Media})$$

3.2.3 Il funzionale di costo $J(\bullet)$.

Ricordiamo che il nostro obbiettivo é quello di limitare il dispendio complessivo della rete, quindi minimizzare $J_T(\bullet)$ possibilmente per qualsiasi T, e limitare la differenza tra i vari livelli energetici dei sensori componenti la rete, quindi minimizzare $J_1(\bullet)$. Notiamo che minimizzare uno dei due funzionali appena ricordati non significa minimizzare anche l'altro. Infatti i vari sensori potrebbero essere governati da una legge di controllo che faccia spendere a loro la stessa quantitá di energia la quale potrebbe essere molto elevata. Inoltre si potrebbe anche pensare ad

$$
\begin{aligned}
J(T) = \quad & J_0(T) + J_1(T) \\
= \quad & (V_T)^T * V_T + (V_T - V_T^{Media})^T * (V_T - V_T^{Media}) \\
= \quad & (V_T)^T * V_T + \left(V_T - \tfrac{[1\cdots1]}{N} * V_T * \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \right)^T * \left(V_T - \tfrac{[1\cdots1]}{N} * V_T * \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \right) \\
= \quad & (V_T)^T * V_T + \left(\left(I_N - \tfrac{ones(N)}{N} \right) * V_T \right)^T * \left(I_N - \tfrac{ones(N)}{N} \right) * V_T \\
= \quad & (V_T)^T * V_T + V_T^T * \left(I_N - \tfrac{ones(N)}{N} \right)^2 * V_T \\
= \quad & (V_T)^T * V_T + V_T^T * \left(I_N - \tfrac{ones(N)}{N} \right) * V_T \\
= \quad & V_T^T * \left(2 * I_N - \tfrac{ones(N)}{N} \right) * V_T \\
= \quad & \left(\underbrace{\sum_{k=0}^{T} [X_R(k) + X_i(k) + U(k)] + X_R(k+1)}_{A} \right)^T * \left(2 * I_N - \tfrac{ones(N)}{N} \right) * (A)
\end{aligned}
$$

una legge di controllo che nel complesso faccia spendere poca energia alla rete ma purtroppo faccia spendere la maggior parte di tale energia a pochi nodi che quindi si scaricano velocemente. Evidentemente si dovrá cercare di minimizzare una combinazione dei due funzionali. La scelta fatta é stata quella di considerare il seguente funzionale:

$$J(T) = J_0(T) + J_1(T)$$

Notiamo che dalla catena di relazioni introdotta ad inizio pagina si riesce a riscrivere il funzionale $J(T)$ come una forma quadratica in $(V_T)^T$. Viene lasciato al lettore verificare che la matrice dei pesi $(2 * I_N - ones(N)/N)$ é definita positiva. Infatti essa ha un autovalore pari ad uno ed $(N-1)$ autovalori pari a due.
Arrivati a questo punto dobbiamo lasciare ogni speranza di trovare una legge formale che minimizza il funzionale J. Infatti se avessimo a disposizione un sistema anche non lineare il quale non é soggetto al vincolo sui possibili valori dello stato e dell'ingresso che invece é presente nel sistema da noi considerato allora si potrebbe almeno tentare di minimizzare tale funzionale attraverso l'utilizzo di gradienti o operazioni simili. Purtroppo peró ció non é possibile ed inoltre non si riescono a trovare altre tecniche utili per determinare la legge formale di controllo $X_r(k)$ che minimizza il funzionale J(T) per ogni T.

3.3 Minimizzazione del funzionale di costo J(•).

Nella fase di sviluppo del progetto come prima cosa si é tentato di minimizzare $J(T)$ cercando la legge formale $X_r(k)$ di evoluzione della posizione della radice espressa in funzione dello stato del sistema $X(k)$. Come prima cosa si é riscritto il funzionale come $J(T) = J(T-1) + A(T)$ dove $A(T)$ é una funzione che é risultata dai calcoli. Notiamo che $J(T-1)$ dipende da $X_r(k)$ con k che arriva fino a T mentre $J(T)$ dipende da $X_r(k)$ con k che arriva fino a $(T+1)$. Dopo aver fatto ció abbiamo preso per ipotesi l'**esistenza della legge di controllo** $X_r(k)$ **ottima che minimizza** $J(T)$ **qualsiasi T**. Tale ipotesi verrá verificata a posteriori. Fissato tale punto ci si é avvalso di uno dei concetti fondamentali del Controllo Ottimo:
se esiste la legge di controllo ottimo $X_r(k)$ con $0 \leq k \leq (T+1)$ che minimizza $J(k)$ per $0 \leq k \leq T$ allora la stessa legge $X_r(k)$ con invece $0 \leq k \leq T$ minimizza $J(k)$ per $0 \leq k \leq (T-1)$. Quindi fissata la legge di controllo ottima $X_r(k)$ con $0 \leq k \leq T$ per minimizzare $J(T)$ basterá trovare $X_r(T+1)$ che minimizza $A(T)$. Non

é necessario riportare la formula di $A(T)$ per capire che essa é una funzione complicata nella quale compare pesantemente la funzione $\Phi(\bullet)$. Il calcolo a mano necessario per minimizzare $A(T)$ si é rivelato troppo avanzato e quindi é stata intrapresa una strada algoritmica per tale minimizzazione.

3.3.1 Algoritmo di minimizzazione di $J(\bullet)$.

Ricordiamo che per ipotesi si ha l'esistenza della legge di controllo $X_r(k)$ ottima che minimizza $J(T)$ qualsiasi T. Quindi fissato $X_r(k)$ con $0 \leq k \leq T$ che minimizza $J(T-1)$ si vuole trovare:

$$X_r(T+1) = \\ argmin\left\{J(T)\ con\ X_r(0)\dots X_r(T)\ fissati\right\}$$

Notiamo che si puó scrivere la forma ricorsiva $V_T = V_{T-1} + X_i(T) + U(T) + X_R(T+1)$ dove solamente $X_i(T)$ ed $X_R(T+1)$ dipendono da $X_r(T+1)$. Quello che si fa é calcolare tutti i possibili valori di $V_T(X_r(T+1))$ i quali sono in numero pari al numero N di sensori componenti la rete. In corrispondenza ai valori appena trovati si va a valutare $J(V_T(X_r(T+1)))$. Trovati questi ultimi N valori si va ad individuare il minimo di essi e come $X_r(T+1)$ si prende quello che genera il valore minimo di $J(V_T(X_r(T+1)))$ appena trovato. Notiamo che non é detto che la radice venga spostata ad ogni istante discreto k anché se la procedura di minimizzazione appena indicata viene compiuta ad ogni istante k.

Le simulazioni sono state eseguite con tutti i possibili criteri di scelta dei figli, ovvero criterio FigliEquispaziati, criterio FigliCasuali e con il criterio di scelta PrimoUno. Nel seguito viene solamente riportato il codice MATLAB del criterio FigliEquispaziati in quanto il codice degli altri casi é identico. Prima di riportare il codice peró é necessario dire che l'algoritmo appena descritto é stato modificato per rendere minore il tempo impiegato dai messaggi per arrivare alla radice.

3.3.2 Algoritmo di minimizzazione modificato di $J(\bullet)$.

L'algoritmo appena descritto in linea di principio potrebbe spostare la radice ad ogni iterazione. Risulta abbastanza ovvio che almeno dal punto di vista teorico se la radice viene spostata ad ogni iterazione alcuni messaggi potrebbero transitare a lungo attraverso la rete prima di incontrare un nodo con il ruolo di radice. Per aggirare questo problema invece che ad ogni istante k si potrebbe **spostare la radice trascorso un tempo definito TempoInteradice**. Nelle simulazioni si sono provati vari TempoInteradice differenti inoltre si é fatta una simulazione con un certo TempoInteradice minimo. Nell'ultima simulazione menzionata quando si sposta la radice si calcola il numero di livelli del nuovo albero e prima di spostare la radice si aspetta un TempoInteradice pari al numero di livelli. In questo modo si é sicuri che **il numero di volte che un messaggio viene reinviato prima di raggiungere la radice é minore del numero di nodi dell'albero**.

3.4 Tutorial e commenti dei risultati della simulazione.

Per fare girare la simulazione basta mettere come **Current Directory** in **MATLAB** la cartella **SimulazioneControlloUniformitaFlussoTempoInterRadice**. Aperta tale directory si deve avviare il file **SimulazioneControlloUniformitaFlussoTempoInterRadice.m** . Si aprirá una finestra con dei grafici. Osservati tali grafici si puó premere un tasto qualunque della tastiera dal workspace (come se si stesse scrivendo un comando sul workspace), si chiuderá la finestra e se ne aprirá un'altra. Si hanno nel complesso sei finestre da aprire. La simulazione che segue é stata eseguita sul particolare grafo rappresentato in FIGURA 10. Si dará la possibilitá di modificare il grafo della rete nella simulazione finale.

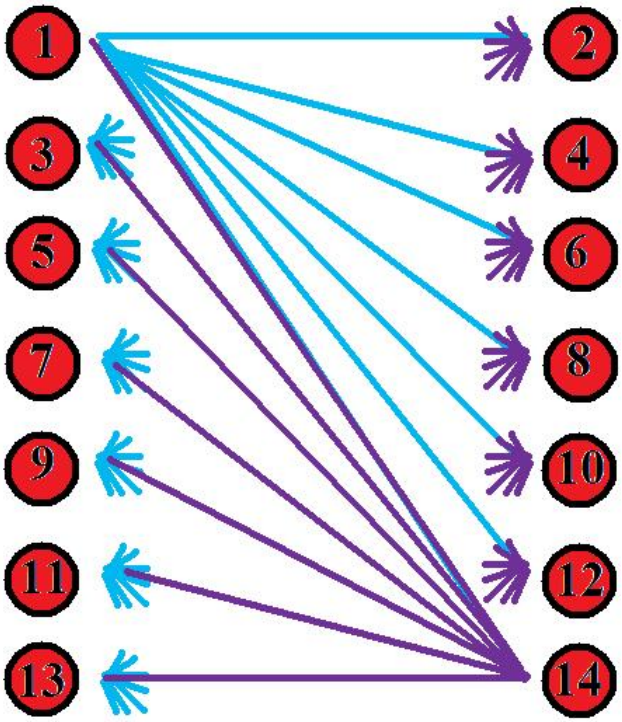

Figura 10. Nel grafo utilizzato nella simulazione si hanno nodi pari e nodi dispari. Se un nodo é pari puó comunicare solo con uno qualsiasi dei nodi dispari e viceversa. M_g é una matrice fatta a scacchiera.

3.4.1 Finestra 1 / Finestra 2.

Nelle prime due finestre che si possono osservare vengono riportate le evoluzioni dello stato del sistema retroazionato nei vari casi di scelta della tecnica di distribuzione dei nodi figli di un livello (FigliEquispaziati, ...). Nella prima finestra si usa una retroazione con TempoInteradice pari ad 1 mentre nella seconda si utilizza per ogni istante di simulazione un TempoInteradice pari al numero di livelli dell'albero che rappresenta la rete in tale istante.
Il tempo di simulazione é pari a 1000 quindi

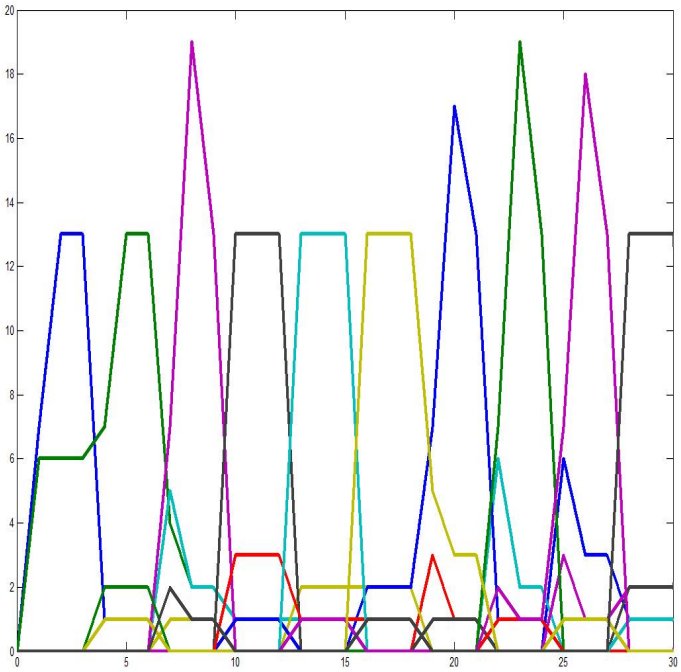

Figura 11. Ogni segnale rappresenta la stato di uno dei nodi della rete, cioé il numero di messaggi ricevuti dal generico nodo per ogni istante. Si hanno 14 nodi.

viene lasciata la possibilitá di osservare lo stato della rete in istanti differenti da quelli che compaiono nella schermata iniziale. Notiamo che il grafico é dello stesso tipo di quello presente in FIGURA 8. Inoltre come ci si puó aspettare l'evoluzione con TempoInteradice unitario é molto piú caotica di quella presente nella seconda schermata visto che nella prima la rete viene modificata molte piú volte di quanto accade nella seconda. Visionando l'output della simulazione si nota che la tecnica PrimoUno da risultati molto peggiori di quanto ci si potesse aspettare. Infatti dal grafico delle schermate della simulazione di nota che ci sono alcuni nodi della rete che durante tutta la simulazione permangono nello stato di foglia. Evidentemente ci si aspettava una prestazione

peggiore di tale tecnica rispetto a quella delle altre visto che la scelta dei figli PrimoUno al fine di ottenere una simulazione piú veloce comporta una asimmetria dell'albero rappresentante la rete. Ad ogni modo si sperava che la retroazione potesse compensare tale svantaggio di questa tecnica. Purtroppo questa speranza non é stata realizzata e la tecnica PrimoUno si é rivelata inefficiente rispetto alle altre.

3.4.2 *Finestra 3 / Finestra 4.*

Nelle schermate 3 e 4 si puó osservare l'evoluzione del funzionale $J(\bullet)$ nella simulazione in esame con tempo di simulazione pari a 1000. Nella Finestra 3 sono presenti i grafici relativi alle tre tipologie di scelta dei figli di un livello. Tali grafici sono simili a quello riportato in FIGURA 12. Notiamo come prima cosa che tali grafici

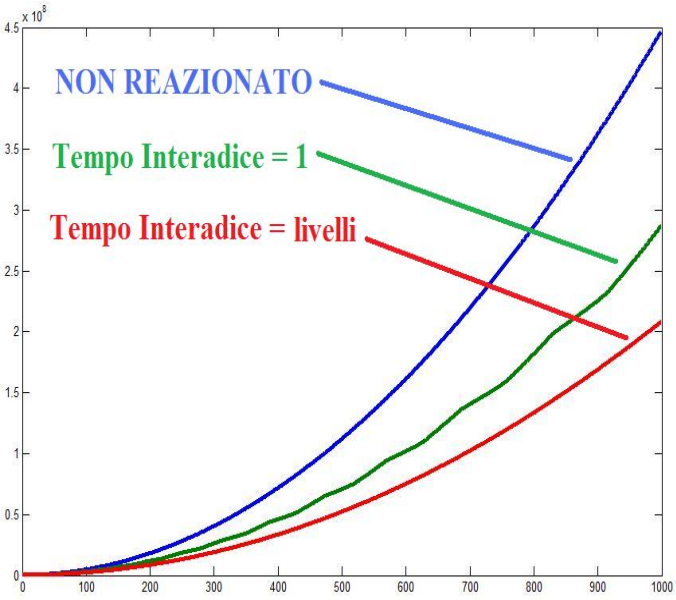

Figura 12. Evoluzione di $J(\bullet)$ con sistema NON REAZIONATO, con TempoInteradice pari a 1 e con TempoInteradice pari al numero di livelli dell'albero.

presentano un andamento quasi parabolico. Questo fatto é senza dubbio corretto in quanto $J(\bullet)$ é una forma quadratica su V_T il quale nel migliore dei casi aumenta all'aumentare di T in modo lineare. Notiamo inoltre che l'evoluzione non reazionata presenta valori del funzionale maggiori delle altre due. Quindi **la retroazione trovata migliora le prestazioni del sistema**.

Notiamo ora un fatto importantissimo: l'evoluzione con TempoInteradice unitario da un grafico di $J(\bullet)$ (linea verde di FIGURA 12) che sta sopra il grafico con TempoInteradice pari al numero di livelli dell'albero (linea rossa di FIGURA 12). Ricordiamo brevemente che avevamo preso come ipotesi l'esistenza della legge di controllo che minimizza $J(T)$ qualsiasi T e da questa ipotesi avevamo ricavato tale legge. Questa legge é appunto quella con TempoInteradice pari ad 1. L'altra legge con TempoInteradice diverso da 1 dovrebbe essere quindi una legge di controllo subottima, quindi peggiore della prima. Visto che si é verificato l'opposto di quanto ci si aspettava allora significa che é falsa l'ipotesi di partenza. Quindi:

NON ESISTE LA LEGGE DI CONTROLLO OTTIMA

Nella quarta schermata vengono plottati anche i grafici con altri TempiInteradice si puó notare che negli istanti iniziali tali grafici coincidono con il grafico della simulazione non reazionata. Infatti negli istanti compresi tra l'istante iniziale di simulazione ed il primo istante pari a TempoInteradice si ha che il sistema va in evoluzione libera e quindi si comporta come il sistema non reazionato.

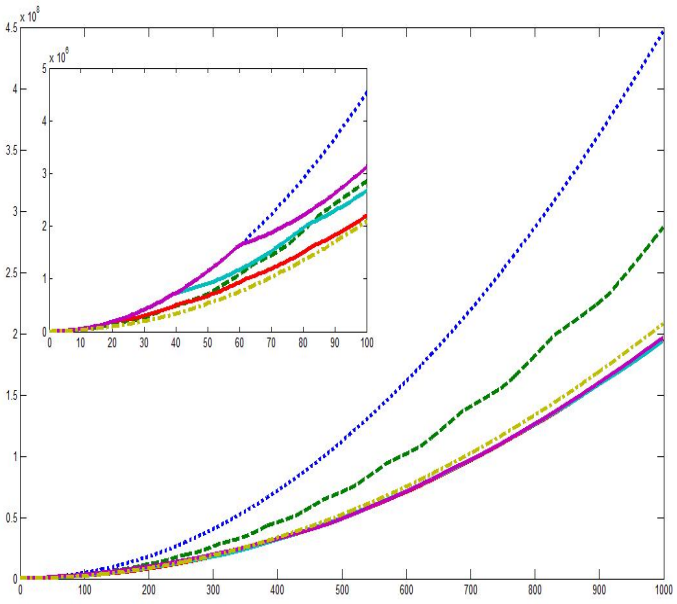

Figura 13. Evoluzione di $J(\bullet)$ con sistema NON REAZIONATO, con TempoInteradice pari a 1, con TempoInteradice pari al numero di livelli dell'albero ed altri TempoInteradice maggiori.

In FIGURA 13 i grafici tratteggiati sono quelli che vengono rappresentati anche in FIGURA 12. Notiamo che le evoluzioni aggiunte in FIGURA 13 hanno prestazioni quasi uguali tra loro e leggermente migliori della simulazione con TempoInteradice pari al numero di livelli dell'albero. Diamo ora una spiegazione intuitiva di questi fatti.
Visto che non esiste una legge di controllo che minimizza il funzionale per ogni T é possibile, ma soprattutto é ció che succede in simulazione, che prese due leggi di controllo quella che magari minimizza $J(\bullet)$ nei primi istanti di simulazione si rivela piú scadente dell'altra legge in tutti gli istanti successivi. In un certo senso sembrerebbe sia meglio avere una legge di controllo scadente all'inizio dell'evoluzione della rete la quale poi si rivela buona per altri istanti di simulazione.

3.4.3 Finestra 5 / Finestra 6.

Nelle ultime due schermate della simulazione si ha una rappresentazione REAL-TIME della energia spesa (V_T) dai vari nodi della rete. In FIGURA 14 si vede quale tipo di grafico

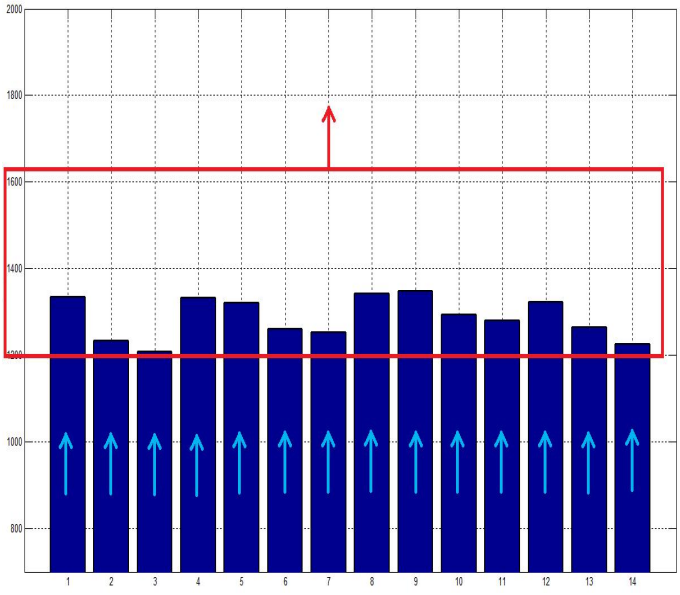

Figura 14. Evoluzione REAL-TIME. Ogni istogramma rappresenta l'energia spesa (V_T) dal singolo nodo a partire dall'istante iniziale di evoluzione della rete.

comparirá nella simulazione. Notiamo che gli istogrammi di figura aumentano in altezza al passare del tempo. Per rendere piú leggibile il grafico nelle schermate di tale simulazione si é deciso di rappresentare solo parte dell'istogramma di figura, piú precisamente la parte interna al riquadro rosso. Tale riquadro viene "appoggiato" sull'istogramma piú basso. In questo modo viene resa piú evidente la differenza in altezza tra i vari istogrammi rappresentanti l'energia spesa dai vari nodi, inoltre l'altezza dei vari istogrammi puó essere comunque letta sull'asse y del grafico. Da questa simulazione viene resa ancora piú evidente la minore qualitá della prestazione con criterio di scelta PrimoUno rispetto alle simulazioni con gli altri due criteri.

4 CONTROLLO DELLA RETE CON REINVIO TOTALE DEI MESSAGGI TRAMITE TECNICA MASSIMABATTERIA.

Nella ultima sezione uno dei risulti trovati é stata la mancanza della esistenza di una legge di controllo ottima. Tale fatto ci obbliga a cercare altre leggi di controllo che magari soddisfano delle condizioni tali da renderle delle buone candidate per un possibile utilizzo. In questa sezione si é sviluppata la tecnica di controllo definita MassimaBatteria.

4.1 Definizione della tecnica MassimaBatteria.

Tale tecnica di controllo é molto semplice e si basa sulla idea intuitiva che in prossimitá della radice i sensori si scaricano piú velocemente. Quando si sposta il nodo radice ció che fa la tecnica di controllo MassimaBatteria é posizionare la radice in corrispondenza al nodo con livello energetico maggiore. Quindi tale tecnica sposta la radice sul nodo che ha consumato meno energia ovvero sul nodo in corrispondenza al quale si ha la componente del vettore V_T precedentemente definito di valore minore.

4.2 Tutorial e commenti dei risultati della simulazione.

Per fare girare la simulazione basta mettere come **Current Directory** in **MATLAB** la cartella **SimulazioneControlloMassimaBatteriaTempoInterRadice**. Aperta tale directory si deve avviare il file **SimulazioneControlloMassimaBatteriaTempoInterRadice.m**. Si avranno da visionare due schermate attraverso le stesse modalitá introdotte per la simulazione UniformitáFlusso. La simulazione é stata eseguita sul grafo di FIGURA 10.

4.2.1 Finestra 1.

In tale schermata si trova l'evoluzione del funzionale $J(\bullet)$ con le tre diverse metodologie di scelta dei figli.

Si puó notare nelle simulazioni che tale legge

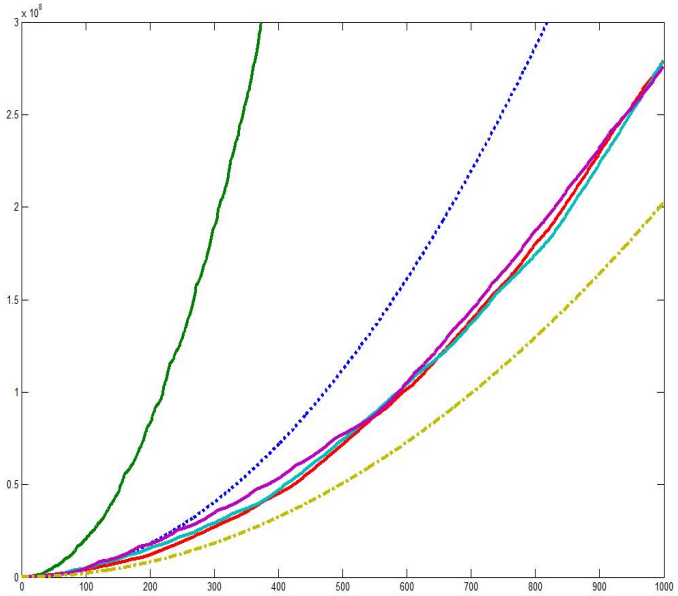

Figura 15. Evoluzione del funzionale $J(\bullet)$.

di controllo se attuata ad ogni istante di simulazione (linea verde di FIGURA 15) destabilizza in un certo senso il sistema. In tale sistuazione accade che alcuni messaggi si trovano a dover fare dei percorsi piuttosto lunghi per raggiungere la radice. Intuitivamente succede che la radice viene spostata troppo spesso ed in un certo senso scappa da alcuni messaggi. Tale situazione evidentemente non é accettabile. Notiamo ancora che tra i vari grafici presenti in FIGURA 15 quello migliore (cioé quello tratteggiato e giallo) é quello con il TempoInteradice pari al numero di livelli dell'albero. Aumentando invece il TempoInteradice si vede dagli altri grafici presenti in figura che le prestazioni degradano restando comunque migliori della situazione non reazionata (grafico tratteggiato blu di FIGURA 15). Se si confrontano i risultati di tale tecnica con quelli della tecnica UniformitáFlusso si puó notare che essi sono dello stesso ordine di grandezza

quindi la tecnica MassimaBatteria in alcuni casi potrebbe essere utilizzata al posto di quella introdotta nella sezione precedente in quanto richiede un minor numero di calcoli.

4.2.2 *Finestra 2.*

Nella seconda schermata della simulazione si puó vedere invece l'evoluzione temporale della energia spesa dal nodo piú scarico della rete. Si riporta nel grafico quindi istante per istante il minimo tra i valori delle componenti di V_T. Dalle simulazioni si puó notare come tale

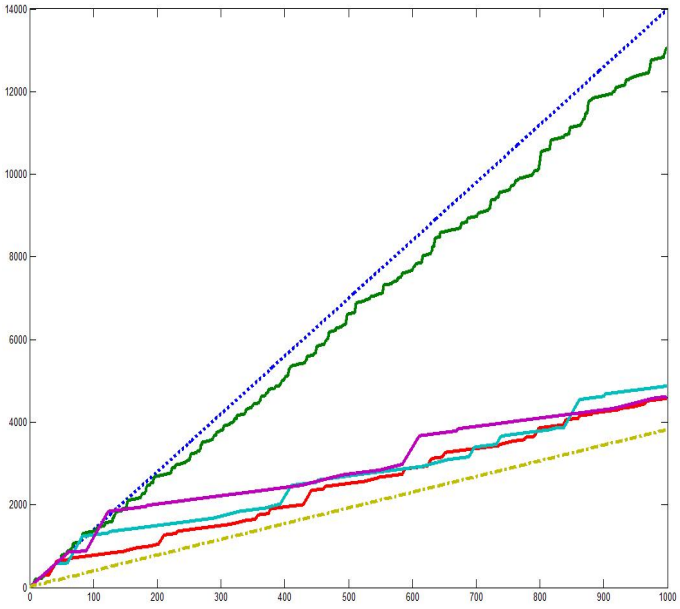

Figura 16. Evoluzione temporale della energia spesa dal nodo piú scarico della rete.

tecnica con TempoInteradice pari al numero di livelli (linea tratteggiata gialla) dia tra le varie alternative possibili i risultati migliori. Notiamo inoltre che in FIGURA 15 la tecnica in questione con TempoInteradice pari a 1 presenta valori di $J(\bullet)$ maggiori di quelli del sistema non retroazionato peró in FIGURA 16 il grafico della prima situazione menzionata sta sotto a quello della evoluzione non retroazionata. Notiamo che in linea di principio ci si poteva aspettare anche il contrario.

5 Controllo della rete con reinvio totale dei messaggi tramite tecnica FoglieScariche.

Accenniamo velocemente tale tecnica la quale é molto simile alla tecnica MassimaBatteria. Nella seguente metodologia di controllo quando si sposta la radice si da tale ruolo al nodo che estrae un albero rappresentativo della rete con somma della energia spesa dalle foglie massima. Le operazioni da fare sono le seguenti: per ogni nodo della rete estraggo l'albero rappresentativo della rete, considero le foglie del singolo albero, sommo le componenti del vettore V_T relative ai nodi foglia, ottengo N numeri con N pari al numero di nodi della rete, come radice prendo il nodo che ha generato il massimo tra tali N numeri. L'idea intuitiva é che le foglie sono i nodi che consumano meno e quindi con tale tecnica si cerca di far consumare meno ai nodi piú scarichi.

5.1 Tutorial e commenti dei risultati della simulazione.

Per fare girare la simulazione basta mettere come **Current Directory** in **MATLAB** la cartella **SimulazioneControlloFoglieScaricheTempoInterRadice**. Aperta tale directory si deve avviare il file **SimulazioneControlloFoglieScaricheTempoInterRadice.m**. Si avranno da visionare due schermate attraverso le stesse modalitá introdotte per la simulazione UniformitáFlusso. I risultati ottenuti sono stati simili a quelli del controllo MassimaBatteria. Nella simulazione finale si noterá che in alcuni casi patologici la tecnica di controllo in esame da risultati piuttosto scarsi rispetto alle altre due. La simulazione é stata eseguita sul grafo di FIGURA 10.

6 CONTROLLO DELLA RETE CON REINVIO DEI MESSAGGI DI TIPO SENSOR-FUSION TRAMITE TECNICA UNIFORMITÁFLUSSO.

6.1 Introduzione alla metodologia di controllo.

Nelle tecniche di controllo viste in precedenza si governa il funzionamento di reti di sensori che comunicano tra loro tramite il procedimento di reinvio totale dei messaggi. In tali reti quindi ogni nodo reinvia verso la radice tutti i messaggi che riceve dai suoi nodi figli ed inoltre viene reinviato anche il messaggio di misura ricevuto dal nodo stesso. All'inizio di tutta la trattazione riguardante la ricerca della legge di controllo della rete si era introdotto anche la tipoogia di rete con reinvio dei messaggi tramite Sensor-Fusion. Ricordiamo che in tali reti ogni nodo puó inviare al proprio genitore solamente un messaggio che racchiude in se tutta l'informazione ricevuta dal singolo nodo all'istante discreto k. Ogni nodo dovrá quindi fare delle operazioni definite di sintesi dei messaggi ricevuti. Un possibile esempio di una operazione di sintesi potrebbe essere fornito dal Filtro di Kalman. In tale esempio ogni nodo a partire dalle stime ricevute dai nodi figli e dal proprio messaggio di misura costruisce una stima che reinvia al proprio nodo genitore.
I ragionamenti che sono stati sviluppati per risolvere il problema del controllo di queste tipologie di reti sono sostanzialmente gli stessi giá visti per le reti con reinvio totale dei messaggi di sezione 3, cambiano solamente i significati di alcuni simboli che ridefiniremo nel seguito immediato. Ricordiamo che nel caso di utilizzo di reivio dei messaggi tramite Sensor-Fusion la equazione di aggiornamento dello stato della rete diventa:

$$X(k+1) = M_a(M_g, X_r(k+1)) * \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}$$

6.2 Il vettore V_T modificato.

Il vettore V_T nel caso di reinvio totale dei messaggi rappresenta l'energia spesa dai vari nodi a partire dall'istante iniziale di evoluzione della rete a meno di una costante moltiplicativa. Tale significato attribuito al vettore in questione viene mantenuto anche nel caso di reinvio dei messaggi con tecnica Sensor-Fusion. La differenza tra la definizione di V_T nei due casi citati risiede nella diversa metodologia di impiego dell'energia che questi presentano. Ricordiamo che nel caso di reinvio totale dei messaggi l'impiego di energia veniva attribuito attraverso una proporzionalitá diretta al numero di mesaggi ricevuti ed inviati dai vari nodi. Ora bisogna tener conto che l'energia attribuibile all'invio di un messaggio deve essere considerata minore rispetto a quella che si puó attribuire alla ricezione di un messaggio. Si suppone che a causa dei calcoli da compiere l'energia spesa in seguito all'arrivo di un messaggio sia $\alpha > 1$ volte quella necessaria per un invio se il nodo non é la radice mentre a causa di calcoli che si possono supporre piú laboriosi tale energia si pone $\beta > \alpha$ volte l'energia di invio per il nodo che é radice. Il vettore V_T diventa quindi:

$V_T = \sum_{k=0}^{T} [X_i(k) + C_R(k) + C_{NR}(k)] + X(T+1)$
dove $X_i(k)$ é il numero di messaggi inviati all'istante k e vale:
$X_i(k) = (I - \Phi(Xr(k+1))) * \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}$
$X(T+1)$ é il valore dello stato all'istante $(T+1)$ cioé il numero di messaggi ricevuti in tale istante e con $C_R(k)$ si é indicato il costo relativo ai messaggi ricevuti da parte del nodo radice:

$$C_R(k) = \beta\Phi(Xr(k)) \left(X(k) + \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \right)$$

Per finire specifichiamo che $C_{NR}(k)$ é il costo relativo ai messaggi ricevuti da parte dei nodi non radice:

$$C_{NR}(k) = \alpha(I - \Phi(Xr(k))) \left(X(k) + \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \right)$$

Ricordiamo inoltre che $X_r(k)$ é il vettore che indica quale nodo é la radice all'istante k.

6.3 Il funzionale $J(\bullet)$ modificato.

Ricordiamo ora la definizione del funzionale $J_1(\bullet)$ introdotto all'inizio della trattazione del controllo della rete:

$$J_1(T) = (V_T - V_T^{Media})^T * (V_T - V_T^{Media})$$

In tale formula si era posto:

$$V_T^{Media} = \frac{[1\cdots1]}{N} * V_T * \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}$$

Dove N é il numero di stati del sistema. Grazie a tale funzionale ed al funzionale $J_0(\bullet)$ si era riusciti a definire il funzionale finale $J(\bullet)$ come somma dei due. Notiamo che la presenza di $J_0(\bullet)$ non é piú necessaria a causa dell'impiego della tecnica Sensor-Fusion. Il ruolo del funzionale che ora viene omesso era quello di tenere il piú limitato possibile il numero di messaggi transitanti nella rete. Tale compito viene ora reso inutile perché nella tecnica Sensor-Fusion ogni nodo invia solo un messaggio ad esclusione del nodo che sará radice al passo che segue l'istante di invio. Quindi nella rete in questione con N nodi in ogni istante discreto k si nota l'invio di $(N-1)$ messaggi. Detto ció é ragionevole voler minimizzare il seguente funzionale:

$$J(T) = J_1(T)$$

Visto che il numero di messaggi circolanti é limitato allora si vuole solamente fare in modo che il vettore V_T presenti componenti di valore il piú vicino possibile ad un valore comune. Nei metodi precedenti questa scelta del funzionale da minimizzare non era sufficente perché in tale situazione vi era la possibilitá di trovare una legge di controllo che minimizzando il funzionale in questione (J_1) aumentasse tantissimo il numero di messaggi presenti nella rete. Fissiamo quindi il funzionale che utilizzeremo nel seguito come:

$$J(T) = J_1(T) = V_T^T * \left(I_N - \frac{ones(N)}{N} \right) * V_T$$

Notiamo che $J(T)$ é una forma quadratica semidefinita positiva in V_T. Viene lasciata al lettore la semplice verifica del fatto che la matrice di $J(T)$ presenta un solo autovalore nullo con relativo autovettore $[1\ldots1]^T$ mentre tutti gli altri autovalori sono pari ad 1.

6.4 Tutorial, commenti dei risultati e modifiche.

Per fare girare la simulazione basta mettere come **Current Directory** in **MATLAB** la cartella **SensorFusionUniformitaFlussoTempoInterRadice**. Aperta tale directory si deve avviare il file **SimulazioneUniformitaFlussoSensorFusionTempoInterRadice.m**. Si avranno da visionare due schermate attraverso le stesse modalitá introdotte per la simulazione UniformitáFlusso. La simulazione é stata eseguita sul grafo di FIGURA 10.

Portiamo subito il lettore a conoscenza del fatto che utilizzando il vettore V_T sopra definito in una prova di simulazione si é ottenuto un andamento del funzionale $J(T)$ con presenza di un drift con andamento parabolico. In FIGURA 17 il grafico blu tratteggiato rappresenta l'andamento di $J(T)$ nel caso non reazionato, i grafici giallo e verde sono rispettivamente quelli dei casi con retroazione con TempoInteradice pari al numero di livelli dell'albero e con TempoInteradice unitario. Negli ultimi due grafici si nota la presenza del drift parabolico accennato in precedenza. Gli altri grafici sono quelli relativi a retroazioni con differenti TempoInteradice. Negli ultimi grafici permane il drift parabolico e si nota che tra due cambi di radice consecutivi $J(T)$ é costituito da degli

archi di parabola che **non sempre comportano un aumento del valore del funzionale**. Questo ultimo fatto é di importanza notevole e trova spiegazione nel fatto che la matrice di $J(\bullet)$ in tale sede é **semidefinita positiva** mentre nel caso di reinvio totale essa era **definita positiva**.

Anticipiamo che anche in tale sede si puó ripetere il ragionamento giá fatto in sezione 3.5.2 relativo alla **non esistenza della legge di controllo ottima**, quindi é lecito cercare una legge diversa che magari riesce ad eliminare la presenza del drift parabolico che si é trovato.

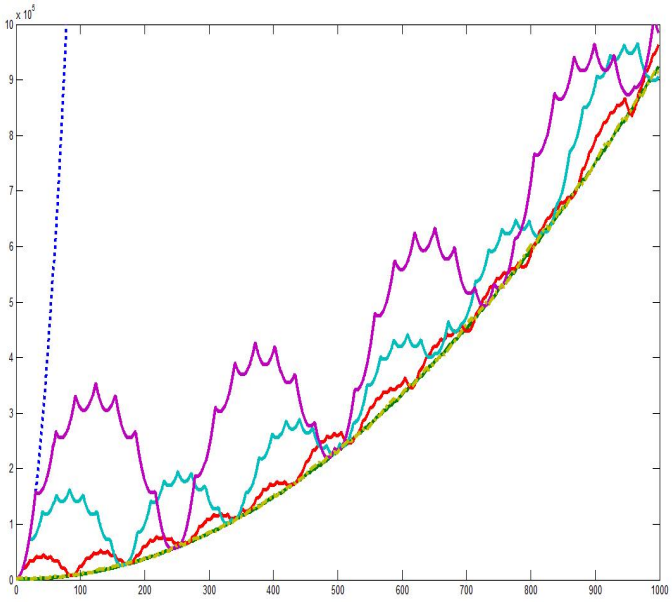

Figura 17. Evoluzione temporale di $J(T)$ non reazionato, e con retroazione con diversi TempoInteradice. Con presenza del drift

6.4.1 *Eliminazione del drift. Finestra 1*

In simulazione al fine di eliminare il drift parabolico si é utilizzata una legge di controllo che minimizza il funzionale $J(T)$ dove peró invece di V_T si é utilizzato un vettore leggermente differente:

$$V_M = \sum_{k=0}^{T} [C_R(k) + C_{NR}(k)] + X(T+1)$$

La differenza di tale vettore rispetto a V_T sta nella eliminazione del termine $X_i(k)$. Chiariamo subito che la legge di controllo é stata dedotta minimizzando $J(T)$ con la presenza di V_M peró i grafici della simulazione riportano l'andamento di $J(T)$ con la presenza di V_T in quanto si ritiene che V_T sia il miglior rappresentante dell'energia spesa dai vari nodi. I risultati riportati in FIGURA 18 sono simili a quelli di FIGURA 17 a meno del termine di **drift il quale é stato eliminato**. Nella prima schermata della simulazione si avranno dei grafici simili a quelli riportati in FIGURA 18.

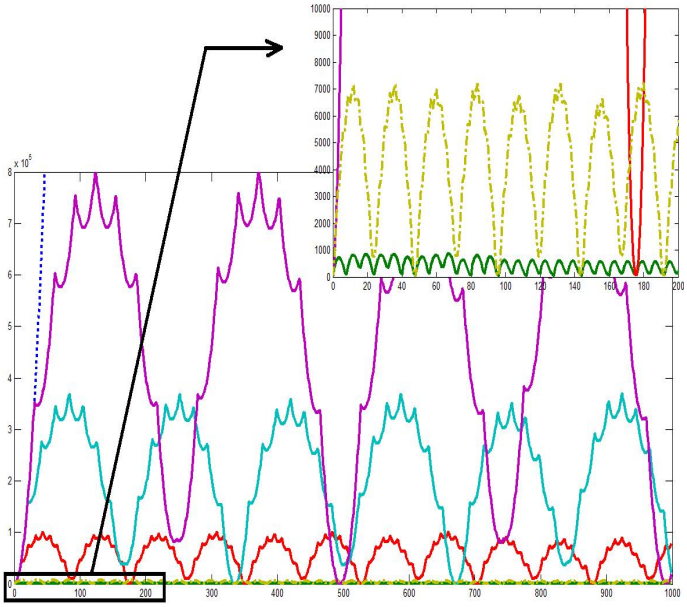

Figura 18. Evoluzione temporale di $J(T)$ non reazionato, e con retroazione con diversi TempoInteradice. Senza il drift parabolico.

6.4.2 *Finestra 2*

Nella seconda schermata della simulazione invece viene riportato per ogni istante di tempo discreto k l'energia spesa fino a quel momento dal nodo piú scarico della rete. Viene quindi riportato in un grafico il valore massimo delle componenti del vettore V_T per ogni istante di evoluzione della rete. Il grafico che si ottiene presenta andamenti giá trovati in precedenza. Notiamo che l'evoluzione con TempoInteradice unitario sembrerebbe essere quella che in entrambi i grafici da risultati migliori.

C'é purtroppo un problema che viene nascosto dall'utilizzo della tecnica Sensor-Fusion. Ricordiamo che nella trattazione sulle reti con reinvio totale dei messaggi si é parlato del problema di retroazioni che possono costringere alcuni messaggi a compiere percorsi molto lunghi prima di arrivare ad un nodo con la

carica di radice. Utilizzando la tecnica Sensor-Fusion questo fatto non si nota perché ogni nodo invia sempre un solo messaggio e quindi l'aumento del percorso che l'informazione deve compiere non si traduce in un aumento del numero dei messaggi che transitano nella rete. Anche se il numero di messaggi che transitano nella rete non aumenta e quindi non vi é un aumento della energia spesa tale situazione non é accettabile in quanto si vuole che l'informazione arrivi il piú velocemente possibile alla radice. Risulta quindi consigliabile l'utilizzo del TempoInteradice pari al numero di livelli dell'albero che rappresenta la rete in tale modo si garantisce che tutta la informazione giunge alla radice in un tempo pari al numero di nodi della rete.

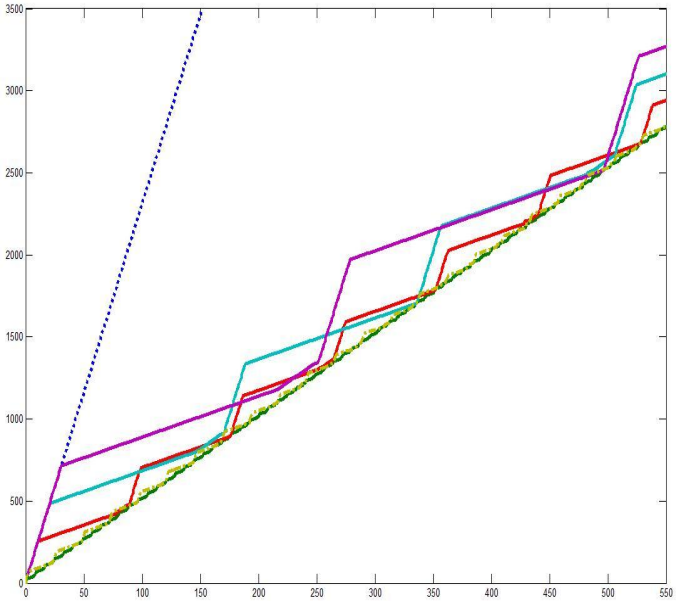

Figura 19. Energia spesa dal nodo piú scarico della rete nei diversi istanti di evoluzione nel caso non reazionato, e nel caso di retroazione con diversi TempoInteradice.

7 CONTROLLO DELLA RETE CON REINVIO DEI MESSAGGI DI TIPO SENSOR-FUSION TRAMITE LA TECNICA VALORE-FINALEMINIMO.

7.1 Introduzione alla tecnica ValoreFinaleMinimo.

In tale sede ci prepariamo ad introdurre una tecnica di controllo che si basa su una idea di fondo differente da quelle su cui si basano invece le tecniche di controllo precedentemente introdotte. In tali metodologie di controllo infatti l'obbiettivo era cercare di minimizzare per ogni istante di evoluzione della rete l'energia dissipata dai vari nodi e cercare di mantenere il livello energetico degli stessi il piú vicino possibile alla media tra i vari livelli energetici. Notiamo che tale situazione é sicuramente sufficente a garantire un buon funzionamento della rete ma in alcuni casi puó non essere necessaria. Infatti l'obbiettivo primario é riuscire a scaricare tutti i nodi nello stesso momento il piú distante nel tempo possibile dal tempo di inizio della evoluzione della rete. Notiamo che la tecnica Sensor-Fusion garantisce che il numero di messaggi inviati é invariante per tutti gli istanti di evoluzione quindi durante la evoluzione della rete non é importante che tutti i nodi presentino lo stesso livello energetico bensí é necessario solamente che nell'istante finale della evoluzione della rete tutti i nodi abbiano livello energetico nullo. Notiamo che a partire dalla conoscenza del livello energetico iniziale dei vari sensori é possibile stimare un tempo T tale che la rete sia dotata delle necessarie risorse energetiche per lavorare fino a tale istante e probabilmente oltre tale tempo non abbia piú tali risorse. Quello che si vuole é quindi che nell'istante T tutti i nodi abbiano speso a partire dall'istante iniziale la stessa energia. Ribadiamo i concetti introdotti fino ad ora: la tecnica Sensor-Fusion limita la energia spesa istante per istante, durante l'evoluzione della rete in alcuni casi non importa se qualche nodo si scarica piú o meno velocemente degli altri, il tempo T é il tempo finale di evoluzione della rete, si vuole che al tempo T tutti i nodi abbiano speso la stessa energia a partire dallo stato iniziale.

7.2 Deduzione formale della legge di controllo.

Notiamo che la equazione di aggiornamento dello stato per una rete con reinvio dei messaggi di tipo Sensor-Fusion puó essere riscritta come:

$$X(k) = M * V(k) * \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}$$

dove si ha:
$M = [M_a(X_r = x_1)| \dots |M_a(X_r = x_N)]$

$$V(k) = \begin{bmatrix} \oslash_N \\ \vdots \\ I_N(k) \\ \vdots \\ \oslash_N \end{bmatrix}$$

quindi se N é il numero di nodi che compongono la rete allora M é una matrice con N righe e N^2 colonne costituita da N blocchi quadrati di dimensione N dove il generico blocco j-esimo é costituito dalla matrice M_a ottenuta ponendo come radice il nodo j-esimo. V(k) invece é una matrice con N^2 righe e N colonne, é costituita da N blocchi quadrati di dimensione N i quali hanno tutti elementi nulli ad eccezione del blocco j-esimo il quale é la matrice identitá di ordine N se all'istante N il ruolo di radice viene assegnato al nodo j-esimo.

7.2.1 Ipotesi sull'energia dissipata ed avvio alla formulazione risolutiva.

Per riuscire a proseguire nella trattazione seguente si ipotizza che l'energia spesa dalla rete al passo k-esimo sia scrivibile a meno di

una costante moltiplicativa come:

$$\begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} + \alpha * \left(M * V(k) * \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \right)$$

dove α é la costante introdotta in sezione 6.2 .Tale formulazione dell'energia spesa al passo k-esimo nasconde due approssimazioni: viene ipotizzato che il costo relativo alle operazioni che la radice compie sui messaggi ricevuti sia lo stesso del costo delle operazioni di fusione dei messaggi compiute dai nodi non radice, inoltre si compie una seconda approssimazione considerando come vettore dei messaggi inviati un vettore con tutti elementi 1 mentre si sa che solamente $(N-1)$ messaggi vengono inviati al generico istante k e quindi si sarebbe dovuto considerare un vettore di elementi pari a 1 ad eccezione di un elemento che avrebbe dovuto essere 0.
Chiariti i punti appena introdotti e definito con Δ il vettore colonna costituito solo da elementi pari a 1 si nota che l'energia spesa dai vari nodi a partire dall'istante iniziale fino all'istante T si puó scrivere come:

$$\sum_{k=0}^{T} \{\Delta + \alpha * (M * V(k) * \Delta + \Delta)\} =$$

$$= \sum_{k=0}^{T} \{\alpha * M * V(k) * \Delta + (\alpha + 1) * \Delta\} =$$

$$= \alpha * M * \sum_{k=0}^{T} \{V(k)\} * \Delta + (\alpha+1) * (T+1) * \Delta =$$

$$= \lambda * \Delta$$

dove λ é una qualche costante. Notiamo subito che $\lambda > (\alpha + 1) * (T + 1)$. Fino a questo punto si é semplicemente riscritto l'energia spesa fino all'istante T e la si é posta uguale ad un vettore con le componenti di ugual valore. Ribadiamo che si vogliono all'istante finale tutti i nodi con la stessa energia spesa a partire dall'istante iniziale di evoluzione della rete. Notiamo che dall'ultima uguaglianza scritta discende la seguente equazione:

$$M * \sum_{k=0}^{T} \{V(k)\} * \Delta = \frac{\lambda - (\alpha+1) * (T+1)}{\alpha} * \Delta$$

Tale equazione puó essere riscritta in forma diversa se si nota che :

$$\sum_{k=0}^{T} \{V(k)\} * \Delta = \begin{bmatrix} \Delta * n_1 \\ \vdots \\ \Delta * n_N \end{bmatrix}$$

dove n_j rappresenta il numero di volte che viena attribuita la carica di radice al nodo j-esimo. Notiamo che il vettore a secondo membro ha N^2 righe ma sopratutto riportiamo la seguente relazione tra T ed i vari n_j:

$$\sum_{j=1}^{N} [n_j] = T + 1$$

Dopo quanto detto si puó riscrivere l'equazione lascita in sospeso come:

$$M * \begin{bmatrix} \Delta * n_1 * \frac{\alpha}{\lambda - (\alpha+1) * (T+1)} \\ \vdots \\ \Delta * n_N * \frac{\alpha}{\lambda - (\alpha+1) * (T+1)} \end{bmatrix} = \Delta$$

Introduciamo ora il cambiamento di variabili seguente:

$$\widetilde{n_j} = n_j * \frac{\alpha}{\lambda - (\alpha+1) * (T+1)} = n_j * \beta$$

Grazie a questo cambiamento di variabili é abbastanza facile riuscire a capire che la equazione in questione si puó riscrivere ancora una volta come:

$$Q * \begin{bmatrix} \widetilde{n}_1 \\ \widetilde{n}_2 \\ \vdots \\ \widetilde{n}_N \end{bmatrix} = \Delta$$

dove si é posto:

$$Q = [M_a(X_r = x_1) * \Delta | \ldots | M_a(X_r = x_N) * \Delta]$$

L'elemento q_{ij} della matrice Q rappresenta il numero di figli nel nodo i-esimo nella configurazione della rete con nodo radice j-esimo. Ragionando su questo fatto si puó affermare che le colonne della matrice Q sono linearmente indipendenti e quindi Q risulta invertibile.

Il sistema lineare appena scritto ammette quindi una unica soluzione che in generale puó avere componenti di segno qualsiasi. Notiamo peró che vale la seguente implicazione:

$$(n_j \geq 0 \, \forall \, j) \wedge (\lambda > (\alpha{+}1) * (T{+}1)) \Rightarrow (\widetilde{n_j} \geq 0 \, \forall \, j)$$

Quindi per il conseguimento del nostro scopo non é sufficente trovare la soluzione $\widetilde{n} = Q^{-1} * \Delta$ in quanto tale soluzione puó presentare componenti di valore negativo bensí si vuole trovare il vettore a componenti positive che in un certo senso approssima meglio la soluzione del sistema lineare. Si vuole quindi trovare il vettore $\overline{n}$ tale che la norma di $(\Delta - Q * \overline{n})$ sia minima. Attraverso l'utilizzo della norma due si puó quindi scrivere:

$$\begin{aligned}
\overline{n} = \quad & \underset{n \geq 0}{argmin} \, \|\Delta - Q * n\|_2^2 \\
= \quad & \underset{n \geq 0}{argmin} \left\| Q * (\underbrace{Q^{-1} * \Delta - \overline{n}}_{C}) \right\|_2^2 \\
= \quad & \underset{C \leq Q^{-1} * \Delta}{-argmin} \, \|Q * C\|_2^2 + Q^{-1} * \Delta \\
= \quad & \underset{C \leq Q^{-1} * \Delta}{-argmin} \, C^T Q^T Q C + Q^{-1}\Delta \\
= \quad & Q^{-1}\Delta - \overline{C}
\end{aligned}$$

Si riconosce subito che trovare $\overline{C}$ é un problema di programmazione quadratica che puó essere risolto attraverso una moltitudine notevole di metodi. In sede di simulazione non é stato sviluppato un algoritmo di risoluzione dei problemi di programmazione quadratica ma ci si é appoggiati a MATLAB il quale per risolvere questa tipologia di problemi fornisce la funzione "quadprog()".
Dopo aver trovato il valore di $\overline{n}$ si puó ricavare facilmente il valore di β. Infatti:

$$T + 1 = \sum_{j=1}^{N} [n_j] = \sum_{j=1}^{N} [\tfrac{\overline{n}_j}{\beta}]$$

$$\Rightarrow \beta = \tfrac{\Delta^T * \overline{n}}{T+1}$$

Grazie al risultato dell'ultimo calcolo si puó trovare finalmente:

$$n = \tfrac{\overline{n}}{\beta}$$

Notiamo che il vettore n é a componenti positive peró non é detto che sia a componenti naturali. Ricordiamo che la componente j-esima del vettore appena trovato rappresenta il numero di istanti in cui il nodo j-esimo deve permanere nello stato di radice. Notiamo che é non influente ai fini che ci siamo posti l'ordine nel quale si mettono come radice i vari nodi. Da quanto detto risulta doveroso approssimare il vettore a componenti reali appena trovato con un vettore a componenti positive ed intere. Questa approssimazione é stata realizzata direttamente attraverso un casting delle componenti del vettore n, quindi si é sostituito ogni elemento del vettore n con il naturale piú vicino ad esso. L'errore che si commette con tale metodo é in generale piccolo in quanto le componenti di n che si trovano sono di solito tutte molto maggiori di 1 e quindi la loro parte decimale é trascurabile rispetto alla loro parte intera.

7.3 Tutorial, commenti dei risultati e modifiche.

Per fare girare la simulazione basta mettere come **Current Directory** in **MATLAB** la cartella **SimulazioneValoreFinaleMinimo**. Aperta tale directory si deve avviare il file **EvoluzioneValoreFinale.m**. La simulazione é stata eseguita sul grafo di FIGURA 10 ed il suo risultato viene rappresentato in una schermata con tre grafici.

7.3.1 Evoluzione del funzionale $J(\bullet)$.

Il funzionale $J(\bullet)$ considerato in tale simulazione é quello introdotto in sezione 6.3 . Si va quindi a valutare nei vari istanti di tempo la norma dello scostamento dalla loro media dei vari livelli energetici dei nodi della rete.
Si nota che il grafico riportato in FIGURA 20 ha

Figura 20. Evoluzione del funzionale $J(\bullet)$.

andamenti tipici giá incontrati in precedenza. Un fatto molto importante é che il funzionale assume valore quasi nullo in corrispondenza all'istante finale di simulazione.

7.3.2 Rappresentazione 3-D del livello energetico dei nodi.

Tra i vari risultati della simulazione viene riportato un grafico 3-D rappresentante lénergia residua di ogni nodo nei vari istanti di tempo. Sui tre assi vengono riportati rispettivamente i possibili valori dell'energia residua dei nodi, il numero identificativo del generico nodo e sull'ultimo asse vengono invece riportati i vari istanti temporali. In FIGURA 21 si vede come all'istante finale della simulazione tutti i nodi siano completamente scarichi, ció significa che nella particolare rete presa in esame per la simulazione la soluzione $Q^{-1} * \Delta$ é a componenti tutte positive.

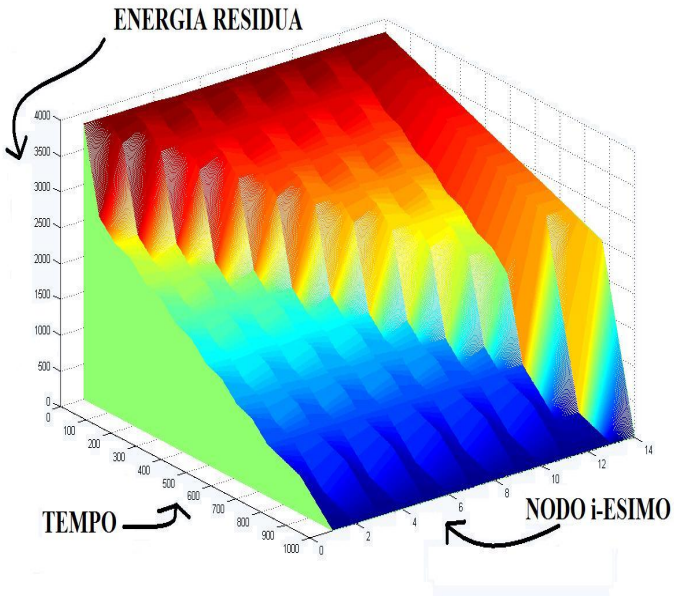

Figura 21. Rappresentazione 3-D del livello energetico dei nodi.

7.3.3 Rappresentazione Real-Time del livello energetico dei nodi.

Nella terza finestra dei risultati della simulazione si hanno invece degli istogrammi la cui altezza diminuisce nel tempo. Il generico istogramma j-esimo rappresenta il livello della batteria del nodo j-esimo, tale livello all'aumentare del tempo di simulazione cala piú o meno velocemente se il nodo si trova nello stato di radice, foglia o nodo intermedio.

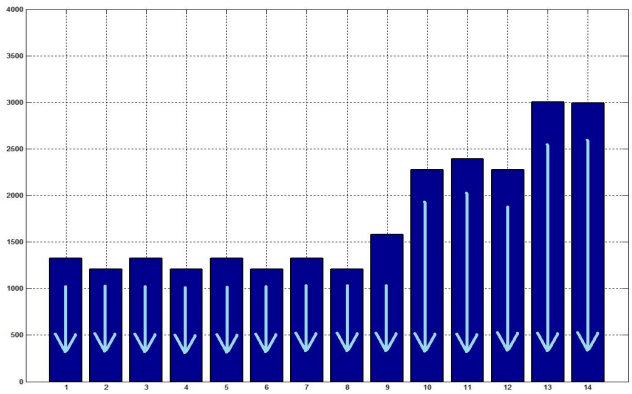

Figura 22. Rappresentazione Real-Time del livello energetico dei nodi.

8 SIMULAZIONE RIASSUNTIVA DEI VARI METODI DI CONTROLLO.

8.1 Introduzione alla simulazione.

Nella simulazione riassuntiva vengono simulate le varie tecniche di controllo introdotte in precedenza. Viene data la possibilitá all'utente di introdurre la rete sulla quale simulare il programma in questione tramite una interfaccia introduttiva. Anticipiamo che il risultato della simulazione é una evoluzione real-time della energia spesa dai vari nodi al variare delle diverse tecniche di controllo che vengono utilizzate.

8.2 Avvio della simulazione.

Per fare girare la simulazione basta mettere come **Current Directory** in **MATLAB** la cartella **SimulazioneRiassuntiva**. Aperta tale directory si deve avviare il file **Simulazione.m**. Sul workspace comparirá una breve introduzione alle modalitá di simulazione. Una volta finito di leggere tale introduzione cliccare con il mouse sulla "Command Window" di MATLAB e premere un tasto qualunque della tastiera. Si aprirá la finestra di introduzione della rete.

8.3 Modalitá di introduzione della rete.

All'apertura della finestra di introduzione della rete si puó iniziare ad introdurre il grafo rappresentativo della rete. La finestra che si apre presenta un grafico cartesiano vuoto nel quale si introdurranno i vari nodi ed archi del grafo. Come prima cosa vengono introdotti i nodi del grafo. Per introdurre un nodo é sufficente cliccare una sola volta sul grafico cartesiano con il pulsante sinistro del mouse. Per introdurre l'ultimo nodo e quindi terminare la procedura di inserimento dei nodi stessi é sufficente fare un unico clic del mouse con il pulsante destro. Terminata la procedura di inserimento dei nodi inizia l'inserimento degli archi. Per inserire un arco che collega i nodi A e B si deve cliccare in vicinanza del nodo A (o del nodo B) con il pulsante sinistro del mouse e poi cliccare con lo stesso pulsante sinistro del mouse in vicinanza del nodo B (o del nodo A). L'inserimento dell'ultimo arco, che quindi porta a termine la procedura di definizione del grafo, avviene cliccando con il pulsante sinistro del mouse in vicinanza di un nodo e poi cliccando con il pulsante DESTRO del mouse in vicinanza di un altro nodo.

8.4 Commento della rappresentazione Real-Time.

Terminata la procedura di inserimento del grafo la finestra si chiude e viene riportata sulla "Command Window" per alcuni secondi la matrice M_g cosí ottenuta. Trascorso questo breve arco di tempo nel quale viene riportata la matrice M_g compariranno delle scritte che elencano le varie metodologie di controllo che verranno simulate. Per avviare la simulazione cliccare con il pulsante sinistro del mouse sulla Command Window e premere un tasto qualsiasi della tastiera. Si aprirá un istogramma 3-D nel quale sui tre assi viene riportato il livello della energia spesa dal singolo nodo a partire dall'istante iniziale (quindi il valore della componente del vettore V_T corrispondente ad ogni nodo), i numeri identificativi dei vari nodi e sull'ultimo asse vengono riportati i numeri identificativi della tecnica di controllo che viene utilizzata. Tali numeri vengono definiti nelle ultime scritture che compaiono sulla Command Window prima della apertura della ultima finestra. Sul lato destro dell'istogramma si trovano dei grafici cartesiani che rappresentano i nodi della rete introdotti in precedenza in funzione di quale tecnica di controllo venga adottata. In tali grafici il nodo radice viene rappresentato con il colore verde e durante la simulazione si puó notare come la radice venga continuamente spostata attraverso la rete.

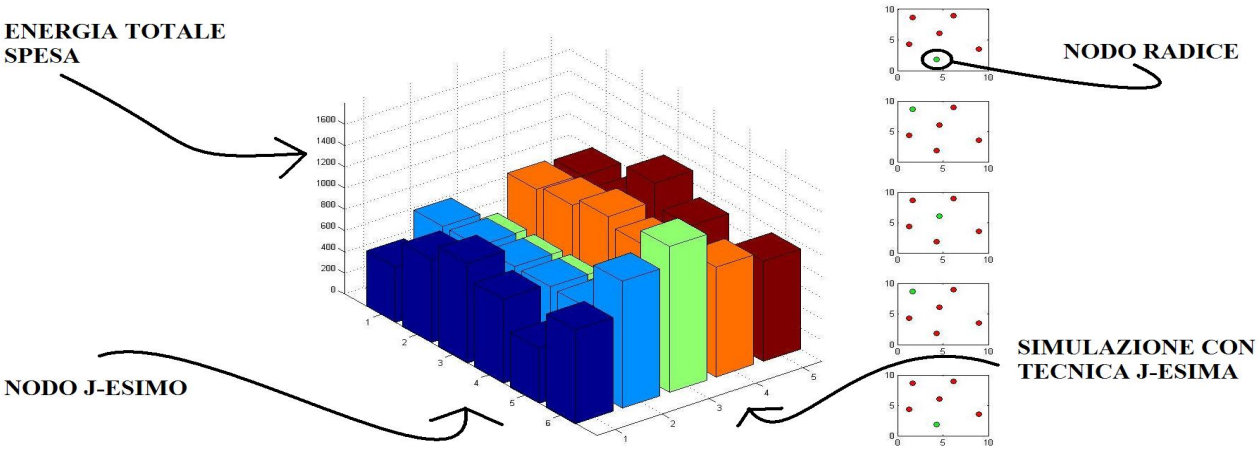

Figura 23. Schermata della simulazione finale.

8.5 Commenti su due esempi di reti.

Si consiglia di simulare un grafo dove ogni nodo comunica con tutti gli altri. Un esempio di tale grafo é quello in FIGURA 24.

Si puó facilmente capire che la tecnica di

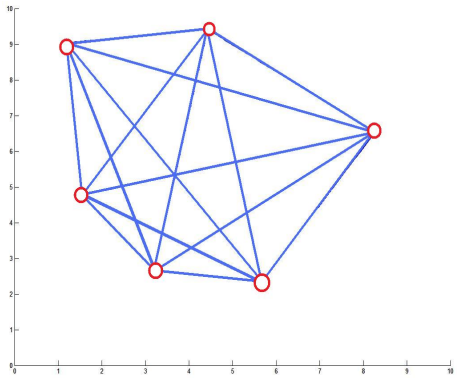

Figura 24. Rete con ogni nodo che comunica con qualsiasi altro nodo.

controllo migliore sará un tecnica che nel complesso cerca di far diventare ogni nodo radice un numero N di volte costante per tutti i nodi. Notiamo che non é necessario cambiare ogni volta il nodo radice ma serve solamente che nel complesso ogni nodo diventi radice un numero N di volte. Dai risultati della simulazione si vede che tale traguardo viene raggiunto da tutte le tecniche da noi introdotte e quindi tale fatto é indice della bontá di queste tecniche.

Notiamo subito che ció che viene rappresentato nell'istogramma non é direttamente l'energia spesa ma invece é il valore delle varie componenti di V_T. Dato che l'energia spesa é direttamente proporzionale a V_T attraverso una costante moltiplicativa i grafici dell'istogramma 3-D che possono essere messi al confronto sono solo quelli relativi alle tecniche di controllo con uguale definizione di V_T. Per avere un confronto esatto dei fari metodi bisogna ricavare empiricamente le varie costanti di proporzionalitá che legano V_T alla energia spesa. Si consiglia inoltre di simulare la rete di FIGURA 25. In tale esempio si cade nel caso della tecnica ValoreFinaleMinimo con $Q^{-1}\Delta$ a componenti negative.

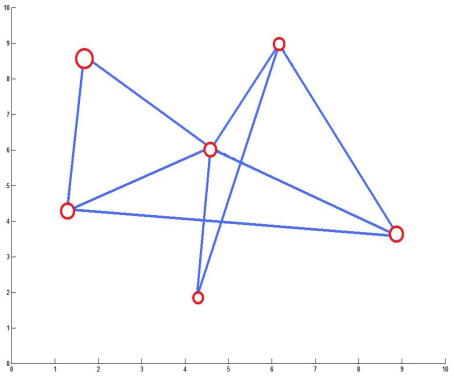

Figura 25. Rete che non permette un consumo di energia equo tra i vari nodi.

9 Leader Election and Sub-Leader Election

9.1 Introduzione

In questa parte andremo ad analizzare sempre all'interno del problema di equidistribuzione dei consumi metodi e algoritmi per una leader election robusta in una rete, prendendo in analisi dapprima casi particolari di reti (anelli) e poi reti piu' generiche.
Con robustezza intenderemo esclusivamente il fatto che nel caso di guasto del nodo leader non vi sia perdita di informazione ne' particolari rallentamenti al processo di raccolta dati e alla loro comunicazione all'esterno, insomma che non si compromettano le funzioni della rete (ignorando pero' le ripercussioni della rottura del leader per quando riguarda l'aspetto morfologico e di connessione della rete).

Al problema appena posto si provera' a dare una risposta andando ad analizzare metodi che eleggano sì un leader ma anche un sottoleader, in modo tale che quest'ultimo quando si verifichi un guasto al primo ne possa fare le veci e sostituendosi alle sue mansioni.

Procedere in questo modo nel caso di rottura e' senz'altro piu' veloce e meno dispendioso per la rete rispetto al fatto di dover riavviare le procedure per una nuova individuazione di un leader.

Questa soluzione non e' pero' sicura al cento per cento in quanto si potrebbe verificare il problema che si possano guastare conteporanemante ambedue (sia leader che sottoleader), chiaramente questa e' una situazione particolare che non dovrebbe capitare molto di sovente; e' insomma una soluzione parziale che non risolve totalmente il problema anche se lo limita molto; una soluzione per arginare ulteriormente il problema potrebbe essere quella di trovare anche un sottosottoleader per andare a compersare il caso sfortunato di rottura contemporanea degli altri due (si riproporrebbe pero' il problema nrl caso di guasto/rottura contemporanea di 3 nodi contemporaneamente).

Gli algoritmi di leader election noti e descritti in letteratura effettuano un'unica elezione basandosi su valori dei singoli nodi costanti per tutta la vita del sistema e la scelta del leader ricade sul nodo che ha questo valore migliore degli altri nodi presenti sulla rete.
Con valore migliore si può intendere che sia esso il minimo o il massimo a seconda del tipo di algoritmo e a seconda dell'interesse, adesempio: nel caso di una rete di calcolatori si può pensare di eleggere come leader il calcolatore con cpu massima o in una rete di sensori quello con capacita' oppure livello iniziale di batteria massimo o quello con la percentuale minore di batteria consumata. E'comunque relativamente facile riuscire ricondursi da un problema di massimo ad un problema di minimo, o viceversa; non e' nemmeno un problema ricondursi da un problema di 'vicinanza' ad un problema di minor distanza da un determinato valore

Nei casi di nostro interesse (cioe' quelli di equidistribuzione dei consumi delle batterie dei nodi di una rete) e' necessario pero' che il leader debba girare tra i nodi e che non sia sempre lo stesso, il valore/la grandezza in base al quale si effettuera' la scelta non dovra' essere costante, ma sara' un valore che variera' nel tempo: potra' ad esempio rappresentare il livello di batteria o il carico di cpu di un singolo nodo, oppure un funzionale calcolato in base ai valori degli x nodi piu' vicini al nodo preso in esame (anche con $x = n$ cioe' con tutti i nodi della rete).

Dato che l'agente leader deve soddisfare alcune richieste particolari (raccolta, elaborazione ed invio all'esterno dei dati)

si puo' pensare in un primo approccio di eleggere il nodo che abbia la batteria piu' carica (stia utilizzando meno la propria Cpu) e questo dato e' per natura variabile nel tempo. Si vuole che il leader della rete venga eletto in modo dinamico, non si vuole che il leader sia sempre lo stesso agente, bensi' che sia in continua evoluzione, questo in concreto sia per far fronte a possibili attacchi informatici, sia ad eventuali danni fisici del sistema (si pensi ad esempio ad un caso di una rete di videoagenti di sorveglianza che possono essere instalati anche alleterno).
E' estremamente importante che la il passaggio da leader election statica a leader election dinamica.
Poi si puo' notare come affidarsi al singolo valore di carica della batteria del nodo da designare leader puo' essere riduttivo e possa essere piu' produttivo valutare un funzionale che dipenda anche sui nodi vicini poiche' anche questi saranno soggetti ad un lavoro maggiore rispetto agli altri nodi poiche' attraverso questi passera' un maggior quantitativo di messaggi(informazioni).

9.2 Definizioni preliminari

Andremo ora a dare delle definizioni di base per poi poter illustrare alcuni algoritmi di Leader Election classici (e statici) per eleggere un leader in base al singolo valore migliore che saranno successivamente modificati per lavorare su funzionali che dipendono da valori di piu' nodi e per individuare leader e sottoleader.

Definizione 1 (UID): Sia C un anello di n nodi e $V = 1, 2, ..., n$ l'insieme dei nodi, ad ogni nodo $m \in V$ e' associato un valore $c = kn + m$, dove k identifica il k-esimo stato di ogni nodo di V . Il valore $c \in N$ e' un identificatore univoco per ogni nodo (UID), tale che $mod_n c = m$.

Definizione 2 (Processo): Viene definito come processo l'evolversi dello stato di uno stesso nodo.

Viene invece chiamato transizione il passaggio di informazione da un nodo ad un altro. In un anello le transizioni che coinvolgono trasmettitori diversi possono essere contemporanee, quindi dando una definizione formale

Definizione 3 (Transizione): per transizione in un anello si intende l'invio e la relativa ricezione di un messaggio da parte di ogni nodo al suo successivo ;
in una rete connessa viene detta transizione il trasferimento di informazione da un nodo verso tutti i suoi vicini (nodi direttamente connessi)

Si noti che nel secondo caso (rete connessa) le transizioni di ogni nodo avvengono contemporaneamente, sfruttando canali full-duplex, quindi per transizione si considera linvio di messaggi da parte di ogni nodo verso tutti i suoi vicini.

9.3 Considerazioni preliminari

Ci si propone di andare a descrivere algoritmi di leader election gia' esistenti e descritti in letteratura per reti ad anello con comunicazione di tipo uni- o bi-direzionale,o per reti generiche. L'elezione per tutti gli algoritmi in letteratura va ad individuare il nodo con uid migliore (dove cose si intenda per migliore e' stato definito in precedenza). Inoltre il numero n di nodi della rete puo' essere noto o meno; si noti che nel caso in cui si voglia disporre della conoscenza del numero dei nodi della rete, o un algoritmo la richieda per il suo funzionamento basta introdurre una fase preliminare atta a questo compito.

Teorema 4: Se tutti i processi sono indistinguibili il problema non ha soluzione (anche se l'anello e' bidirezionale e n e' noto).
Dimostrazione
Sia A un sistema, senza perdita di generalita' si puo' assumere che ogni processo di A abbia un unico stato iniziale, se per assurdo tutti i processi sono nello stesso stato, dopo un numero r di transizioni eseguite, essi saranno ancora in uno stato identico, allora se un processo ha raggiunto lo stato di leader ogni altro processo lo ha raggiunto (in contraddizione con l'unicita' del leader).cvd

Nel caso tutti i processi processi fossero identici se li si etichettano con l'UID non

si verificano problemi, Esitono due tipi di approcci per risolvere il problema: quello degli algoritmi comparativi che individuano il leader confrontando gli UID, e quello degli algoritmi non comparativi che sfruttano diverse velocita' di trasmissione dei messaggi. Andremo ora ad analizzare gli algoritmi esistenti per poi generalizzarli/modificarli, uno ad uno per fare in modo che facciano leader e sottoleader election e che la scelta del leader e del sottoleader non cada sul nodo con l'UID migliore ma su quello che ha il valore di un funzionale che dipende dal valore del proprio UID e anche da quello dei propri vicini.

9.4 Ipotesi di lavoro

Per ora si prenderanno come ipotesi di lavoro le seguenti:

- Comunicazione sicura (cioe' che i messaggi arrivino sempre)
- Rete sincrona (tutti i nodi siano sincronizzati e che i messaggi partano e arrivino contemporaneamente in tutti i nodi)
- Senza Guasti (che durante la fase di Leader Election non vi siano guasti)
- Si procedera' alla leader election in base agli UID dei nodi all'istante iniziale dell'elezione, cio' ovviamente potrebbe non dare risultati perfetti perche' anche durante l'elezione questi valori possono modificarsi, ma si assume qui come ipotesi e d'ora in poi che durante i processo di elezione gli UID rimangano costanti (semplificazione che va bene nel caso il tempo di cambiamento degli UID sia piu' lungo rispetto al tempo di elezione)

9.5 Parametri di interesse

Si prenderanno come parametri di interesse per i vari algoritmi:

- Il tempo totale dell'algoritmo
- Il numero di messaggi totale inviati
- La necessitá o meno di conoscere il numero totale di nodi (il costo che comporta questa conoscenza in termini di tempo e di messaggi)
- Se ci sia informazione completa, cioe' se al termine dell'algoritmo ogni nodo conosca il leader (e il sottoleader)

9.6 Tipi di funzionali

Possono essere usati diversi funzionali applicati ai nodi per confrontarne la bonta' come leader:

- Funzionale dipendente solo dallo stato del singolo nodo, cioe' lo stato di quel nodo, il funzionale in pratica e' l'UID
- Funzionale dipendente dai valori di x nodi vicini
- Funzionale dipendente dagli stati di tutti i nodi
  (qui si pone il problema che ogni nodo avendo disponibile l'informazione di tutti gli altri puo' calcolareautonomamente anche i funzionali degli altri, quindi non servirebbe nessun'altra comunicazione visto che ogni nodo disponendo della conoscenza di tutti i funzionali puo' valutare quali siano il funzionale migliore e il secondo migliore, tutti i nodi ovviamente giungerebbero alla stessa conclusione e non si necessiterebbe di alcun altra comunicazione, altrimenti si potrebbe procedere facendo in modo che solo un nodo (o in caso piu' d'uno per velocizzare i tempi successivi di comunicazione) calcoli i funzionali di tutti, li confronti e poi diffonda i risultati dell'elezione agli altri, in quest'ultimo caso il tempo sarebbe maggiore di quello precedente, aumenterebbe anche il numero di messaggi mentre i conti di calcolo dei funzionali sarebbero molto minori)'

9.7 Algoritmi Comparativi

Verranno illustrati ora algoritmi comparativi classici per la leader election e loro possibili modifiche e generalizzazioni per la leader & subleader election e l'utilizzo per i confronti di funzionali dipendenti anche dagli stati dei nodi vicini.

9.7.1 Algoritmo LCR [anello unidirezionale]

L'algoritmo LCR, il cui nome deriva da coloro che l'hanno sviluppato e proposto cioe' Lann,

Chang e Roberts, e' un algoritmo di tipo comparativo per un anello unidirezionale di taglia sconusciuta, sul quale siano definiti gli UID per ogni nodo.
Ora una breve spiegazione del suo funzionamento e dimostrazione del funzionamento della sua versione classica, prima di addentrarsi nelle versioni modificate.

Ogni processo trasmette il proprio UID nell'anello, ogni processo che riceve un UID lo confronta con il proprio:

- se l'UID ricevuto e' maggiore del proprio, il processo mantiene il proprio UID e trasmette l'UID ricevuto,
- se l'UID ricevuto e' minore del proprio, il processo trasmette il proprio UID,
- se sono uguali il processo passa nello stato di leader.

Dato i_{max} il processo con il massimo UID (u_{max}), e' sufficente dimostrare i seguenti due lemmi:

Lemma 1: Il processo i_{max} e' il leader dopo n transizioni.
Dimostrazione
Il processo i_{max} ha valore di UID u_{max}, che e' il massimo di tutto lanello, e' allora sufficente dimostrare che:
Dopo n transizioni lo stato di $i_{max} = leader$.
Per r tale che $0 \leq r \leq n-1$, dopo r transizioni il processo $i_{max} + r$ trasmettera' u_{max}. si dimostra per induzione su r.
Per $r = 0$, i_{max} trasmette u_{max}, dato che dopo nessuna transizione ha ricevuto alcun messaggio in input.
Il passo induttivo si basa sul fatto che ogni nodo, eccetto i_{max}, accetta il massimo valore e poi lo ritrasmette.

Lemma 2: Nessun altro processo (oltre a i_{max}) e' nello stato di leader.
Dimostrazione
E' noto che tutti gli altri nodi sono sempre nello stato unknown. Se i e j sono due processi distinti dell'anello, si definisce $[i, j)$ l'insieme di indici $i, i+1, ..., j-1$ (dove le addizioni sono in modulo n). $[i, j)$ e' l'insieme di processi sucessivi ad i in senso orario fino a $j-1$. L'asserzione sucessiva dice che nessun UID v puo' raggiungere un processo tra i_{max} e il generatore di v.

Asserzione 1: Per ogni r e ogni i, j. Dopo r transizioni, se $i \neq i_{max}$ e $j \in [imax, i)$ allora j non trasmettera' u_i.
che e' verificato dal fatto che essendo u_max maggiore di tutti gli UID della rete, u_i non potra' mai attraversare il processo i_{max} e raggiungere i.

E' quindi dimostrato che l'algoritmo LCR risolve il problema di leader - election.

Analisi delle complessita'
Il numero di transizioni necessarie allalgoritmo e': $T(n) = \Theta(n)$
e il numero di messaggi prodotti dalla rete e': $M(n) = \Theta(n^2)$.

Terminata l'elezione il processo leader puo' inviare un messaggio di repeat sulla rete consentendo ai processi esitanti (quelli che non si sono dichiarati leader) di dichiararsi non-leader. Cio' raddoppia il tempo di esecuzione dell'algoritmo e aumenta il numero di messaggi di un addendo $\Theta(n)$, ma ogni processo puo' cosi' definire esattamente il suo stato, e se dispone di una cella di memoria atta allo scopo memorizzare quale sia il nodo leader.

9.7.2 Algoritmo LCR-smart [anello unidirezionale]

E' una variante dell'algoritmo LCR che permette una riduzione dei messaggi inviati, ma nessuna riduzione in termini di tempo, cio' avviene modificando il processo di confronto-trasmissione degli UID:
ogni processo trasmette il proprio UID nell'anello, ogni processo che riceve un UID lo confronta con il proprio:

- se l'UID ricevuto e' maggiore del proprio, il processo mantiene il proprio UID e trasmette l'UID ricevuto,
- se l'UID ricevuto e' minore del proprio, il processo mantiene il proprio UID e non trasmette alcun UID,

- se sono uguali il processo passa nello stato di leader. Il numero di messaggi effettivamente inviati dipende dalle condizioni della rete ma comunque rimane in genere $\Theta(n)$

9.7.3 Algoritmo LCR-smart2 [anello unidirezionale]

E' un'ulteriore variante dell'algoritmo LCR ed e' in vero una variante a LCR-smart che permette un'ulteriore riduzione dei messaggi inviati,sempre senza andare a modificare il tempo impiegato dall'algoritmo, cio' avviene con un'ulteriore modifica al processo di confronto-trasmissione degli UID, questa versione necessita che tutti inodi dispongano di una cella di memoria nella quale inizialmente sara' scritto il proprio UID:
ogni processo trasmette il proprio UID nell'anello, ogni processo che riceve un UID lo confronta con quello memorizzato:

- se l'UID ricevuto e' maggiore di quello memorizzato nellca cella di memoria del nodo del processo, il processo mantiene il proprio UID, mette nella cella di memoira l'UID ricevuto e lo trasmette,
- se l'UID ricevuto e' di quello memorizzato nellca cella di memoria del nodo del processo, il processo mantiene il proprio UID e non trasmette alcun UID,
- se sono uguali il processo passa nello stato di leader.

questo algoritmo permette un'ulteriore abbattimanto del numero dei messaggi che pero' dipende dalle condizioni dei nodi comunque in genere rimane $\Theta(n)$
, inoltre non ci sono costi aggiuntivi ne' in termini di tempo ne' in termini di messaggi inoltre al termine dell'algoritmo vi e' informazione completa poiche' tutti i nodi conoscano il leader,

9.7.4 Algoritmo 2LCR [anello unidirezionale]

Questo algoritmo permette di trovare sia il leader che il sottoleader, si basa sull'algoritmo LCR appena illustrato e consiste fondamentalmente in due passate di quest'ultimo; la prima individua il leader e la seconda il sottoleader.
La dimostrazione della correttezza del funzionamento e' banale e discende da quella dell'algoritmo LCR, sostanzialmente sia applica due volte quest'ultimo nella seconda volta pero' si fa tacere il nodo leader individuato nella passata precedente (oppure puo' semplicemente fare da eco ritrasmettendo i messaggi che riceve).

Analisi delle complessita'
Il numero di transizioni necessarie all'algoritmo e' il doppio di quelle dell'algoritmo LCR: $T(n) = \Theta(2n)$
e il numero di messaggi prodotti dalla rete e' anch'essoe' il doppio rispetto a quelli inviati dall'algoritmo LCR classico: $M(n) = \Theta(2n^2)$.
Terminata l'elezione il processo leader e quello sottoleader possono inviare un messaggio di repeat sulla rete consentendo ai processi esitanti (quelli che non si sono dichiarati ne' leader ne' sottoleader) di dichiararsi non-leader. Cio' aumenta il tempo di esecuzione dell'algoritmo si somma un tempo $\Theta(n)$ e aumenta il numero di messaggi di un addendo $\Theta(n)$, ma ogni processo puo' cosi' definire esattamente il suo stato e, se possiede due celle di memoria, puo' memorizzare quali siano il nodo leader ed il nodo sottoleader.
Si puo' ovviamente derivare quest'algoritmo usando LCR-smart o LCR-smart2 risucendo il numero di messaggi necessario alle elezioni.
Ad un'analisi attenta sorge pero' un problema, le due passate dell'algoritmo LCR avvengono in momenti diversi quindi si rischia che quello trovato con la prima passata al termine della seconda passata abbia ormai mutato il suo carico e a ben guardare al termine della seconda passata non sia piu' effettivamente il nodo che meriterebbe lo stato di leader, si lavora quindi con dati incoerenti perche' provenienti da due istanti temporali diversi, se invece nella seconda passata di utilizzano gli UID utilizzati nella prima passata si mantiene la coerenza temporale, ma si vanno ad utilizzare dati meno aggiornati di quelli che sarebbero disponibili

9.7.5 Algoritmi 2LCR-smart [anello unidirezionale]

Quest'algoritmo (2LCR-smart) ha come prerequisito che i nodi abbiano due celle di memo-

ria e sfruttando queste si riescono ad ottenere prestazioni migliori dell'algoritmo 2LCR in termini di tempo (che in termini di numero di messaggi). Entrando un minimo in dettaglio la prima cella di memoria di ogni nodo e' inizializzata al valore di UID del nodo dell'istante in cui inizia la leader election mentre la seconda e' inizializzata a zero, come procede l'algoritmo: ogni processo trasmette il proprio UID nell'anello; ogni processo che riceve un UID lo confronta con il proprio:

- se l'UID ricevuto e' maggiore di quello memorizzato nella prima cella di memoria del nodo del processo, il processo mantiene il proprio UID, mette nella seconda cella di memoria l'UID memorizzato nella prima, memorizza quello ricevuto nella prima cella e lo trasmette,
- se l'UID ricevuto e' minore di quello memorizzato nella prima cella di memoria del nodo del processo ma maggiore di quello memorizzato nella seconda cella, il processo mantiene il proprio UID, mette l'UID ricevuto nella seconda cella di memoria e lo trasmette,
- se l'UID ricevuto e' minore di quello memorizzato nella seconda cella di memoria del nodo del processo, il processo mantiene il proprio UID e non trasmette alcun UID,
- se l'UID ricevuto e quello contenuto nella prima cella sono uguali il processo passa nello stato di leader,
- se l'UID ricevuto e quello contenuto nella seconda cella sono uguali il processo passa nello stato di sottoleader.

In pratica si basa sempre sull'algoritmo LCR, consta di un'unica passata, ogni nodo ha due celle di memoria che memorizzano il massimo e il sottomassimo dei valori che sono già passati per il nodo. Ad ogni valore che arriva ad un nodo questo lo confronta dapprima con quello che ha memorizzato come massimo e se è maggiore questo diventa il nuovo massimo e il vecchio massimo diventa sottomassimo, altrimenti si verifica se è maggiore del sottomassimo, in caso positivo il valore arrivato diventa il nuovo sottomassimo altrimenti la situazione rimane immutata. Si noti come quest'algoritmo fonrisca gia' di per se' l'infromazione completa a tutti i nodi. E' inoltre importante sottolineare come leader e sottoleader vengano individuati su dati presi in uno stesso momento evitando cosi' di creare bisticci temporali.

Analisi delle complessita'
Il numero di transizioni necessarie all'algoritmo e': $T(n) = \Theta(n)$
e il numero di messaggi prodotti dalla rete e' nel caso peggiore $M(n) = \Theta(n^2)$ ma normalmente inferiore a questo upper-bound. Si noti che anche in questo caso vi e' informazione completa da parte di tutti i nodi e come per tutti gli algoritmi sviluppati a partire dall'algoritmo LCR non e' necessario conoscere la taglia dell'anello.

9.7.6 Algoritmi LCR e derivati con funzionali dipendenti da piu' nodi [anello unidirezionale]

Nel caso si vogliano utilizzare dei funzionali dipendenti da piu' nodi per confrontarli si dovra' modificare la natura degli UID, l'UID rimarra' sempre un valore $c = kn + m$ dove n e' il numero totale dei nodi e m il numero del nodo in considerazione mentre k diverra' il valore del funzionale.
In questo caso si dovra' introdurre una fase preliminare, prima dell'utilizzo degli algoritmi di leader election illustrati precedenza; nella quale fase verranno calcolati i funzionali e dopo che si sara' proceduto alle transizioni necessarie al calcolo di questi.
Nel caso che stiamo analizzando cioe' di anelli monodirezionali i messaggi che possono viaggiare solo in una direzione, se il funzionale del nodo m dipende ad esempio anche dallo stato del nodo $m + 1$, verso il quale puo' soltanto inviare messaggi e non riceverne direttamente, affinche' il nodo m riceva lo stato del nodo $m + 1$ questo dovrebbe girare per tutto l'anello; con un piccolo accorgimento si puo' risolvere questo problema: l'UID del nodo m non verra' calcolato dal nodo m stesso bensì dal nodo $m + x_{dx}$ che e' l'ultimo nodo dal quale dipende il funzionale del nodo m.
Gli algoritmi non subiscono problemi al funzionamento anche se i nodi al primo passo non trasmettono il proprio UID ma quello di

un altro nodo, l'importante e' che si compia un intero giro di confronti cioe' se il nodo $m + x_{dx}$ parte con l'UID del nodo m l'elezione sara' conclusa dopo n transizioni e se m e' il leader il suo UID ritornera' al nodo $m + x_{dx}$; questa fase introduttiva avra' durata $\Theta(x)$ dove x e' il numero di nodi dal quale dipende il funzionale di ogni nodo.

9.7.7 Algoritmi comparativi per funzionali dipendenti da tutti i nodi

Come gia' esposto in precedenza, in questo caso particolare dopo la fase in cui si trasmettono gli stati dei nodi ogni nodo puo' calcolare i funzionali e quindi gli UID di tutti gli altri nodi quindi non occorrono ulteriori trasmissioni perl'individuazione del leader e del sottoleader.

9.7.8 Rottura nodi

Il risparmio di messaggi nei vari algoritmi puo' pero' essere negativo al fine di controllare che ogni nodo continui a funzionare, non permette che ci si accorga immediatamente della rottura di un nodo come nel caso in cui non arrivi un messaggio da parte di questo al nodo successivo.

9.7.9 Algoritmo HS [anello bidirezionale]

Si illustra ora un altro algoritmo di tipo comparativo proposto da Hirschberg Sinclair (per questo detto algoritmo HS) il quale lavora su un anello bidirezionale, nel quale cioe' i messaggi possono viaggiare in ambedue le direzioni. L'algoritmo procede per fasi, ad ogni fase i processi inviano dei messaggi qui chiamati token composti dalla tripla $(u_i, flag, h)$; dove u_i indica l'UID del nodo i, $flag$ identifica la direzione in cui sta viaggiando il token: se e' uguale a out si sta dirigendo dal processo che l'ha generato verso quello destinatario, se e' uguale a in sta tornando verso il processo generante. Ogni token intende viaggiare per una distanza $h = 2^l$, dove l e' un numero crescente inizializzato a uno che indica la fase in cui si trova il processo. Nella fase 1 ogni nodo inizializza il token che viene trasmesso in entrambe le direzioni, in ognuna delle quali effettua due transizioni. Ogni processo invia in ambo le direzioni il proprio token (con $flag = out$). Se entrambi i token compiono il giro dell'anello e arrivano al processo che li ha generati, allora il processo passa alla fase successiva, altrimenti il processo non partecipera' piu' all'elezione. I token una volta giunti al processo posto a distanza h da u_i, tornano indietro ponendo il loro $flag = in$ e senza effettuare altri confronti. Mentre il token u_i procede, ogni processo j (appartenente al cammino) lo confronta con il proprio UID:

- se $u_i < u_j$ allora j rimuove il token,
- se $u_i > u_j$ allora j rilascia il token,
- se $u_i = u_j$ allora j ha ricevuto il proprio token (ha percorso tutto lanello) e si dichiara leader.

Ogni processo che termina una fase raddoppia la distanza percorsa dai propri token (incrementando l), finche' uno riesce a percorrere tutto l'anello senza essere bloccato da un altro processo con UID maggiore, l'unico token che riesce a compiere cio' e' quello con il maggior UID e il processo che l'ha generato si dichiarera' leader.
Vi e' una semplice variante di quest'algoritmo che permette un riapsrmio di messaggi, nel caso i nodi dispongano di una cella di memoria, possono memorizzare l'u_{cellaj} maggiore che e' fino a quel momento passato attraverso di loro e poi confrontare quelli dei token ricevuti con quello e non con il proprio; inizialmente ogni nodo j inizializza $u_{cellaj} = u_j$ poi seguono le trasmissioni ed i confronti:

- se $u_i < u_{cellaj}$ allora j rimuove il token,
- se $u_i > u_{cellaj}$ allora j rilascia il token,
- se $u_i = u_{cellaj}$ allora j e $u_i \neq u_j$ rilascia il token,
- se $u_i = u_{cellaj}$ allora j e $u_i = u_j$ ha ricevuto il proprio token (ha percorso tutto lanello) e si dichiara leader.

Analisi delle complessita'
Il numero di transizioni impiegate per ogni fase eccetto l'ultima e' 2^{l+1}, l'ultima fase $\lceil logn - 1$ e' incompleta e impiega n transizioni, la complessita' dell'ultima fase e' maggiore o uguale alla complessita' di tutte le fasi precedenti, allora: $T(n) = 2 \cdot 2^{\lceil logn \rceil} = \Theta(n)$.

Il numero di messaggi generati da ogni processo alla fase 0 e' 2 (tali messaggi devono poi anche tornare al mittente), sono quindi $4ntoken$ per questa fase. Per $l > 0$, un processo invia il token di fase l se ha ricevuto entrambi i token di fase $l-1$, cio' avviene se il token non 'e stato rimosso da nessun processo che si trova a distanza $2l-1$. Questo implica che per ciascun gruppo di $2l-1+1$ processi consecutivi, al piu' un processo passera' alla fase sucessiva l. In tutto saranno $\left\lfloor \frac{n}{2^{l-1}+1} \right\rfloor$ i processi che passeranno alla fase successiva.
Allora il numero di messaggi prodotti in ogni fase e': $4\left(2^l \cdot \left\lfloor \frac{n}{2^{l-1}+1} \right\rfloor\right) \leq 8n$.
Questo perche' alla fase l i token viaggiano per una distanza $2l$ e il numero di fasi e' $\lceil logn \rceil + 1$ (compresa la fase 0), quindi il numero totale di messaggi e':
$M(n) = 8n(1 + \lceil logn \rceil) = O(n \lceil logn \rceil)$. Nel caso di cella di memoria vi e' informazione completa.

In letteratura e' noto che per gli algoritmi comparativi il limite inferiore per il numero di messaggi e' proprio:
$M(s) = \theta(n \lceil logn \rceil)$.

9.7.10 Algoritmo 2HS [anello bidirezionale]

Questo algoritmo permette di trovare sia il leader che il sottoleader, si basa sull'algoritmo HS appena illustrato e consiste fondamentalmente in due passate di quest'ultimo; la prima individua il leader e la seconda il sottoleader.
La dimostrazione della correttezza del funzionamento e' banale e discende da quella dell'algoritmo HS. In pratica si applica due volte l'algoritmo HS ma nella seconda volta pero' si fa tacere il nodo leader individuato nella passata precedente (oppure puo' semplicemente fare da eco ritrasmettendo i messaggi che riceve). Complessita' temporale e numero di messaggi saranno rispettivamente il doppio di quelle dell'algoritmo HS. $T(n) = \Theta(n)$
$M(n) = 2(n \lceil logn \rceil)$.

Anche in questo caso si ripongono in modo del tutto analogo le problematiche dell'algoritmo 2LCR sulle discrepanze temporali.

9.7.11 Algoritmo 2HS-smart [anello bidirezionale]

Anche per quanto riguarda una variazione intelligente e meno dispendiosa dell'algoritmo HS e' necessario che tutti i nodi abbiano due celle di memoria,ogni nodo inizializzera' la prima con il valore del proprio u_j e la seconda con zero. seguiranno poi le trasmissioni ed i confronti

- se $u_i > u_{cella1}$ allora j rilascia il token e pone $u_{cella2} = u_{cella1}$ e $u_{cella1} = u_i$,
- se $u_i = u_{cella1}$ allora j e $u_i \neq u_j$ rilascia il token,
- se $u_i = u_{cella1}$ allora j e $u_i = u_j$ ha ricevuto il proprio token (ha percorso tutto lanello) e si dichiara leader.
- se $u_i < u_{cella1} u_i < u_{cella2}$ allora j rimuove il token,
- se $u_i < u_{cella1} u_i > u_{cella2}$ allora j rilascia il token e pone $u_{cella2} = u_i$,
- se $u_i < u_{cella1} u_i = u_{cella2}$ e $u_i \neq u_j$ allora j rilascia il token,
- se $u_i < u_{cella1} u_i = u_{cella2}$ e $u_i = u_j$ allora j ha ricevuto il proprio token (ha percorso tutto lanello) e si dichiara sottoleader

La complessita'

9.7.12 Algoritmi HS e derivati con funzionali dipendenti da piu' nodi [anello bidirezionale]

Anche in questo caso basta introdurre una fase preliminare in modo che ad ogni nodo giungano i valori degli x nodi vicini che andranno ad influenzare il suo UID, gli algoritmi funzionano analogamente e senza problemi

9.7.13 Algoritmo Flood Max [rete connessa]

Sia $G = (V, E)$ un grafo connesso che rappresenta una rete generica, in cui V sia l'insieme dei nodi ed E l'insieme degli archi. Sia $diam(G) = d$. Si considera che ad ogni nodo sia associato un UID come abbiamo fatto in precedenza, l'agoritmo Flood Max si propone di eleggere come leader il nodo con maggior UID. Ogni processo mantiene in una cella di memoria il maggior UID che ha visto. Inizialmente mantiene il proprio, ad ogni transizione

ogni processo invia il tale record a tutti i suoi vicini connessi che lo confrontano con il proprio. Dopo d transizioni il processo che avra' il proprio UID uguale a quello presente sul proprio record sara' il leader.
Analisi delle complessita'
Il tempo impiegato per raggiungere lo stato di leader ad un unico nodo e' esattamente d, mentre per definire il numero di messaggi prodotti definiamo m il numero massimo di vicini connessi per ogni nodo e quindi il numero totale di messaggi sara' al piu' $m \cdot d$, quindi:
$T(n) = \Theta(d)$
$M(n) = \Theta(m \cdot d)$ Per diminuire il numero di messaggi si puo' fare in modo che ogni nodo invii l'UID contenuto della sua cella di memoria solo se diverso da quello che aveva precedentemente inviato, in questo caso il numero di messaggi dipende dalle condizioni della rete.

9.7.14 Algoritmo 2 Flood Max [rete connessa]

Consiste sostanzialmente in due passate dell'algoritmo Flood Max, nella seconda il nodo eletto nella prima tacera', anche in questo caso si ripropongono le problematiche degli algoritmi con due passate.
Analisi delle complessita'
Le complessita' saranno esattamente il doppio di quelle dell'algoritmo Flood Max
$T(n) = \Theta(2d)$
$M(n) = \Theta(2m \cdot d)$

9.7.15 Algoritmo 2 Flood Max smart [rete connessa]

Questo algoritmo anch'esso mutuato da Flood Max permette nel caso si possano trasmettere due UID in un unico messaggio e i nodi abbiano a disposizione due celle di memoria di riuscire a procedere alla leader e alla subleader election con un unica passata, quindi in tempo
$T(n) = \Theta(d)$
e numnero di messaggi:
$M(n) = \Theta(m \cdot d)$

9.7.16 Algoritmi Flood Max e derivati con funzionali dipendenti da piu' nodi [reti generiche]

Basta anche in questo caso introdurre una fase preliminare per permettere di raccogliere le informazioni e calcolare gli opportuni UID con al posto dello stato il funzionale e poi utilizzare l'algoritmo in modo normale

9.8 Algoritmi non comparativi

Verranno illustrati ora alcuni algoritmi non-comparativi classici per la leader election e loro possibili modifiche e generalizzazioni per la leader & subleader election con anche l'utilizzo di funzionali dipendenti anche dagli stati dei nodi vicini al posto degli stati dei singoli nodi. Gli algoritmi non comparativi hanno come scopo l'abbattimento del numero di messaggi utilizzati dall'algoritmo e non usano il confronto tra UID, ma tale valore identifica la velocita' con cui viaggiano tali messaggi, in tal modo si ottengono algoritmi che utlizzano meno la rete.

9.8.1 Algoritmo Time Slice [anello unidirezionale]

Dato un anello di taglia n nota con comunicazioni unidirezionali, si vuole eleggere come leader il nodo con il minor UID (quello cha ha consumato meno carica della batteria nel caso della volonta' di uniformare i consumi delle batterie). La commutazione procede in fasi, ogni fase consiste in n transizioni, ognuna delle quali e' dedicata alla possibile circolazione dei token contenenti gli UID attraverso lanello. Piu' precisamente in ogni fase v, che consiste nelle transizioni $(v-1)n+1, ..., v$, solo al token contenente l'UID di valore v e' consentito viaggiare. Se un processo i con UID v esiste e la transizione $(v-1)n+i$ e' raggiunta senza che i abbia ricevuto precedentemente un token non nullo, allora il processo i si elegge leader e trasmette il token con il suo UID attraverso lanello.
Il tempo dell'algoritmo dipende dall'UID con valore minore UID_{min} e sara' pari a $\Theta(UID_{min} + n)$ i messaggi che circoleranno nella rete saranno invece soltanto n

9.8.2 Algoritmo 2 Time Slice [anello unidirezionale]

Si basa sull'algoritmo Time Slice, consiste in due passate dello stesso, la prima atta ad individuare il leader la seconda il sottoleader. Nella

seconda il leader non individuato in quella precedente non partecipa (ma nel frattempo puo' esserci un controllo che verifichi se il leader si rompa prima ancora che si completi la seconda elezione. Il tempo dell'algoritmo dipende dagli UID con valori minori al quale si somma il tempo necessario per comunicare agli altri nodi $\Theta(UID_{min}) + \Theta(UID_{sotto-min}) + \Theta(2n)$ i messaggi che circoleranno nella rete saranno invece $\Theta(2n)$

9.8.3 Algoritmo 2 Time Slice-smart [anello unidirezionale]

Si basa anche questo su TS ma i messaggi che possono girare sono due. Ogni nodo avra' in una cella di memoria una variabile booleana che dice se sia gia' stato individuato o meno il leader (fino all'arrivo della fase in cui un nodo si dichiara leader l'algoritmo e' uguale all'algoritmo Time Slice), nel caso sia stato individuato i nodi devono attendere la fase successiva a quella con v pari al proprio UID prima di poter trasmettere il messaggio con cui si dichiarano sottoleader, perche' non potendo circolare due messaggi contemporaneamente nello stesso canale nella stessa direzione non si possono sovrapporre il messaggio del leader che si dichiara tale a quello in cui il nodo si dichiara sotto-leader; quindi ponendo di essere nella fase $f = n \cdot c + i$ e il nodo i ha proprio come UID f allora

- se non gli e' arrivato nessun token in precedenza si dichiara leader e trasmette il suo token
- se gli arriva un token in questa time slice o gli e' arrivato un token in precedenza (ha la cella contenente la variabile booleana uguale a $true$) attende una time slice e si possono verificare due casi:
  - se non gli arriva alcun token si dichiara sottoleader e trasmette il suo token
  - se gli arriva un token, il token che gli arriva e' quello del sottoleader, allora il nodo lo ritrasmette

Il tempo dell'algoritmo dipende dall'UID con valore sub-minimo al quale si somma il tempo necessario per comunicare agli altri nodi $\Theta(UID_{sotto-min} + 1) + \Theta(n)$ i messaggi che circoleranno nella rete saranno invece $\Theta(2n)$

9.8.4 Algoritmi Time Slice e derivati con funzionali dipendenti da piu' nodi [anello unidirezionale]

Anche in questo caso basta introdurre una fase preliminare esattamente come fatto nel caso degli algoritmi LCR e non vi e' alcun problema se gli UID come nel caso degli algoritmi LCR non vengono calcolati nel nodo cui fanno riferimento: il nodo k trasmettera' l'UID che 'ospita' se il nodoe lo stato cui questo corrisponde alla fase temporale in cui ci si trova.

10 CONCLUSIONI

Sono state elencate le problematiche concernenti l'intero funzionamento di una rete di sensori che effettui leader election, poi ci si e' pero' concentrati sulla minimizzazione di un opportuno funzionale di costo per trovare una strategia ottima di cambio del nodo leader e sugli algoritmi che permettono la piu'robusta elezione anche di un nodo che sia investito del ruolo di sotto-leader.
Al termine della trattazione compiuta si puó affermare che uno degli scopi che ci si erano prefissi, ovvero controllare l'evoluzione della rete tramite assegnazione dinamica del ruolo di radice non sempre puó essere raggiunto. La strada che si é tentato di percorrere si é rivelata ricca di complicazioni derivanti dalla struttura complessa del modello in esame. Ricordiamo che la rete é stata modellizzata attraverso un sistema non lineare controllabile tramite un ingresso che puó assumere solamente un set limitato di valori. Come giá spiegato questi fattori complicano notevolmente la trattazione formale del problema. Dopo aver cercato di risolvere il problema per via algoritmica attraverso la minimizzazione di funzionali di costo ed attraverso algoritmi che sono in un certo senso di tipo euristico ci si é ricondotti ad un problema di programmazione quadratica grazie all'introduzione di una formulazione approssimativa dell'energia spesa dai vari nodi della rete che si é rivelata necessaria per riuscire ad affrontare il problema dal punto di vista analitico. Anche in questa occasione abbiamo trovato che non sempre il problema é risolvibile nel modo migliore per le nostre aspettative, in particolare abbiamo trovato che per alcune configurazioni di nodi della rete non é possibile far teminare l'evoluzione della rete con lo stesso livello energetico per tutti i nodi.

Per quanto riguarda la parte algoritmica per l'individuazione del leader e del sotto leader sono stati modificati con successo algoritmi esistenti che funzionano per leader election singola. Gli algortimi modificati sono riusciti a mantenere a livello di numero di messaggi e di tempo impiegato prestazioni e taglie simili a quelle dei problemi di leader election singola. Si sono inoltre generalizzati i suddetti algoritmi anche al fine di poter operare le elezioni non solo sugli stati dei singoli nodi bensi' su funzionali dipendenti da piu' nodi vicini.
Possibili futuri sviluppi potrebbero prevedere una maggiore integrazione tra le due parti principali di questo lavoro proponendo funzionali 'ad hoc' per la scelta migliore al contempo di leader e sottoleader modellizzando quindi al loro interno anche le comunicazioni verso quest'ultimo, un'ulteriore sviluppo potrebbe essere quello di integrare la parte del controllo della vitalita' dei nodi all'interno delle altre fasi e del funzionale di costo.

APPENDICE A

Codice MATLAB CAP 2.

A.1 Implementazione della funzione $\Phi(\bullet)$.

```
function M = Phi(H)
dim = size (H);
v = H*ones(dim(1,2),1);
M = zeros (dim(1,1));
for i = 1:dim(1,1)
    if v(i) ~= 0
        M(i,i) = 1;
    end
end
```

A.2 Implementazione della funzione $M_a(\bullet)$ con tecnica PrimoUno.

```
function M = MaPrimoUno (Mg,xr)
dim = size (Mg);
A = Phi(xr);
F = (eye(dim(1,1)) -A) * Mg * A;
P = F;
while norm(P) > 0
    A = A + P;
    P = (eye(dim(1,1)) -Phi(A)) * Mg * Phi(P);
    %Voglio che in ogni riga di P ci sia al massimo un 1
    for i = 1:dim(1,1)
        y = 0;
        for w = 1:dim(1,1)
            if (y == 1)%Se ho gia trovato un 1 metto tutti
                       %gli altri elementi a zero
                P(i,w) = 0;
            end
            if (P(i,w) == 1)%Appena trovo un 1 metto y a 1
                y = 1;
            end
        end
    end
    F = F + P;
end
M=F';
```

A.3 Implementazione della funzione $M_a(\bullet)$ con tecnica FigliCasuali.

```
function M = MaFigliCasuali (Mg,xr)
dim = size (Mg);
A = Phi(xr);
F = (eye(dim(1,1)) -A) * Mg * A;
P = F;
```

```
while norm(P) > 0
    A = A + P;
    P = (eye(dim(1,1)) -Phi(A)) * Mg * Phi(P);
    for i = 1:dim(1,1)
        y = 0;
        d = fix(1 + dim(1,1) * rand);
        verso = binornd(1,0.5);
        if (verso == 1)
            for w = 1:dim(1,1)
                if (y == 1)%Se ho gia trovato un 1
                           %metto tutti gli altri elementi a zero
                    P(i,d) = 0;
                end
                if (P(i,d) == 1)%Appena trovo un 1 metto y a 1
                    y = 1;
                end
                if (d < dim(1,1))
                    d = d + 1;
                else
                    d = 1;
                end
            end
        else
            for w = 1:dim(1,1)
                if (y == 1)%Se ho gia trovato un 1
                           %metto tutti gli altri elementi a zero
                    P(i,d) = 0;
                end
                if (P(i,d) == 1)%Appena trovo un 1 metto y a 1
                    y = 1;
                end
                if (d > 1)
                    d = d - 1;
                else
                    d = dim(1,1);
                end
            end
        end
    end
    F = F + P;
end
M=F';
```

A.4 Implementazione della funzione $M_a(\bullet)$ con tecnica FigliEquispaziati.

```
function M = MaFigliEquispaziati (Mg,xr)
dim = size (Mg);
A = Phi(xr);
F = (eye(dim(1,1)) -A) * Mg * A;
```

```
P = F;
while norm(P) > 0
    A = A + P;
    P = (eye(dim(1,1)) -Phi(A)) * Mg * Phi(P);
    %Voglio che in ogni riga di P ci sia al massimo un 1
    %Considero la matrice con i primi uni delle righe
    B = P;
    for i = 1:dim(1,1)
        y = 0;
        for w = 1:dim(1,1)
            if (y == 1)%Se ho gia trovato un 1
                       %metto tutti gli altri elementi a zero
                B(i,w) = 0;
            end
            if (B(i,w) == 1)%Appena trovo un 1 metto y a 1
                y = 1;
            end
        end
    end

   v = ones(1,dim(1,1)) * B;%dice quanti 1 ci sono nella colonna i di B
   r = zeros(1,dim(1,1));

   flagOne = 0;%Mi dice fino a quando ciclare
   while (flagOne == 0)
       minv = v(1,1);
       maxv = 0;
       index = 1;
       for i = 1:dim(1,1)%Prendo il massimo degli
                         %elementi di v con rispettivo r = 0
           if (v(1,i) < minv)
               minv = v(1,i);
           end
           if ((v(1,i) > maxv) && (r(1,i) == 0))
               maxv = v(1,i);
               index = i;
           end
       end
       if (minv > (maxv - 2))%Condizione raggiunta la quale si ha che
                             %in ogni riga di P ci sia al massimo un 1
           flagOne = 1;
       else
           %Inizio a scorrere la colonna index-esima di B
           j = 1;
           flagTwo = 0;%Diventa uno appena faccio l'aggiornamento
           while ((flagTwo == 0) && (j < (dim(1,1) + 1)))
           %Ciclo finche non faccio l'aggiornamento o
           %finche non finisco la colonna in questione
               if (B(j,index) == 1)%Se trovo 1 inizio a scorrere
                                   %la riga di P rispettiva
```

```
                    i = index + 1;
                    while ((flagTwo == 0) && (i < (dim(1,1) + 1)))
                    %Ciclo la riga finche non faccio
                    %aggiornamento o finisco la riga
                        if ((P(j,i) == 1) && (v(i) < maxv))
                            v(i) = v(i) + 1;
                            v(index) = v(index) - 1;
                            B(j,index) = 0;
                            B(j,i) = 1;
                            flagTwo = 1;
                        end
                        i = i + 1;
                    end
                end
                j = j + 1;
            end
            if (j == (dim(1,1) + 1)) %Se non ho aggiornato dopo
                                     %aver passato tutta la colonna
                                     %di B il nodo ha il minimo
                                     %numero di figli
                r(index) = 1;
            end
        end
    end
    F = F + B;
end
M=F';
```

APPENDICE B

Codice MATLAB CAP 3.

B.1 Funzione di scelta della radice con criterio FigliEquispaziati.

```
function [W,M,G] = RadiceFigliEquispaziati (statoprec, Mg,C,radiceprec)

dim = size (Mg);
radicesucc = zeros(dim(1,1),1);
j = 1;
radicesucc(1,1) = 1;

messinvio = (eye(dim(1,1))-Phi(radicesucc)) * ((eye(dim(1,1))-
            Phi(radiceprec))*statoprec + ones(dim(1,1),1));

statosucc = MaFigliEquispaziati(Mg,radicesucc) * ((eye(dim(1,1))-
            Phi(radiceprec))*statoprec + ones(dim(1,1),1));

Q = C + statoprec + messinvio + ones(dim(1,1),1) + statosucc;
W = Q - 1/(dim(1,1)) * ones(1,dim(1,1))* Q * ones(dim(1,1),1);
m = W' * W + Q' * Q ;

radicesucc(1,1) = 0;
for l = 2:dim(1,1)
    radicesucc(l,1) = 1;

    messinvio = (eye(dim(1,1))-Phi(radicesucc)) * ((eye(dim(1,1))-
                Phi(radiceprec))*statoprec + ones(dim(1,1),1));

    statosucc = MaFigliEquispaziati(Mg,radicesucc) * ((eye(dim(1,1))-
                Phi(radiceprec))*statoprec + ones(dim(1,1),1));

    Q = C + statoprec + messinvio + ones(dim(1,1),1) + statosucc;
    W = Q - 1/(dim(1,1)) * ones(1,dim(1,1))* Q * ones(dim(1,1),1);
    n = W' * W + Q' * Q ;

    if (n < m)
        j = l;
        m = n;
    end
    radicesucc(l,1) = 0;
end
W = m;
M = j;
radicesucc = zeros(dim(1,1),1);
radicesucc(j,1) = 1;
G = MaFigliEquispaziati(Mg,radicesucc);

return
```

B.2 Funzione che simula l'evoluzione della rete con criterio FigliEquispaziati, scelta della radice che minimizza $J(\bullet)$ e TempoInteradice fissato.

```
function [T,X,K] = UniformitaFlussoMessaggiFigliEquispaziati
                   (TempoInterRadice,Mg,tempo)

dim = size(Mg);
xOTTIMA = zeros (dim(1,1),1);
JOTTIMA = 0;
Somma = zeros (dim(1,1),1);
A = zeros(dim(1,1),1);

radiceatt = zeros(dim(1,1),1);
radiceatt(1,1) = 1;
radicesucc = zeros(dim(1,1),1);
radicesucc(1,1) = 1;
posrad = 1;

F = MaFigliEquispaziati(Mg,radicesucc);

xOTTIMA(:,2) = F * ((eye(dim(1,1))-Phi(radiceatt))*
               xOTTIMA(:,1) + ones(dim(1,1),1));

z = 0;
for i = 2:tempo

    messinvio = (eye(dim(1,1))-Phi(radicesucc)) * ((eye(dim(1,1))-
    Phi(radiceatt))*xOTTIMA(:,(i-1)) + ones(dim(1,1),1));

    A = A + xOTTIMA(:,(i-1)) + messinvio + ones(dim(1,1),1);

    z = z + 1;
    if (z == TempoInterRadice)
        [valoreJ,posrad,F] = RadiceFigliEquispaziati (xOTTIMA(:,i),
                             Mg,A,radicesucc);
        z = 0;
    end

    radiceatt = radicesucc;
    radicesucc = zeros(dim(1,1),1);
    radicesucc(posrad,1) = 1;

    xOTTIMA(:,(i+1)) = F * ((eye(dim(1,1))-Phi(radiceatt))*
                       xOTTIMA(:,i) + ones(dim(1,1),1));

    messinvio = (eye(dim(1,1))-Phi(radicesucc)) * ((eye(dim(1,1))-
                 Phi(radiceatt))*xOTTIMA(:,i) + ones(dim(1,1),1));
```

```
    Somma(:,i) = A + xOTTIMA(:,i) + messinvio + ones(dim(1,1),1) +
                 xOTTIMA(:,(i+1));
    W = Somma(:,i) - 1/(dim(1,1)) * ones(1,dim(1,1))* Somma(:,i)
        * ones(dim(1,1),1);

    JOTTIMA(i,1) = W' * W;

end
K = JOTTIMA;
X = xOTTIMA;
T = Somma;
```

B.3 Funzione che simula l'evoluzione della rete con criterio FigliEquispaziati, scelta della radice che minimizza $J(\bullet)$ e TempoInteradice minimo ricalcolato ad ogni spostamento della radice.

```
function [T,X,K] = UniformitaFlussoMessaggiFigliEquispaziati
                   TempoInterRadiceMinimo (Mg,tempo)

dim = size(Mg);
xOTTIMA = zeros (dim(1,1),1);
JOTTIMA = 0;
Somma = zeros (dim(1,1),1);
A = zeros(dim(1,1),1);

radiceatt = zeros(dim(1,1),1);
radiceatt(1,1) = 1;
radicesucc = zeros(dim(1,1),1);
radicesucc(1,1) = 1;
posrad = 1;

F = MaFigliEquispaziati(Mg,radicesucc);
xOTTIMA(:,2) = F * ((eye(dim(1,1))-Phi(radiceatt))*xOTTIMA(:,1)
               + ones(dim(1,1),1));

% Trovo il numero di livelli dell'albero
v = F * ones(dim(1,1),1); % Gli elementi nulli del vettore
                          % v corrispondono alle foglie
v = ones(dim(1,1),1) - v;
s = 0;
while (norm(v) > 0)
    s = s + 1;
    v = F * v;
end
%Alla fine delle iterazioni s rappresenta
%il numero di livelli dell'albero
TempoInterRadice = s;
```

```
z = 0;
for i = 2:tempo

    messinvio = (eye(dim(1,1))-Phi(radicesucc)) * ((eye(dim(1,1))-
    Phi(radiceatt))*xOTTIMA(:,(i-1)) + ones(dim(1,1),1));

    A = A + xOTTIMA(:,(i-1)) + messinvio + ones(dim(1,1),1);

    z = z + 1;
    if (z == TempoInterRadice)
        [valoreJ,posrad,F] = RadiceFigliEquispaziati (xOTTIMA(:,i),
                             Mg,A,radicesucc);

        % Trovo il numero di livelli dell'albero
        v = F * ones(dim(1,1),1); % Gli elementi nulli del vettore
                                  % v corrispondono alle foglie
        v = ones(dim(1,1),1) - v;
        s = 0;
        while (norm(v) > 0)
            s = s + 1;
            v = F * v;
        end
        %Alla fine delle iterazioni s rappresenta il
        %numero di livelli dell'albero

        TempoInterRadice = s;

        z = 0;
    end

    radiceatt = radicesucc;
    radicesucc = zeros(dim(1,1),1);
    radicesucc(posrad,1) = 1;

    xOTTIMA(:,(i+1)) = F * ((eye(dim(1,1))-Phi(radiceatt))*
                       xOTTIMA(:,i) + ones(dim(1,1),1));

    messinvio = (eye(dim(1,1))-Phi(radicesucc)) * ((eye(dim(1,1))-
                 Phi(radiceatt))*xOTTIMA(:,i) + ones(dim(1,1),1));

    Somma(:,i) = A + xOTTIMA(:,i) + messinvio + ones(dim(1,1),1)
                 + xOTTIMA(:,(i+1));

    W = Somma(:,i) - 1/(dim(1,1)) * ones(1,dim(1,1))* Somma(:,i)
        * ones(dim(1,1),1);

    JOTTIMA(i,1) = W' * W;

end
K = JOTTIMA; X = xOTTIMA; T = Somma;
```

APPENDICE C

Codice MATLAB CAP 4.

C.0.1 Funzione di scelta della radice con criterio FigliEquispaziati.

```
function [radice,M] = RadiceMassimaBatteriaFigliEquispaziati (statoprec,Mg)

dim = size(statoprec);
n = statoprec(1,1);
k = 1;
for i = 2:dim(1,1)
    if (n > statoprec(i,1))
        n = statoprec(i,1);
        k = i;
    end
end
rad = zeros(dim(1,1),1);
rad(k,1) = 1;
M = MaFigliEquispaziati(Mg,rad);
radice = rad;
```

C.0.2 Funzione che simula l'evoluzione della rete con criterio FigliEquispaziati, scelta della radice secondo il criterio MassimaBatteria e TempoInteradice minimo ricalcolato ad ogni spostamento della radice.

```
function [H W] = MassimaBatteriaFigliEquispaziati
                 TempoInterRadiceMinimo (Mg,tempo)

dim = size(Mg);
xMaxBatOTTIMA = zeros (dim(1,1),1);
JMaxBatOTTIMA = 0;
JMaxBatOTTIMALivelloMinimo = 0;
A = zeros(dim(1,1),1);
B = 0;

radicefut = zeros(dim(1,1),1);
radicefut(1,1) = 1;
radiceatt = zeros(dim(1,1),1);
radiceatt(1,1) = 1;
radicesucc = zeros(dim(1,1),1);
radicesucc(1,1) = 1;

F = MaFigliEquispaziati(Mg,radicesucc);

% Trovo il numero di livelli dell'albero
v = F * ones(dim(1,1),1); % Gli elementi nulli del vettore
                          % v corrispondono alle foglie
```

```
v = ones(dim(1,1),1) - v;
s = 0;
while (norm(v) > 0)
    s = s + 1;
    v = F * v;
end
%Alla fine delle iterazioni s rappresenta
%il numero di livelli dell'albero
TempoInterRadice = s;

xMaxBatOTTIMA(:,2) = F * ones(dim(1,1),1);

z = 0;
for i = 2:tempo
    messinvio = (eye(dim(1,1))-Phi(radicesucc)) * ((eye(dim(1,1))-
                Phi(radiceatt))*xMaxBatOTTIMA(:,(i-1)) + ones(dim(1,1),1));
    A = A + xMaxBatOTTIMA(:,(i-1)) + messinvio + ones(dim(1,1),1);

    B = A + xMaxBatOTTIMA(:,i);

    n = B(1,1);
    pos = 1;
    for j = 2:dim(1,1)
        if (B(j,1) > n)
            n = B(j,1);
            pos = j;
        end
    end
    JMaxBatOTTIMALivelloMinimo(i,1) = n;


    z = z + 1;
    if (z == TempoInterRadice)
        [radicefut,F] = RadiceMassimaBatteriaFigliEquispaziati (B,Mg);

        % Trovo il numero di livelli dell'albero
        v = F * ones(dim(1,1),1); % Gli elementi nulli del vettore
                                  % v corrispondono alle foglie
        v = ones(dim(1,1),1) - v;
        s = 0;
        while (norm(v) > 0)
            s = s + 1;
            v = F * v;
        end
        %Alla fine delle iterazioni s rappresenta
        %il numero di livelli dell'albero
        TempoInterRadice = s;
```

```
        z = 0;
    end

    radiceatt = radicesucc;
    radicesucc = radicefut;
    xMaxBatOTTIMA(:,(i+1)) = F * ((eye(dim(1,1))-Phi(radiceatt))*
                             xMaxBatOTTIMA(:,i) + ones(dim(1,1),1));

    messinvio = (eye(dim(1,1))-Phi(radicesucc)) * ((eye(dim(1,1))-
                 Phi(radiceatt))*xMaxBatOTTIMA(:,i) + ones(dim(1,1),1));
    Q = B + messinvio + ones(dim(1,1),1) + xMaxBatOTTIMA(:,(i+1));
    W = Q - 1/(dim(1,1)) * ones(1,dim(1,1))* Q * ones(dim(1,1),1);

    JMaxBatOTTIMA(i,1) = W' * W + Q'*Q;
end
W = JMaxBatOTTIMA;
H = JMaxBatOTTIMALivelloMinimo;
```

APPENDICE D

Codice MATLAB CAP 5.

D.0.3 Funzione di scelta della radice con criterio FoglieScariche.

```
function [radice,M] = RadiceFoglieScaricheFigliEquispaziati (livello,pos,Mg)
dim = size(Mg);


batteriafoglie = -1;
posrad = 1;
B = [];

for i = 1:dim(1,1)
    rad = zeros(dim(1,1),1);
    rad(i,1) = 1;

    F = MaFigliEquispaziati(Mg,rad);
    v = F * ones(dim(1,1),1);%Invio dei messaggi. Le foglie
                             %sono i nodi che riceveranno zero messaggi
    for j = 1:dim(1,1)
        if (v(j,1) > 0)
            v(j,1) = 1;
        end
    end

    u = livello' * (ones(dim(1,1),1) - v);
    if (u > batteriafoglie)%Prendo la radice che nel complesso
                           %prende come foglie i nodi piu scarichi
        posrad = i;
        batteriafoglie = u;
        B = F;
    end
end
rad = zeros(dim(1,1),1);
rad(posrad,1) = 1;
radice = rad;

M = B;
return
```

APPENDICE E

Codice MATLAB CAP 6.

E.1 Funzione di scelta della radice con criterio FigliEquispaziati.

```
function [W,M,G] = RadiceFigliEquispaziati (statoprec, Mg,C,J,
                   alfa,beta,radiceprec)
dim = size (Mg);
radice = zeros(dim(1,1),1);
j = 1;
radice(1,1) = 1;

B = MaFigliEquispaziati(Mg,radice);


Q = C + B * ones(dim(1,1),1) + (alfa * eye(dim(1,1)) +
    (beta - alfa) * Phi(radiceprec)) * (statoprec + ones(dim(1,1),1));
W = Q - 1/(dim(1,1)) * ones(1,dim(1,1))* Q * ones(dim(1,1),1);

m = W' * W ;

radice(1,1) = 0;
for l = 2:dim(1,1)
    radice(l,1) = 1;

    A = MaFigliEquispaziati(Mg,radice);

    Q = C + A * ones(dim(1,1),1)+ (alfa * eye(dim(1,1)) + (beta - alfa)
        * Phi(radiceprec)) * (statoprec + ones(dim(1,1),1));
    W = Q - 1/(dim(1,1)) * ones(1,dim(1,1))* Q * ones(dim(1,1),1);
    n = W' * W;

    if (n < m)
        j = l;
        m = n;
        B = A;
    end
    radice(l,1) = 0;
end
W = m;
M = j;
G = B;
return
```

E.2 Funzione che simula l'evoluzione della rete con Sensor-Fusion con criterio FigliEquispaziati, scelta della radice secondo il criterio UniformitáFlusso e TempoInteradice fisso.

```
function [T,K] = UniformitaFlussoMessaggiFigliEquispaziati
                 (TempoInterRadice,Mg,tempo,alfa,beta)

dim = size(Mg);
xOTTIMA = zeros (dim(1,1),1);
JOTTIMA = 0;
JOTTIMALivelloMinimo = 0;
A = zeros(dim(1,1),1);

radiceprec = zeros(dim(1,1));
radiceprec(1,1) = 1;

[valoreJ,posrad,F] = RadiceFigliEquispaziati (xOTTIMA(:,1)
                     ,Mg,A,JOTTIMA,alfa,beta,radiceprec);
xOTTIMA(:,2) = F * ones(dim(1,1),1);


radice = zeros(dim(1,1),1);
radice(posrad,1) = 1;
radicefut = radice;

A = xOTTIMA(:,2)+ (eye(dim(1,1)) - Phi(radice))*ones(dim(1,1),1)
    + (alfa * eye(dim(1,1)) + (beta - alfa) * Phi(radiceprec))
    * (xOTTIMA(:,1) + ones(dim(1,1),1));

W = A - 1/(dim(1,1)) * ones(1,dim(1,1))* A * ones(dim(1,1),1);
JOTTIMA(1,1) = W' * W;



z = 0;
for i = 2:tempo

    z = z + 1;
    if (z == TempoInterRadice)
        [valoreJ,posrad,F] = RadiceFigliEquispaziati (xOTTIMA(:,i),
                             Mg,A,JOTTIMA((i-1),1),alfa,beta,radiceprec);
        z = 0;
        radicefut = zeros(dim(1,1),1);
        radicefut(posrad,1) = 1;
    end

    radiceprec = radice;
    radice = radicefut;
```

```
    xOTTIMA(:,(i+1)) = F * ones(dim(1,1),1);

    A = A+ (eye(dim(1,1)) - Phi(radice))*ones(dim(1,1),1) + (alfa *
        eye(dim(1,1)) + (beta - alfa) * Phi(radiceprec)) *
        (xOTTIMA(:,i) + ones(dim(1,1),1));

    B = A + xOTTIMA(:,(i+1));

    W = B - 1/(dim(1,1)) * ones(1,dim(1,1))* B * ones(dim(1,1),1);
    JOTTIMA(i,1) = W' * W;

    n = B(1,1);
    pos = 1;
    for j = 2:dim(1,1)
        if (B(j,1) > n)
            n = B(j,1);
            pos = j;
        end
    end
    JOTTIMALivelloMinimo(i,1) = n;
end

K = JOTTIMA;
T = JOTTIMALivelloMinimo;
```

E.3 Funzione che simula l'evoluzione della rete con Sensor-Fusion con criterio FigliEquispaziati, scelta della radice secondo il criterio UniformitáFlusso e TempoInteradice minimo ricalcolato ad ogni spostamento della radice.

```
function [T,K] = UniformitaFlussoMessaggiFigli
                 EquispaziatiMinimo (Mg,tempo,alfa,beta)

dim = size(Mg);
xOTTIMA = zeros (dim(1,1),1);
JOTTIMA = 0;
JOTTIMALivelloMinimo = 0;
A = zeros(dim(1,1),1);

radiceprec = zeros(dim(1,1));
radiceprec(1,1) = 1;

[valoreJ,posrad,F] = RadiceFigliEquispaziati (xOTTIMA(:,1),
                     Mg,A,JOTTIMA,alfa,beta,radiceprec);

% Trovo il numero di livelli dell'albero
v = F * ones(dim(1,1),1); % Gli elementi nulli del vettore
                          % v corrispondono alle foglie
```

```
v = ones(dim(1,1),1) - v;
s = 0;
while (norm(v) > 0)
    s = s + 1;
    v = F * v;
end

%Alla fine delle iterazioni s rappresenta il
%numero di livelli dell'albero

TempoInterRadice = s;

xOTTIMA(:,2) = F * ones(dim(1,1),1);

radice = zeros(dim(1,1),1);
radice(posrad,1) = 1;
radicefut = radice;

A = xOTTIMA(:,2)+ (eye(dim(1,1)) - Phi(radice))*ones(dim(1,1),1)
    + (alfa * eye(dim(1,1)) + (beta - alfa) * Phi(radiceprec))
    * (xOTTIMA(:,1) + ones(dim(1,1),1));

W = A - 1/(dim(1,1)) * ones(1,dim(1,1))* A * ones(dim(1,1),1);
JOTTIMA(1,1) = W' * W;

z = 0;
for i = 2:tempo

    z = z + 1;
    if (z == TempoInterRadice)
        [valoreJ,posrad,F] = RadiceFigliEquispaziati (xOTTIMA(:,i),
                             Mg,A,JOTTIMA((i-1),1),alfa,beta,radiceprec);

        % Trovo il numero di livelli dell'albero
        v = F * ones(dim(1,1),1); % Gli elementi nulli del vettore
                                  % v corrispondono alle foglie
        v = ones(dim(1,1),1) - v;
        s = 0;
        while (norm(v) > 0)
            s = s + 1;
            v = F * v;
        end
        %Alla fine delle iterazioni s rappresenta
        %il numero di livelli dell'albero
        TempoInterRadice = s;

        z = 0;
        radicefut = zeros(dim(1,1),1);
        radicefut(posrad,1) = 1;
```

```
    end

    radiceprec = radice;
    radice = radicefut;

    xOTTIMA(:,(i+1)) = F * ones(dim(1,1),1);

    A = A+ (eye(dim(1,1)) - Phi(radice))*ones(dim(1,1),1) +
        (alfa * eye(dim(1,1)) + (beta - alfa) * Phi(radiceprec))
        * (xOTTIMA(:,i) + ones(dim(1,1),1));

    B = A + xOTTIMA(:,(i+1));

    W = B - 1/(dim(1,1)) * ones(1,dim(1,1))* B * ones(dim(1,1),1);
    JOTTIMA(i,1) = W' * W;

    n = B(1,1);
    pos = 1;
    for j = 2:dim(1,1)
        if (B(j,1) > n)
            n = B(j,1);
            pos = j;
        end
    end
    JOTTIMALivelloMinimo(i,1) = n;
end

K = JOTTIMA;
T = JOTTIMALivelloMinimo;
```

E.4

Codice MATLAB CAP 7.

E.4.1 Implementazione del metodo di controllo ValoreFinaleMinimo.

```
function soluzione = Msistema(Mg,tempo,alfa)
dim = size (Mg);

% Trovo la matrice da risolvere
S = [];
for i = 1:dim(1,1)
    rad = zeros(dim(1,1),1);
    rad(i,1) = 1;
    O = MaFigliEquispaziati(Mg,rad);
    O = O * ones(dim(1,1),1);
    S = [S O];
end

Q = inv(S) * ones(dim(1,1),1) - quadprog (2*S'*S ,
    zeros(dim(1,1),1) , eye(dim(1,1)) , (inv(S) * ones(dim(1,1),1)));

% Normalizzo la soluzione in funzione del tempo di simulazione
lambda = (alfa+1) * (tempo+1) + alfa * (tempo+1) / (ones(1,dim(1,1)) * Q);
Q = Q * (lambda - (1+alfa) * (tempo+1)) / alfa;

% Approssimo la soluzione a valori naturali
vet = zeros(dim(1,1),1);
for i = 1: dim(1,1)
    vet(i,1) = fix (Q(i,1));
end

n = tempo - vet' * ones(dim(1,1),1); % Quanti valori perdo con la
                                     % approssimazione. Devo rimetterli

for i = 1: n
    vet (i,1) = vet (i,1) + 1;
end

soluzione = vet;
```

ACKNOWLEDGMENTS

The authors would like to thank...our families

RIFERIMENTI BIBLIOGRAFICI

[1] C. Lora, *Leader election in sistemi multi-agenti con intelligenza distribuita*, Tesi di laurea, 2007.
[2] Andrew S. Tanenbaum, Maarten Van Steen, *Distributed Systems: Principles and Paradigms*, 2nd ed. Prentice Hall 2007.
[3] Lynch, Nancy A., *Distributed Algorithms*, I Morgan Kaufmann, 1997.
[4] Hector Garcia-Molina, *Election in a Distributed Computing System*, 1982.
[5] J. Brunekreef, J.P. Katoen, R. Koymans, S. Mauw *Design and Analisys of Dinamic Leader Election Protocol in Broadcast Network*, 1993.
[6] Scott D. Stoller, *Leader Election in Asyncronous Distributed System*, 2000.
[7] S. Vasudevan, J. Kurose, D Towsley, *Design and Analisys of a Leader Election Algorithm for Mobile Ad Hoc Network*
[8] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.

Claudio Trinca Studente del corso di laurea specialistica di ingegneria dell'automazione presso il DEI di Padova, laureato in ingegneria dell'informazione presso lo stesso dipartimento

Paolo Ticozzi Studente del corso di laurea specialistica di ingegneria dell'automazione presso il DEI di Padova, laureato in ingegneria dell'informazione presso lo stesso dipartimento