UNIVERSITÀ DEGLI STUDI DI PADOVA

Facoltà di Ingegneria
Corso di Laurea Magistrale in Ingegneria dell'Automazione

Corso di Progettazione di Sistemi di Controllo
a.a. 2011-2012

Average Consensus con Perdita di Pacchetto

Moscatelli Luca, Sciortino Alessandro

Padova, 22 marzo 2012

Indice

Abstract

Lo scopo di questo progetto è la valutazione di alcuni algoritmi di average consensus nel caso di perdita di pacchetto; in letteratura sono presenti molti algoritmi di average consensus, tuttavia solo alcuni di questi tengono conto della perdita di pacchetto. In questo lavoro sono stati implementati i suddetti algoritmi introducendo, ove già non vi fosse, la perdita di pacchetto; tipicamente questa operazione è stata effettuata con l'aggiunta di variabili aleatorie binomiali rappresentanti la probabilità che si incorra nella perdita di dati. Nel corso del progetto vengono trattate varie tipologie di comunicazione, distinguendo reti sincrone, reti asincrone, comunicazione di tipo broadcast, di tipo gossip e di tipo gossip esteso basato su broadcast. Inoltre sono previsti diversi metodi di reazione al fallimento della comunicazione: si vedrà cosa accade sia nel caso in cui un agente sia a conoscenza dell'avvenuto tentativo di trasmissione, sia nel caso in cui non lo sia. Obiettivo finale del progetto è valutare la robustezza degli algoritmi ottenuti dopo aver introdotto la perdita di pacchetto.

1 Introduzione

Si parla di consensus quando si è in presenza di un problema in cui vari agenti cercano di sincronizzarsi raggiungendo un valore di stima comune; quando questo valore corrisponde alla media delle stime dei singoli agenti, si parla di average consensus. Un'applicazione pratica consiste ad esempio nello scambio di dati fra agenti di una stessa rete, quali sensori che comunicano fra loro tramite una rete wireless; più precisamente, si assuma che ogni sensore compia un'operazione di misura in modo indipendente dagli altri agenti; questa misura sarà affetta da un certo errore; un semplice modo per minimizzare la rilevanza dell'errore consiste nel fare la media delle misure effettuate dai sensori; affinchè ciò sia possibile, è necessario che i diversi agenti comunichino fra loro, e che quindi facciano parte di una rete di comunicazione. Gli algoritmi di average consensus sono quegli algoritmi che permettono ai vari sensori di convergere alla media delle misure; ovviamente sarà tanto più facile convergere alla media delle misure quanto maggiori saranno le interazioni frai i sensori della rete; una situazione ideale è quella in cui ogni agente può comunicare con tutti gli altri nodi in modo da avere quante più informazioni possibili che lo aiutino a raggiungere la media. Tuttavia spesso vi sono delle limitazioni fisiche che impediscono una comunicazione capillare fra i vari agenti. In particolare in questo lavoro verranno presi in considerazione algoritmi distribuiti (quindi algoritmi in cui ogni nodo della rete ha eguale potere

decisionale riguardo le azioni da compiere) sia per reti sincrone che asincrone; in una rete sincrona ad ogni istante temporale ogni agente comunica con tutti i suoi vicini simultaneamente, mentre in una rete asincrona avviene solo un certo numero di comunicazioni istante per istante. Un'ulteriore distinzione sarà fatta fra diverse tipologie di comunicazione: si vedranno comunicazioni di tipo broadcast, di tipo gossip e di tipo gossip esteso basato su broadcast; in una comunicazione di tipo broadcast un agente della rete trasmette informazioni agli agenti vicini; gli altri nodi della rete non ricevono nè trasmettono; in una comunicazione di tipo gossip, due nodi contigui trasmettono fra loro, mentre gli altri sono inattivi; nel caso di gossip esteso basato su broadcast, la struttura di comunicazione gode di un forte parallelismo: presa una qualunque coppia di nodi distanti l'uno dall'altro, ognuno dei due può avviare una trasmissione di tipo broadcast senza interferire con la comunicazione avviata dell'altro agente.
In un contesto come quello delle reti di comunicazione, è possibile che durante la trasmissione si perdano alcuni pacchetti di dati; il contributo di questo lavoro consiste nell'implementare la perdita di pacchetto in alcuni dei numerosi algoritmi presenti in letteratura che permettono di raggiungere l'average consensus; questo fenomeno è stato modellato introducendo delle variabili aleatorie binomiali, il cui scopo è rappresentare la probabilità che la trasmissione di dati da un nodo all'altro della rete vada a buon fine. Ovviamente ci si aspetta che questa modifica peggiori le prestazioni degli algoritmi, infatti sarà mostrato che anche se la perdita di dati avviene a livello locale, gli effetti influenzano il comportamento globale della rete. Lo scopo finale del progetto è valutare le prestazioni degli algoritmi con la perdita di pacchetto in funzione dell'errore e della velocità di convergenza; in particolare si vedrà che i risultati migliori in termini di errore di convergenza si ottengono con degli algoritmi broadcast in cui l'agente che avrebbe dovuto ricevere l'informazione non è a conoscenza della tantata trasmissione di dati; ovviamente basandosi sulla velocità di convergenza si mostrerà che gli algoritmi sincroni forniscono prestazioni migliori rispetto agli asincroni.
Nelle sezioni successive verrà descritto lo stato dell'arte, illustrando i lavori presenti in letteratura che si occupano di average consensus e di perdita di pacchetto; quindi saranno presentati in modo più dettagliato gli algoritmi presi in considerazione in questo progetto; successivamente si spiegherà come questi algoritmi sono stati modificati con l'introduzione della perdita di pacchetto. Infine saranno presentati i risultati delle simulazioni, in base ai quali verranno valutati e confrontati i diversi algoritmi, mentre nel capitolo conclusivo si tirano le somme di tutto il lavoro svolto, fornendo utili indicazioni sui tipi di comunicazioni da preferire e suggerendo possibili migliorie e sviluppi futuri.

Stato dell'arte

Come già anticipato esiste una vasta letteratura che tratta problemi di average consensus, tuttavia solo in rari casi è stato affrontato il problema della perdita di pacchetto. Lo scenario in cui ci si pone è quello delle reti di comunicazione, che verranno implementate tramite dei grafi; in questo progetto si lavorerà esclusivamente con grafi diretti, ossia con grafi in cui la comunicazione da un agente all'altro è unidirezionale e non bidirezionale; verranno presi in considerazione diversi tipi di rete e diversi tipi di comunicazione fra i vari agenti. Per quanto riguarda reti sincrone si può fare riferimento a [6], mentre per reti asincrone a [4], [5], e [6]. In particolare in [4] e in [6] si tratta il problema dell'average consensus con comunicazione di tipo gossip; in [5] invece si utilizza una tipologia di comunicazione in parallelo riconducibile al broadcast, che verrà illustrata approfonditamente nella sezione 3.5. Il tema d'interesse di questo lavoro è già stato trattato solo in [2], e consiste nell'introdurre la perdita di pacchetto in alcuni algoritmi nei quali non è prevista. La perdita di pacchetto entra in gioco ad esempio nell'ambito di reti di sensori che comunicano fra loro tramite una rete wireless, dove la perdita di alcuni dati è uno scenario che si presenta spesso; a tal proposito negli algoritmi basati su comunicazione broadcast (che si riveleranno essere i più interessanti) verrà fatta una distinzione a seconda che il sensore che avrebbe dovuto ricevere informazione sia a conoscenza o meno del tentativo di trasmettere.

2 Formulazione del problema

In questa sezione saranno descritte le tematiche affrontate, introducendo i concetti di consensus, average consensus e perdita di pacchetto; tuttavia sono prima necessarie alcune nozioni di teoria dei grafi.

2.1 Cenni di teoria dei grafi

Per modellare le reti di comunicazione sono stati usati grafi orientati ed è quindi doveroso descrivere brevemente la terminologia ed i principali risultati teorici del settore. Un grafo orientato (o grafo diretto o digrafo) è una coppia $G = (V, E)$ in cui V è un insieme finito di vertici (o nodi) ed E è una famiglia di coppie ordinate di vertici, dette archi; gli archi sono quindi coppie del tipo (i, j) con $i, j \in V$; l'ordine in cui compaiono i due vertici è rilevante, quindi in generale $(i, j) \neq (j, i)$. Un cammino orientato è una sequenza $e_1, e_2, \ldots, e_k$ di archi consecutivi del tipo $e_1 = (v_1, v_2)$, $e_2 = (v_2, v_3)$, $\ldots$, $e_k = (v_k, v_{k+1})$. Un nodo j si dice connesso ad un nodo i se esiste un cammino orientato da i a j; un digrafo si dice connesso se tutte le coppie di nodi sono connesse; un digrafo si dice fortermente connesso se per ogni coppia di nodi i, j si ha che j è connesso ad i, cioè esiste un cammino orientato da i a j. Un'arborescenza è un grafo parziale $G' = (V, E')$ di G in cui un vertice speciale (la radice) può raggiungere ogni altro vertice del grafo. Un nodo i di un digrafo è detto bilanciato se il numero di archi entranti $\delta_{in,i}(t)$ è pari al numero di archi uscenti $\delta_{out,i}(t)$. Si definisce grafo bilanciato un digrafo i cui nodi sono tutti bilanciati. Inoltre, assumendo che il grafo abbia n nodi, si definiscono le due matrici $n \times n$ $\Delta_{in}(t) = \mathrm{diag}(\delta_{in,1}(t), \ldots, \delta_{in,n}(t))$ e $\Delta_{out}(t) = \mathrm{diag}(\delta_{out,1}(t), \ldots, \delta_{out,n}(t))$. Infine si definisce l'insieme dei vicini del nodo i come $\mathcal{N}_i = \{j : (j, i) \in E\}$.
I grafi possono essere descritti da opportune matrici, dette matrici di adiacenza, che giocano un ruolo importante nella formalizzazione matematica di alcuni problemi di ottimizzazione. La matrice di adiacenza $A = [a_{ij}]$ di un digrafo è la matrice con elementi $a_{ij} = 1$ se $(j, i) \in E$ e $a_{ij} = 0$ altrimenti; se nella matrice al posto degli 1 si trovano dei numeri, questi devono essere interpretati come il peso attribuito a ciascun arco. La matrice laplaciana $L = [l_{ij}]$ di un digrafo è la matrice con elementi $l_{ii} = \sum_j^n a_{ij}$ e $l_{ij} = -a_{ij}$, dove $i \neq j$. Infine una matrice è detta stocastica per righe (colonne) se la somma degli elementi di ogni sua riga (colonna) è pari ad uno; una matrice si dice doppiamente stocastica (o bistocastica) se è stocastica sia per righe che per colonne.

2.2 Il problema del consenso

In questa sezione si vedranno due diversi tipi di approccio al problema del consenso; il primo approccio è quello classico, mentre il secondo prevede l'introduzione di variabili aggiuntive denominate stati aumentati.

Primo approccio Si assuma di avere una rete di n agenti; ogni agente i misura una quantità $q_i \in \mathbb{R}$ e ad ogni istante temporale t può trasmettere un numero reale ad altri agenti; lo scambio di dati può essere descritto da un digrafo G con vertici $\{1, \dots, n\}$, nel quale esiste un arco (i, j) se e solo se il nodo i è in grado di inviare dati al nodo j. L'obiettivo è trovare un algoritmo distribuito che permetta ai vari agenti di ottenere una stima condivisa della media delle misure q_i. Un algoritmo efficiente che risolve questo problema consiste nel seguente sistema dinamico:

$$x_i(t+1) = \sum_{j=1}^{n} P_{ij} x_j(t), \qquad x_i(0) = q_i(0),$$

dove P è un'opportuna matrice tale che $P_{ij} = 0$ se (i, j) non è un arco in G. Si assume che G includa sempre tutti i self–loop (i, i), e che quindi ogni agente i abbia accesso ai suoi dati. Si può scrivere in modo più compatto

$$x(t+1) = Px(t), \qquad x(0) = q, \tag{1}$$

con $x, q \in \mathbb{R}^n$. Secondo questo algoritmo, l'agente j necessita di ricevere il valore $x_i(t)$ dal nodo i per aggiornare il valore di $x_j(t)$ solo se $P_{ij} \neq 0$. In questo contesto si dice che gli agenti raggiungono il *consenso*, se per ogni condizione iniziale $x(0) \in \mathbb{R}^n$, dal sistema (1) si ottiene

$$\lim_{t \to \infty} x(t) = \mathbf{1}\alpha, \tag{2}$$

dove $\mathbf{1} = (1, \dots, 1)^T$, e α è uno scalare che dipende da $x(0)$ e da P. Inoltre, se α coincide con la media $n^{-1} \sum_{i=1}^{n} q_i = n^{-1} \mathbf{1}^T x(0)$, allora si può dire che gli agenti raggiungono l'*average consensus*.

Secondo approccio Nelle stesse condizioni dell'approccio precedente, si può affermare che risolvere il problema dell'average consensus significa costruire degli algoritmi distribuiti nei quali i nodi aggiornano i loro stati usando solo le informazioni locali sui loro vicini, in modo che tutti gli $x_i(t)$ convergano alla media iniziale $x_a = n^{-1} \mathbf{1}^T x(0)$. La principale difficoltà al raggiungimento della media nei digrafi generali consiste nel fatto che, in generale, la somma degli stati $\mathbf{1}^T x$ non rimane costante, e che quindi è possibile che non si riesca a raggiungere x_a. Per risolvere questo problema, si propone di associare ad ogni nodo i una variabile aggiuntiva $s_i(t) \in \mathbb{R}$, chiamata *stato aumentato*;

si scrive $s(t) = [s_1(t) \cdots s_n(t)]^T \in \mathbb{R}^n$ e si pone $s(0) = 0$. Lo scopo dello stato aumentato è tenere conto dei cambiamenti di stato dei singoli agenti, cosicché $\mathbf{1}^T(x(t) + s(t)) = \mathbf{1}^T x(0)$ per ogni istante t; in altre parole, lo stato aumentato fa sì che la quantità $\mathbf{1}^T(x + s)$ rimanga costante nel tempo.
Altre nozioni riguardanti il consenso sono la convergenza in media quadratica e la convergenza quasi certamente; si dice che una rete di n agenti raggiunge il consenso in media quadratica se per ogni condizione iniziale $(x(0), s(0) = 0)$ si ottiene che $E\left[||x(t) - \alpha\mathbf{1}||_2^2\right] \to 0$ e $E\left[||s(t)||_2^2\right] \to 0$ per $t \to \infty$; si dice che che una rete di n agenti raggiunge il consenso quasi certamente se per ogni condizione iniziale $(x(0), s(0) = 0)$ si ottiene che $(x(t), s(t)) \to (\alpha\mathbf{1}, 0)$ per $t \to \infty$ con probabilità uno; se α coincide con la media iniziale x_a si parla di average consensus in media quadratica e quasi certamente.

3 Descrizione algoritmi

In questa sezione saranno descritti gli algoritmi presi in considerazione nelle loro versioni originali, ossia come compaiono nella bibliografia; i primi due algoritmi sono pensati per reti sincrone, quindi ad ogni istante temporale ogni agente comunica con tutti i suoi vicini simultaneamente; il primo algoritmo è l'unico ad aver già implementata la perdita di pacchetto e prevede due versioni che si differenziano nella scelta di distribuzione dei pesi; successivamente verranno presentati tre algoritmi per reti asincrone differenti nella tipologia di comunicazione; in particolare il primo dei tre è basato su una comunicazione di tipo broadcast, il secondo su una comunicazione di tipo gossip, mentre il terzo ha una struttura di comunicazione in parallelo che verrà illustrata in dettaglio nella sezione specifica.

3.1 Algoritmo Sincrono con Perdita (ASP)

Questo primo algoritmo è basato su reti di comunicazione sincrone ed è l'unico ad aver già implementata la perdita di pacchetto nella sua versione originale.
Si parte da un grafo G e si assume che, per ogni arco (i, j) di G, possa esserci comunicazione dal nodo i al nodo j con una certa probabilità p (è in questo modo che viene introdotta la possibilità che la trasmissione di dati non vada a buon fine, e che quindi ci sia perdita d'informazione); per descrivere meglio questo modello si introduce la famiglia di variabili aleatorie indipendenti $L_{ij}(t)$, $t \in \mathbb{Z}_+$, $i, j = 1, \ldots, n$, $i \neq j$, tale che

$$\mathbb{P}[L_{ij}(t) = 1] = p, \qquad \mathbb{P}[L_{ij}(t) = 0] = 1 - p.$$

Si sottolinea il fatto che si suppone vi sia indipendenza fra tutte le $L_{ij}(t)$ al variare di i, j e t. Sia A la matrice di adiacenza di G, e sia $H = A - I$. Si consideri la matrice $\bar{A}(t) = I + \bar{H}(t)$, dove $\bar{H}_{ij}(t) = H_{ij}L_{ij}(t)$; ovviamente $\bar{A}(t)$ è la matrice di adiacenza di un grafo $\bar{G}(t)$ ottenuto da G eliminando l'arco (i, j) quando $L_{ij}(t) = 0$.
Il primo passo dell'algoritmo consiste nella scelta di una matrice stocastica P tale che G raggiunga l'average consensus; questa matrice P viene quindi modificata per compensare la perdita di alcuni dati; in linea di principio, esistono vari modi per modificare P tenendo conto della perdita di pacchetto; in questo lavoro saranno presentate due possibili soluzioni: nella prima, chiamata *metodo della compensazione parziale* (*biased compensation method*), ogni agente, per aggiornare la stima della media, aggiunge i pesi dei dati non disponibili al peso assegnato alla sua stima precedente; nella seconda, chiamata *metodo della compensazione bilanciata* (*balanced compensation method*), la

compensazione per la perdita di dati viene fatta modificando tutti i pesi in modo più bilanciato. Si sottolinea che si sta ipotizzando che tutti gli agenti siano sincronizzati nel tempo; conseguentemente ad ogni istante temporale t ogni agente conosce i dati che ha ricevuto, quindi il nodo i conosce il valore di $L_{ij}(t)$ per ogni vicino j.

Metodo della compensazione parziale. Si considera la seguente legge di aggiornamento:

$$x_i(t+1) = \left(P_{ii} + \sum_{j \neq i} (1 - L_{ij}(t)) P_{ij} \right) x_i(t) + \sum_{j \neq i} L_{ij}(t) P_{ij} x_j(t).$$

Con questa strategia, l'agente i, per calcolare la nuova stima, compensa la perdita d'informazione sommando i pesi dei dati persi al peso assegnato alla stima precedente; intuitivamente, secondo questo approccio il nodo i sostituisce l'indisponibile $x_j(t)$ con $x_i(t)$ nell'algoritmo di consenso. Definendo le matrici stocastiche $D(t)$ e $Q(t)$ come

$$D_{ij}(t) = \begin{cases} P_{ii} + \sum_{j \neq i} (1 - L_{ij}(t)) P_{ij}) = 1 - \sum_{h \neq i} L_{ih}(t) P_{ih} & \text{se } i = j, \\ 0 & \text{se } i \neq j, \end{cases}$$

e

$$Q_{ij}(t) = \begin{cases} 0 & \text{se } i = j, \\ L_{ij}(t) P_{ij} & \text{se } i \neq j, \end{cases}$$

si può descrivere questo metodo con il sistema stocastico

$$x(t+1) = P(t)x(t), \tag{3}$$

dove

$$P(t) = D(t) + Q(t).$$

Metodo della compensazione bilanciata. A differenza del metodo precedente, in questo caso si preferisce distribuire i pesi equamente fra i dati disponibili; pertanto l'equazione di aggiornamento risulta essere

$$x_i(t+1) = \frac{1}{P_{ii} + \sum_{j \neq i} L_{ij}(t) P_{ij}} \left(P_{ii} x_i(t) + \sum_{j \neq i} L_{ij}(t) P_{ij} x_j(t) \right).$$

In questo caso conviene definire, per $i = 1, \ldots, n$, la variabile binaria casuale L_{ii} che è uguale ad uno con probabilità uno; definendo

$$\nu_i(t) = \sum_{j=1}^{n} L_{ij}(t) P_{ij}$$

e introducendo la matrice diagonale $D(t)$ con

$$D_{ii}(t) = \frac{1}{\nu_i(t)}$$

e la matrice $Q(t)$ tale che

$$Q_{ij}(t) = L_{ij}(t)P_{ij},$$

si può scrivere in modo più compatto

$$x(t+1) = P(t)x(t),$$

dove

$$P(t) = D(t)Q(t).$$

Si assuma che P raggiunga il consenso e che $P_{ii} > 0$ per ogni i; allora, sia con il metodo della compensazione parziale che con il metodo della compensazione bilanciata si raggiungono sia il consenso in media quadratica che il consenso quasi certamente, per ogni condizione iniziale $x(0)$; per maggiori dettagli sulle proprietà di convergenza di questo algoritmo si veda [2].

3.2 Algoritmo Sincrono (AS)

Anche questo algoritmo è pensato per reti sincrone; tuttavia, a differenza del precedente, non prende in esame il caso della perdita di pacchetto e ricorre all'aggiunta di pesi per bilanciare il flusso d'informazione entrante ed uscente; inoltre, al fine di risolvere il problema dell'average consensus, prevede l'introduzione di una variabile denominata stato aumentato (il cui valore iniziale è posto pari a zero).
Si consideri un sistema di n agenti rappresentato da un digrafo $G = (V, E)$; per ogni nodo $i \in V$ sia $\mathcal{N}_i^+ = \{j \in V : (j, i) \in E\}$ l'insieme dei vicini entranti, e $\mathcal{N}_i^- = \{h \in V : (i, h) \in E\}$ l'insieme dei vicini uscenti. Ci sono tre operazioni che ogni nodo i esegue al tempo $t \in \mathbb{Z}_+$:

1. Il nodo i trasmette il suo stato $x_i(t)$ e il suo stato aumentato pesato $b_{ih}s_i(t)$ ad ogni vicino uscente $h \in \mathcal{N}_i^-$; i pesi b_{ih} sono tali che $b_{ih} \in (0, 1)$ se $h \in \mathcal{N}_i^-$, $b_{ih} = 0$ se $h \in V - \mathcal{N}_i^-$, e $\sum_{h \in \mathcal{N}_i^-} b_{ih} < 1$. In particolare in questo progetto si è scelto di porre $b_{ih} = 1/(\text{card}(\mathcal{N}_i^-) + 1)$ per ogni $h \in \mathcal{N}_i^-$.

2. Il nodo i riceve l'informazione di stato $x_j(t)$ e lo stato aumentato pesato $b_{ji}s_j(t)$ da ogni vicino entrante $j \in \mathcal{N}_i^+$.

3. Il nodo i aggiorna il suo stato $x_i(t)$ e il suo stato aumentato $s_i(t)$ come segue:

$$x_i(t+1) = x_i(t) + \sum_{j \in \mathcal{N}_i^+} a_{ij}(x_j(t) - x_i(t)) + \epsilon s_i(t), \tag{4}$$

$$s_i(t+1) = \Big((1 - \sum_{h \in \mathcal{N}_i^-} b_{ih}) s_i(t) + \sum_{j \in \mathcal{N}_i^+} b_{ji} s_j(t)\Big) - \Big(x_i(t+1) - x_i(t)\Big), \tag{5}$$

dove il peso aggiornato a_{ij} è tale che $a_{ij} \in (0,1)$ se $j \in \mathcal{N}_i^+$, $a_{ij} = 0$ se $j \in V - \mathcal{N}_i^+$, e $\sum_{j \in \mathcal{N}_i^+} a_{ij} < 1$; inoltre, il parametro ϵ è un numero positivo che indica il peso dello stato aumentato nell'aggiornamento dello stato (non aumentato). In questo progetto si è scelto di porre $a_{ij} = 1/(\text{card}(\mathcal{N}_i^+) + 1)$ per ogni $j \in \mathcal{N}_i^+$, ed $\epsilon = 0.2$, dato che in [6] tale valore garantisce la stabilità dell'algoritmo.

Si nota che per trasmettere informazioni, ogni agente deve conoscere il numero dei suoi vicini uscenti; inoltre è bene sottolineare che le informazioni trasmesse non dipendono da quelle ricevute. Per quanto riguarda gli stati aumentati, si può notare che sono semplicemente delle variabili aggiunte il cui aggiornamento è dato in parte da somme algebriche di combinizioni lineari di stati aumentati pesati, e in parte dal cambiamento di stato del nodo in questione.
Sia data la matrice aggiunta $A = [a_{ij}]$ del grafo G i cui elementi sono i pesi aggiornati. Si definisce ora la matrice laplaciana L come $L = D - A$, dove $D = \text{diag}(d_1, \ldots, d_n)$ con $d_i = \sum_{j=1}^n a_{ij}$; quindi L ha elementi non negativi in diagonale, non positivi fuori diagonale, e somme di riga pari a zero; conseguentemente la matrice $I - L$ è non negativa e la somma di ogni sua riga è pari ad uno ($I - L$ è stocastica per righe). Sia $B = [b_{ih}]^T$, i cui elementi sono i pesi trasmessi; si definisce la matrice $S = (I - \widetilde{D}) + B$, dove $\widetilde{D} = \text{diag}(\tilde{d}_1, \ldots, \tilde{d}_n)$ con $\tilde{d}_i = \sum_{h=1}^n b_{ih}$; quindi S è non negativa e le sue colonne sommano a uno (ossia S è stocastica per colonne). Come si può notare da (5), la matrice S descrive la parte di aggiornamento indotta dalla trasmissione e dalla ricezione dello stato aumentato.
Grazie all'introduzione delle precedenti matrici, l'aggiornamento degli stati e degli stati aumentati può essere scritto in forma matriciale come segue:

$$\begin{bmatrix} \mathbf{x}(t+1) \\ \mathbf{s}(t+1) \end{bmatrix} = M \begin{bmatrix} \mathbf{x}(t) \\ \mathbf{s}(t) \end{bmatrix}, \quad \text{dove } M = \begin{bmatrix} I - L & \epsilon I \\ L & S - \epsilon I \end{bmatrix}. \tag{6}$$

Si noti che (i) la matrice M ha elementi negativi dovuti alla presenza della matrice laplaciana L nel blocco (2,1); (ii) le somme di colonna di M sono

pari ad uno, la qual cosa implica che la quantità $x(t) + s(t)$ è una costante per ogni $t \geq 0$; (iii) l'evoluzione di stato indotta dal blocco in posizione (1,1) di M, ossia

$$x(t+1) = (I - L)x(t), \tag{7}$$

è quella dell'algoritmo di consenso standard.
Si dimostra che usando l'algoritmo (6) con il parametro ϵ sufficientemente piccolo, gli agenti della rete raggiungono l'average consensus se e solo se il digrafo G è fortemente connesso (per maggiori dettagli si veda [6]).
Nel seguito verranno descritti solo algoritmi per reti asincrone.

3.3 Algoritmo Broadcast (AB)

Questo algoritmo è la generalizzazione per reti asincrone dell'AS appena descritto. La struttura di comunicazione è di tipo broadcast, quindi ad ogni istante temporale un solo nodo della rete (scelto in modo casuale) trasmette informazione ai suoi vicini, mentre gli altri agenti sono inattivi.
Lo scenario è lo stesso dell'algoritmo precedente, con un sistema di n agenti rappresentato dal digrafo $G = (V, E)$, e gli insiemi dei vicini entranti ed uscenti del nodo i, rispettivamente $\mathcal{N}_i^+$ e $\mathcal{N}_i^-$. L'algoritmo è composto da tre passi fondamentali:

1. Il nodo i trasmette il suo stato $x_i(t)$ e il suo stato aumentato pesato $b_{ih}s_i(t)$ ad ogni vicino uscente $h \in \mathcal{N}_i^-$; i pesi b_{ih} sono tali che $b_{ih} \in (0, 1)$ se $h \in \mathcal{N}_i^-$, $b_{ih} = 0$ se $h \in V - \mathcal{N}_i^-$, e $\sum_{h \in \mathcal{N}_i^-} b_{ih} < 1$; anche in questo caso si è posto $b_{ih} = 1/(\text{card}(\mathcal{N}_i^-) + 1)$ per ogni $h \in \mathcal{N}_i^-$.

2. Ogni vicino uscente j del nodo i riceve l'informazione di stato $x_i(t)$ e lo stato aumentato pesato $b_{ij}s_i(t)$.

3. Il nodo i che trasmette aggiorna il suo stato ed il suo stato aumentato come segue:

$$x_i(t+1) = x_i(t) + \epsilon s_i(t), \tag{8}$$

$$s_i(t+1) = \Big((1 - \sum_{j \in \mathcal{N}_i^-} b_{ij})s_i(t)\Big) - \Big(x_i(t+1) - x_i(t)\Big). \tag{9}$$

   I nodi $j \in \mathcal{N}_i^-$ che ricevono informazione aggiornano i loro stati e stati aumentati come segue:

$$x_j(t+1) = \frac{x_i(t) + x_j(t)}{2} + \epsilon s_j(t), \tag{10}$$

$$s_j(t+1) = s_j(t) + b_{ij}s_i(t) - \Big(x_j(t+1) - x_j(t)\Big). \tag{11}$$

I nodi non attivi, ossia ogni nodo $k \neq i$, $k \notin \mathcal{N}_i^-$ aggiornano i loro stati e stati aumentati come segue:

$$x_k(t+1) = x_k(t) + \epsilon s_k(t), \tag{12}$$

$$s_k(t+1) = s_k(t) - \Big(x_k(t+1) - x_k(t)\Big). \tag{13}$$

Si nota che il nodo i che trasmette aggiorna il suo stato sommandolo con il suo stato aumentato "pesato" dal parametro ϵ (posto pari a 0.2 come in AS), mentre i nodi riceventi j aggiornano il loro stato facendone la media con il valore ricevuto dal nodo i, e sommando poi il risultato con il loro stato aumentato "pesato" da ϵ. L'aggiornamento degli stati aumentati avviene in modo tale da preservare la media nella rete, tenendo conto del cambiamento di stato dei nodi che aggiornano.
La topologia delle interazioni all'istante t è rappresentata dal grafo $G_i(t)$, ottenuto da $G(t)$ rimuovendo tutti gli archi che non partono dal nodo i; quindi $G_i(t)$ rappresenta il grafo delle comunicazioni che avvengono al tempo t, ossia quando solo un certo nodo i trasmette informazione ai suoi vicini. Le matrici $A_i(t)$, $L_i(t)$, $B_i(t)$, $D_i(t)$, $\widetilde{D}_i(t)$ e $S_i(t)$ vengono calcolate in modo del tutto analogo a quanto fatto nell'algoritmo AS e permettono di arrivare ad una formulazione del tipo:

$$\begin{bmatrix} \mathbf{x}(t+1) \\ \mathbf{s}(t+1) \end{bmatrix} = M_i(t) \begin{bmatrix} \mathbf{x}(t) \\ \mathbf{s}(t) \end{bmatrix}.$$

Si evidenzia il fatto che la dinamica degli stati all'istante t dipende in modo sostanziale da quale nodo i viene selezionato per trasmettere.

3.4 Algoritmo Gossip (AG)

Viene ora presentato un altro algoritmo per reti asincrone che, a differenza del precedente, si basa su una comunicazione di tipo gossip: in ogni istante temporale si sveglia solo un agente, che comunica con un solo vicino scelto in modo casuale, e questi due nodi mediano i loro stati. Con questo algoritmo a stato aumentato gli agenti raggiungono l'average consensus sia in media quadratica che quasi certamente per ogni digrafo fortemente connesso. Si consideri una rete di n agenti modellata dal digrafo $G = (V, E)$; si supponga che ad ogni istante temporale si attivi esattamente un arco in E in modo casuale, indipendentemente da ciò che è accaduto negli istanti precedenti. Sia (i, j) l'arco attivatosi al tempo $t \in \mathbb{Z}_+$ con probabilità costante; l'informazione di stato $x_i(t)$ e lo stato aumentato $s_i(t)$ sono trasmessi lungo l'arco dal nodo i al nodo j. L'aggiornamento indotto è descritto come segue:

1. Per il nodo i che trasmette: $x_i(t+1) = x_i(t)$ e $s_i(t+1) = 0$.

2. Per il nodo j che riceve:

$$x_j(t+1) = x_j(t) + w_{ji}(x_i(t) - x_j(t)) + \epsilon w_{ji} s_j(t), \tag{14}$$

$$s_j(t+1) = s_j(t) + s_i(t) - (x_j(t+1) - x_j(t)), \tag{15}$$

   dove i pesi aggiornati $w_{ji} \in (0,1)$ e il parametro $\epsilon = 0.2$ come in AS e AB. In questo progetto si è scelto di porre $w_{ij} = 0.5$ come in [6].

3. Per gli altri nodi $l \in V - \{i,j\}$: $x_l(t+1) = x_l(t)$ e $s_l(t+1) = s_l(t)$.

Si evidenzia il fatto che nella rete, al tempo t, solo il nodo i trasmette informazione, e solo il nodo j riceve informazione ed esegue l'aggiornamento di stato. Per quanto riguarda le variabili di stato aumentato, si noti che il nodo i che trasmette pone sempre a zero il suo stato aumentato, la qual cosa implica che tutti i suoi stati aumentati sono stati trasmessi; da (15) si nota che il nodo ricevente j aggiorna il suo stato aumentato sommandolo con quello del nodo i e sottraendo il suo cambiamento di stato.
Sia A_{ij} la matrice delle adiacenze del digrafo $G_{ij} = (V, \{(i,j)\})$ data da $A_{ij} = w_{ji} f_j f_i^T$, dove f_i, f_j sono i vettori unità della base standard di $\mathbb{R}^n$. La matrice laplaciana L_{ij} è data da $L_{ij} = D_{ij} - A_{ij}$, dove $D_{ij} = w_{ji} f_j f_j^T$; quindi L_{ij} ha somme di riga pari a zero, e la matrice $I - L_{ij}$ è stocastica per righe. Si definisce $S_{ij} := I - (f_i - f_j) f_i^T$; ovviamente S_{ij} è stocastica per colonne.
Grazie alle matrici appena introdotte, l'aggiornamento degli stati e degli stati aumentati quando l'arco (i,j) viene attivato al tempo t può essere scritto in forma matriciale come segue:

$$\begin{bmatrix} \mathbf{x}(t+1) \\ \mathbf{s}(t+1) \end{bmatrix} = M(t) \begin{bmatrix} \mathbf{x}(t) \\ \mathbf{s}(t) \end{bmatrix}, \quad \text{dove}$$

$$M(t) = M_{ij} = \begin{bmatrix} I - L_{ij} & \epsilon D_{ij} \\ L_{ij} & S_{ji} - \epsilon D_{ij} \end{bmatrix}. \tag{16}$$

Ci sono alcune osservazioni da fare riguardo questo algoritmo: (i) la matrice $M(t)$ ha elementi negativi dovuti alla presenza della matrice laplaciana L_{ij} in posizione (2,1); (ii) le colonne di $M(t)$ sommano ad uno, quindi la quantità $x(t) + s(t)$ è costante per ogni t; (iii) dall'ipotesi sulla distribuzione di probabilità degli archi che si attivano, la sequenza $M(t)$, $t = 0, 1, \ldots$, è indipendente ed identicamente distribuita (i.i.d.).
Si dimostra che usando l'algoritmo (16) con il parametro ϵ sufficientemente piccolo, gli agenti della rete raggiungono l'average consensus sia in media quadratica che quasi certamente se e solo se il digrafo G è fortemente connesso (per maggiori dettagli si veda [6]).

3.5 Algoritmo Gossip basato su Broadcast (AGB)

Si consideri un sistema di n agenti rappresentato da un digrafo $G = (V, E)$; come nei precedenti algoritmi a stato aumentato, anche in questo algoritmo si associa ad ogni nodo i, per $i = 1, \ldots, n$, oltre allo stato $x_i(t)$, la variabile $s_i(t)$ che ha valore iniziale $s_i(0) = 0$; ogni agente della rete, in un determinato istante di tempo, o trasmette, o riceve, o è inattivo. L'algoritmo si basa su 3 regole fondamentali:

REGOLA 1: aggiornamento del nodo che trasmette, ossia il nodo i

$$\begin{cases} x_i(t+1) = x_i(t), \\ s_i(t+1) = 0. \end{cases} \tag{17}$$

REGOLA 2: aggiornamento del nodo che riceve, ossia ogni nodo $j \in \mathcal{N}_i$

$$\begin{cases} x_j(t+1) = \frac{x_j(t)+x_i(t)}{2} + 0.5 s_j(t) + \frac{s_i(t)}{2\delta_{out,i}(t)}, \\ s_j(t+1) = \frac{x_j(t)-x_i(t)}{2} + 0.5 s_j(t) + \frac{s_i(t)}{2\delta_{out,i}(t)}. \end{cases} \tag{18}$$

REGOLA 3: nodi non attivi, ossia ogni nodo $k \neq i$, $k \notin \mathcal{N}_i$

$$\begin{cases} x_k(t+1) = x_k(t), \\ s_k(t+1) = s_k(t). \end{cases} \tag{19}$$

Algoritmo:

1. Sia $t = 0$, $x(0) = x_0$ e $s(0) = 0$.
2. Un nodo i scelto in modo casuale applica la REGOLA 1.
3. Ogni nodo $j \in \mathcal{N}_i$ in ascolto esegue la REGOLA 2.
4. Ogni altro nodo $k \neq i, k \notin \mathcal{N}_i$ mantiene invariati il suo stato ed il suo stato aumentato (REGOLA 3).
5. Sia $t = t + 1$ e si torna al passo 2.

Si nota che il nodo i trasmette il suo stato x_i a tutti i suoi vicini e nel farlo conosce il suo numero di archi uscenti; inoltre trasmette il valore $s_i(t)/\delta_{out,i}(t)$, ossia lo stato aumentato diviso per il numero di nodi che ricevono l'informazione; il nodo i non cambia il suo stato $x_i(t)$, mentre resetta a 0 la variabile $s_i(t)$. I nodi riceventi aggiornano il loro stato $x_j(t)$ facendo la media fra il loro stato e il valore ricevuto e correggendo l'aggiornamento con una frazione

della loro variabile $s_j(t)$ e una frazione della variabile del nodo che trasmette. L'aggiornamento della variabile di stato aumentato viene fatto tramite l'aggiunta di alcuni termini, al fine di preservare la media nella rete ad ogni iterazione mentre si converge al valore medio delle misure iniziali.
La sequenza di nodi che trasmette ai diversi istanti temporali $t \in \mathbb{Z}_+$ definisce il segnale $\mathcal{I}(t)$; assumendo che all'istante t il nodo $i = \mathcal{I}(t)$ trasmetta, la topologia delle interazioni è rappresentata dal grafo $G_i(t)$, ottenuto da $G(t)$ rimuovendo tutti gli archi che non partono dal nodo i. Siano $A_i(t), \Delta_{in,i}(t)$ e $L_i(t)$ rispettivamente la matrice di adiacenza, la matrice degli archi entranti e la laplaciana del grafo $G_i(t)$. Si definiscono

$$P_i(t) = I - 0.5L_i(t), \qquad \hat{\Gamma}_i(t) = \frac{A_i(t)}{2\delta_{out,i}(t)} + 0.5\Delta_{in}(t),$$

$$\Gamma_i(t) = \frac{A_i(t)}{2\delta_{out,i}(t)} - 0.5\Delta_{in}(t) + (I - \mathbf{e}_i\mathbf{e}_i^T),$$

dove I è la matrice identità ed e_i è l'i-esimo vettore canonico di dimensione n. Si pone

$$C_i(t) = \begin{bmatrix} P_i(t) & \hat{\Gamma}_i(t) \\ I - P_i(t) & \Gamma_i(t) \end{bmatrix}. \tag{20}$$

Facendo riferimento alle regole di aggiornamento di stato (17), (18) e (19), la dinamica del sistema all'istante t risulta essere:

$$\begin{bmatrix} \mathbf{x}(t+1) \\ \mathbf{s}(t+1) \end{bmatrix} = C_i(t) \begin{bmatrix} \mathbf{x}(t) \\ \mathbf{s}(t) \end{bmatrix}, \quad i = \mathcal{I}(t). \tag{21}$$

Si evidenzia il fatto che questo algoritmo, a differenza di altri algoritmi broadcast di average consensus, assicura il fatto che si converge esattamente alla media degli stati iniziali, e non a qualche altro valore determinato dalle condizioni iniziali e dall'evoluzione del sistema. Inoltre, assunto che la rete di sensori sia distribuita nello spazio, ogni coppia di nodi in cui un agente sia sufficientemente distante dall'altro può avviare una comunicazione di tipo broadcast, senza che la trasmissione di uno dei due nodi interferisca con quella dell'altro; questa sorta di parallelismo intrinseco della rete è propriamente sfruttato dall'algoritmo e ne migliora sensibilmente il tempo di convergenza (per un'analisi dettagliata delle proprietà di convergenza si veda [5]).

4 Introduzione perdita di pacchetto

In questa sezione si spiega come è stata introdotta la perdita di pacchetto negli algoritmi AS, AB, AG e AGB; l'algoritmo ASP nella sua versione originale prevedeva già il caso che vi potesse essere perdita di dati. In tutti i quattro casi la perdita d'informazione è stata implementata in modo tale che ogni trasmissione da un nodo ad un altro abbia una certa probabilità (supposta uniforme, quindi la stessa per ogni comunicazione) di perdità di dati (modellata con una variabile aleatoria binomiale). E' doveroso sottolineare che nell'introduzione della perdita di pacchetto si è scelto di modificare il meno possibile gli algoritmi originali; con ogni probabilità se si fossero modificati maggiormente gli algoritmi si sarebbero ottenuti risultati migliori nella fase di simulazione, tuttavia sarebbe venuto meno lo scopo di questo lavoro, che si ricorda essere quello di valutare le prestazioni di alcuni algoritmi dopo aver introdotto la possibilità che vi sia perdita d'informazione senza snaturarne l'essenza.

4.1 AS

Per implementare la perdita di pacchetto in questo algoritmo sincrono sono state introdotte le variabili aleatorie binomiali $p_{ij}(t)$, che rappresentano la probabilità che la trasmissione dal nodo i al nodo j abbia avuto successo; dunque $p_{ij}(t) = 1$ se non c'è stata perdita di dati, mentre $p_{ij}(t) = 0$ se l'informazione è andata persa. L'aggiornamento del nodo i avviene come segue:

$$x_i(t+1) = x_i(t) + \sum_{j \in \mathcal{N}_i^+} a_{ij} p_{ji}(t)(x_j(t) - x_i(t)) + \epsilon s_i(t), \tag{22}$$

$$s_i(t+1) = \Big((1 - \sum_{h \in \mathcal{N}_i^-} b_{ih}) s_i(t) + \sum_{j \in \mathcal{N}_i^+} b_{ji} p_{ji}(t) s_j(t)\Big) - \Big(x_i(t+1) - x_i(t)\Big). \tag{23}$$

Per attuare questa modifica si è scelto di moltiplicare le matrici A e B^T dell'algoritmo originale per una matrice $n \times n$ di variabili aleatorie binomiali. Si nota che qualora tutte le comunicazioni vadano a buon fine (i.e. $p_{ij}(t) = 1$, per ogni (i, j) e per ogni t) l'aggiornamento avviene in modo identico a quello dell'algoritmo senza perdita di pacchetto (descritto dalle equazioni (4) e (5)).

4.2 AB

In questo algoritmo asincrono con comunicazione di tipo broadcast, l'aggiornamento del nodo che trasmette e dei nodi inattivi, nel caso vi sia perdita di

pacchetto, rimane lo stesso rispetto all'algoritmo originale (secondo le equazioni (8), (9), (12) e (13)); a cambiare è l'aggiornamento dei nodi che ricevono e si distinguono due casi:

- Nel primo caso, qualora la trasmissione fallisca, i nodi che avrebbero dovuto ricevere sono a conoscenza del tentativo di trasmettere (caso *consapevole*); l'aggiornamento dello stato avviene come nell'algoritmo originale a meno del dato $x_i(t)$ che non è stato ricevuto, mentre l'aggiornamento dello stato aumentato avviene allo stesso modo dei nodi isolati:

$$x_j(t+1) = \frac{x_j(t)}{2} + \epsilon s_j(t), \tag{24}$$

$$s_j(t+1) = s_j(t) - \Big(x_j(t+1) - x_j(t)\Big). \tag{25}$$

  Per attuare questa modifica si è scelto di moltiplicare gli elementi dell'i-esima colonna della matrice $A_i(t)$ e dell'i-esima riga di $B_i(t)$ per un vettore colonna i cui elementi sono variabili aleatorie binomiali che rappresentano la probabilità di successo nella trasmissione dei dati; in questo modo le matrici $A_i(t)$ e $B_i(t)$ sono state modificate solo in corrispondenza alle comunicazioni che avvengono all'istante t, ossia le trasmissioni di dati dal nodo i ai suoi vicini.

- Nel secondo caso invece, qualora la trasmissione fallisca, i nodi che avrebbero dovuto ricevere non sono a conoscenza del tentativo di trasmettere (caso *inconsapevole*) e si comportano come nodi inattivi, quindi aggiornano i loro stati come nelle equazioni (12) e (13):

$$x_j(t+1) = x_j(t) + \epsilon s_j(t),$$

$$s_j(t+1) = s_j(t) - \Big(x_j(t+1) - x_j(t)\Big).$$

  Questa modifica rispetto all'algoritmo originale è stata ottenuta in modo del tutto analogo a quanto spiegato nel caso precedente.

Ovviamente, qualora la trasmissione vada a buon fine e quindi non vi sia perdita d'informazione, anche l'aggiornamento dei nodi riceventi avviene come nell'algoritmo originale, quindi tramite (10) e (11).

4.3 AG

In questo algoritmo asincrono con comunicazione di tipo gossip, per implementare la perdita di pacchetto ci si serve, come già fatto nell'algoritmo AS,

di variabili aleatorie binomiali $p_{ij}(t)$, che rappresentano la probabilità che la trasmissione dal nodo i al nodo j abbia avuto successo. L'aggiornamento del nodo che trasmette e dei nodi inattivi avviene come nell'algoritmo originale, mentre cambia l'aggiornamento del nodo ricevente:

1. Per il nodo i che trasmette: $x_i(t+1) = x_i(t)$ e $s_i(t+1) = 0$.

2. Per il nodo j che riceve:

$$x_j(t+1) = x_j(t) + w_{ji}(x_i(t)p_{ij}(t) - x_j(t)) + \epsilon w_{ji}s_j(t), \tag{26}$$

$$s_j(t+1) = s_j(t) + s_i(t)p_{ij}(t) - (x_j(t+1) - x_j(t)), \tag{27}$$

   dove i pesi aggiornati $w_{ji} \in (0,1)$ e il parametro $\epsilon > 0$; come per l'algotitmo originale si è scelto $w_{ij} = 0.5$ ed $\epsilon = 0.2$.
   Questa modifica è stata attuata moltiplicando i vettori unità f_i per la probabilità di successo della trasmissione.
   Si nota che anche in questo caso, qualora ogni trasmissione vada a buon fine, i.e. $p_{ij}(t) = 1$, per ogni (i,j) e per ogni t, l'aggiornamento avviene come in (14) e (15).

3. Per gli altri nodi $l \in V - \{i,j\}$: $x_l(t+1) = x_l(t)$ e $s_l(t+1) = s_l(t)$.

4.4 AGB

L'AGB è un algoritmo di gossip esteso basato su comunicazione di tipo broadcast; come per l'algoritmo AB, l'aggiornamento dei nodi che trasmettono e di quelli inattivi non cambia rispetto all'algoritmo originale (ed avviene quindi secondo le equazioni (17) e (19)), mentre a cambiare è l'aggiornamento dei nodi riceventi; sono stati analizzati due casi:

- Nel primo caso, qualora la trasmissione fallisca, chi avrebbe dovuto ricevere sa che si è provato a trasmettere (caso *consapevole*) e quindi l'aggiornamento dello stato e dello stato aumentato avviene in modo analogo all'algoritmo originale ad eccezione dei dati persi relativi al nodo che trasmette:

$$\begin{cases} x_j(t+1) = \frac{x_j(t)+s_j(t)}{2}, \\ s_j(t+1) = \frac{x_j(t)+s_j(t)}{2}. \end{cases} \tag{28}$$

  Questa modifica è stata implementata in modo similare a quanto fatto nell'algoritmo AB: gli elementi dell'i-esima colonna della matrice $A_i(t)$ sono stati moltiplicati per le rispettive probabilità di successo

della comunicazione. Si sottolinea inoltre che la matrice $\Delta_{in}(t)$, che rappresenta il numero di archi entranti di ogni nodo, è stata calcolata dalla matrice di adiacenza dell'algoritmo originale, a differenza del caso successivo.

- Nel secondo caso, qualora la trasmissione fallisca, chi avrebbe dovuto ricevere non sa che si è provato a trasmettere (caso *inconsapevole*); l'aggiornamento dei nodi che avrebbero dovuto ricevere avviene allora allo stesso modo dei nodi isolati:

$$\begin{cases} x_j(t+1) = x_j(t), \\ s_j(t+1) = s_j(t). \end{cases} \tag{29}$$

  Anche in questo caso gli elementi dell'i-esima colonna della matrice $A_i(t)$ sono stati moltiplicati per le rispettive variabili aleatorie binomiali; tuttavia, a differenza del caso precedente, la matrice $\Delta_{in}(t)$ è stata calcolata servendosi della nuova matrice delle adiacenze appena ottenuta.

Ovviamente, qualora la trasmissione vada a buon fine e quindi non vi sia perdita d'informazione, anche l'aggiornamento dei nodi riceventi avviene come nell'algoritmo originale (quindi tramite (18)).

5 Simulazioni

Nella seguente sezione saranno presentati i risultati delle simulazioni degli algoritmi con perdita di pacchetto; in particolare, gli algoritmi verranno confrontati a seconda della modalità della rete (sincrona o asincrona) e della tipologia di comunicazione (broadcast o gossip). Le prestazioni vengono valutate in termini di errore e velocità di convergenza al variare della probabilità di perdita d'informazione (si premette che in tutti i grafici che saranno presentati, il valore p rappresenta la probabilità che la comunicazione vada a buon fine, quindi la probabilità di non perdere informazione), del numero di nodi (per un massimo di cinquanta nodi) e del numero di iterazioni (per un massimo di seimila iterazioni). Per errore di convergenza si intende la differenza fra la media degli stati iniziali dei vari nodi e la media degli stati al tempo di convergenza:

$$errore = \frac{1}{n} \left\| \mathbf{1}^T x(0) - \mathbf{1}^T x(\infty) \right\|_2 \text{ con } x(t) = [x_1(t), \ldots, x_n(t)]^T, \qquad (30)$$

dove n è il numero di nodi e $\mathbf{1} = (1, \ldots, 1)^T$. L'errore di convergenza dipende da diversi fattori:

- maggiore è la probabilità $1 - p$ di perdere pacchetti, maggiore sarà l'informazione persa e quindi l'errore di convergenza;
- in una rete con molte comunicazioni circola una grande quantità di dati istante per istante; questo implica che più una rete è connessa, maggiori sono le informazioni che rischiano di essere perse in un dato tempo t; in linea di principio questo fatto dovrebbe portare ad ipotizzare che gli algoritmi sincroni (quindi molto connessi) abbiano prestazioni peggiori rispetto agli asincroni; tuttavia, a seconda delle diverse possibili reazioni alla pardita di pacchetto, si vedranno sia algoritmi sincroni con errore molto basso che algoritmi sincroni con errore più elevato.

Per quanto riguarda la velocità di convergenza, si è scelto di considerare come tempo di convergenza T_{conv} l'istante temporale t in cui:

$$\left\| x(t) - \frac{\mathbf{1}\mathbf{1}^T}{n} x(t) \right\|_2 < \mu, \qquad (31)$$

dove si è deciso di prendere μ pari ad 0.5 (valore di compromesso con il quale si hanno buoni risultati sia in termini errore di convergenza che in termini di tempo necessario a compiere un numero consistente di realizzazioni al fine di avere risultati significativi). La scelta di utilizzare come secondo termine

della sottrazione la media degli stati all'istante t e non la media iniziale (come talvolta avviene) è dovuta al fatto che l'introduzione della perdita di pacchetto porta ad avere un errore di convergenza in generale diverso da zero, quindi l'average consensus non è più assicurato. Per come sono stati definiti, l'errore e la velocità di convergenza possono essere pensati come due parametri che permettono di analizzare le prestazioni di un algoritmo rispettivamente "a regime", ovvero studiando il valore a cui (eventualmente) converge lo stato, e "in transitorio", ossia valutando la velocità con la quale viene raggiunto il valore di regime.
Per le simulazioni saranno usati i random geometric graph (grafi geometrici casuali): sono grafi casuali costruiti posizionando i vertici in modo aleatorio, uniforme ed indipendente, e connettendo due vertici, u e v, se e solo se la distanza fra loro non supera una certa soglia r (che verrà indicato come *raggio di vicinanza*), ossia $d(u, v) \leq r$ (se $r = 1$ ogni nodo $j \neq i$ è vicino del nodo i). Si evidenzia il fatto che questo tipo di struttura ha una forte componente aleatoria che influenza in modo significativo la dinamica degli algoritmi: a seconda ad esempio della disposizione dei nodi, le prestazioni dell'AGB possono cambiare in modo radicale. Per tentare di ridurre l'incidenza dell'aleatorietà dei grafi utilizzati, si è scelto di ripetere più volte la simulazione di ogni algoritmo (sono state effettuate cento prove), valutandone poi la media, e di non tenere conto di quelle realizzazioni in cui il grafo presentava gruppi di nodi isolati che impedivano la convergenza dello stato ad un singolo valore dopo seimila iterazioni.

5.1 Algoritmi Sincroni

In questa sezione vengono analizzate le prestazioni dei due algoritmi per reti sincrone: l'ASP e l'AS nella sua versione con perdita di pacchetto.
Si comincia con l'ASP, l'unico algoritmo che già nella sua forma originale aveva implementata la perdita di pacchetto; si ricorda che di questo algoritmo sono presenti due varianti: il metodo della compensazione parziale (biased) e il metodo della compensazione bilanciata (balanced); in Figura 2 e Figura 3 sono presenti gli andamenti dell'errore e del tempo di convergenza dei metodi della compensazione parziale e bilanciata rispettivamente in funzione della probabilità p che la trasmissione abbia successo (con $r = 1$ costante) ed in funzione del raggio di vicinanza (con $p = 0.4$ costante); per quanto riguarda l'errore di convergenza si nota che più la probabilità di successo è alta, più l'errore è basso, quindi, come è lecito aspettarsi, meno pacchetti si perdono, minore è l'errore, mentre la dipendenza dal raggio di vicinanza è più complessa: dalle Figure 3(a) e 3(c) si può notare che gli errori dei due metodi con raggio pari ad uno risultano molto simili, mentre quelli con

raggio più basso presentano differenze maggiori: nell'ASP parziale si cerca di compensare la perdita di dati con l'informazione in possesso dello stato ricevente all'istante precedente; questa gestione della perdita di pacchetto risulta efficace soprattutto nei casi in cui il raggio di vicinanza è molto piccolo, in quanto i valori dei due stati sono simili e quindi l'errore che si compie è di scarsa rilevanza; al contrario l'ASP bilanciato cerca di spalmare su tutti gli stati l'errore dovuto alla perdita d'informazione, facendo sì che per reti con raggio di vicinanza elevato (ad esempio con $r = 1$) l'errore relativo ai dati persi abbia una scarsa incidenza sull'errore finale, mentre per reti con raggio di vicinanza basso ogni perdita di informazione ha rilevanza maggiore.
Per quanto riguarda la velocità di convergenza, si può notare che, come ci si aspetta a priori, è maggiore (quindi si hanno prestazioni migliori) al crescere della probabilità di successo p e del raggio di vicinanza r; inoltre si evidenzia che l'ASP bilanciato converge più velocemente dell'ASP parziale (sia con r costante che con p costante): mentre nel bilanciato i vari agenti tendono a muoversi verso il baricentro pesato di tutti i vicini, nel parziale il movimento è meno drastico e risente fortemente dei pesi associati agli archi.
Si confrontano ora gli errori di convergenza dell'ASP e dell'AS nella sua versione con perdita di pacchetto, per determinati valori di r e p.

Nodi	ASP parziale	ASP bilanciato	AS
5	4.227	9.19863	50.8068
10	1.96352	4.07714	50.1957
15	1.24001	2.54749	50.1164
20	0.953461	1.52494	50.2019
25	0.885371	1.27471	49.4835
30	0.70758	0.988101	50.0245
35	0.567842	0.977046	49.6724
40	0.527323	0.707667	49.9628
45	0.436022	0.677762	49.6241
50	0.405194	0.631278	49.9521

Tabella 1: Errore di convergenza algoritmi sincroni con $r = 1$ e $p = 0.4$.

Si nota dalle Tabelle 1 e 2 che l'AS ha errore di convergenza enorme rispetto all'ASP; a tal proposito è doveroso sottolineare che sebbene entrambi gli algoritmi siano della stessa natura (sono pensati per reti sincrone), il loro modus operandi realtivo alla perdita di pacchetto è totalmente diverso: l'ASP gestisce l'informazione persa $x_i(t)$ o sostituendola con l'informazione relativa allo stesso nodo i all'istante precedente (compensazione parziale) che ne prende il posto nell'equazione di aggiornamento (e viene dunque sommata agli altri termini) o distribuendola fra tutti i vicini (compensazione bilanciata),

Nodi	ASP parziale	ASP bilanciato	AS
5	2.27447	3.05841	49.3244
10	1.14374	1.34029	51.123
15	0.737217	1.04853	49.632
20	0.580298	0.764593	49.2676
25	0.450678	0.635754	48.9913
30	0.376165	0.494718	48.9195
35	0.349976	0.457199	49.5952
40	0.287984	0.479261	49.2755
45	0.247911	0.421956	48.6604
50	0.231825	0.392276	49.0083

Tabella 2: Errore di convergenza algoritmi sincroni con $r = 0.4$ e $p = 0.8$.

mentre nell'AS la perdita dell'informazione inviata, $x_i(t)$ e $s_i(t)$, non viene compensata: lo stato del nodo ricevente j all'istante $t+1$ tende a stare fermo senza correggere la propria posizione, mentre il valore dello stato aumentato dello stesso nodo viene falsato, quindi la somma $x(t)+s(t)$ non è più costante, a causa dell'errore introdotto su $s_j(t)$. Ciò fa precipitare la somma dello stato verso lo zero, come si può vedere in Figura 1. Si può quindi affermare che l'algoritmo ASP cerca di "recuperare" la perdita d'informazione, mentre l'algoritmo AS "dimentica" il dato perso; questo diverso modo di affrontare la perdita di pacchetto porta a errori di convergenza profondamente diversi: mentre gli agenti dei due metodi dell'ASP si muovono in direzione di quella che pensano essere la media della rete, i nodi dell'AS che non ricevono informazione inizialmente rimangono quasi fermi dove sono per poi muoversi verso lo zero; ciò fa sì che l'errore dell'algoritmo ASP risulti molto più contenuto rispetto all'errore dell'AS.
Per quanto riguarda la velocità di convergenza dell'algoritmo AS, si può affermare aumenta all'aumentare della probabilità di perdere il pacchetto, quindi al diminuire della probabilità di successo, come si può vedere da Figura 4; questo comportamento, che potrebbe sembrare contrario alle aspettative e che invece si ritroverà anche negli algoritmi successivi, è dovuto alla diversa reazione alla perdita di pacchetto: come già spiegato in precedenza, in questo algoritmo la perdita di dati non genera un movimento degli agenti volto alla compensazione, movimento che andrebbe a rallentare la velocità di convergenza; al contrario i vari nodi si dimenticano dell'informazione persa; pertanto la quantità d'informazione globale della rete diminuisce, rendendo più veloce il processo di convergenza. Si nota inoltre che la velocità di convergenza è maggiore quando il raggio di vicinanza è più ampio: in questo modo ci sono più comunicazioni e più nodi hanno a disposizione molti da-

ti, riuscendo così a convergere in tempo minore; questo concetto potrebbe sembrare antitetico a quanto appena affermato riguardo la dipendenza della velocità di convergenza dalla probabilità di successo della trasmissione; tuttavia le due tesi sono diverse: mentre nel primo caso si è fatto riferimento all'informazione globale presente sulla rete, rilevando che al decrescere della stessa la velocità di convergenza risulta maggiore, nel secondo caso è stata considerata l'informazione in possesso di ogni singolo agente.

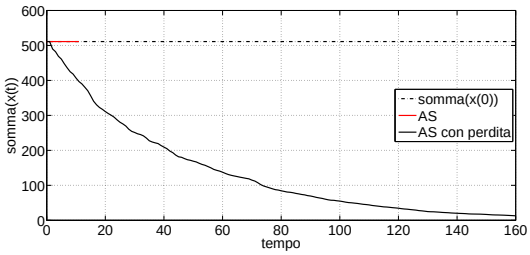
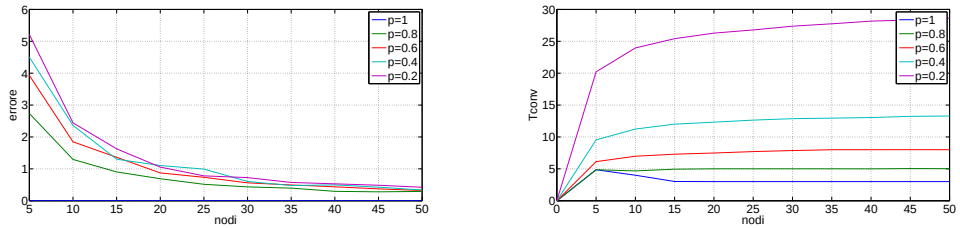
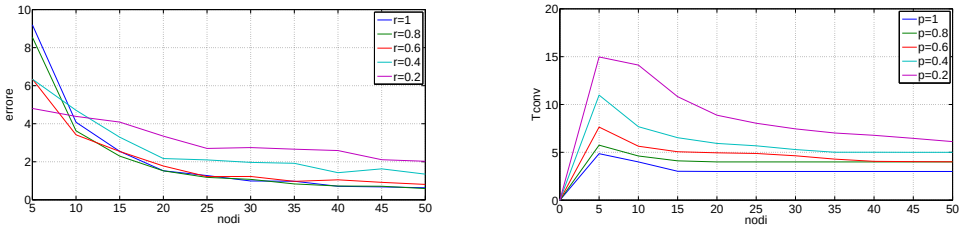


Figura 1: Grafo con 10 nodi, $r = 1$ e $p = 0.9$.



(a) Errore ASP parziale

(b) T_{conv} ASP parziale



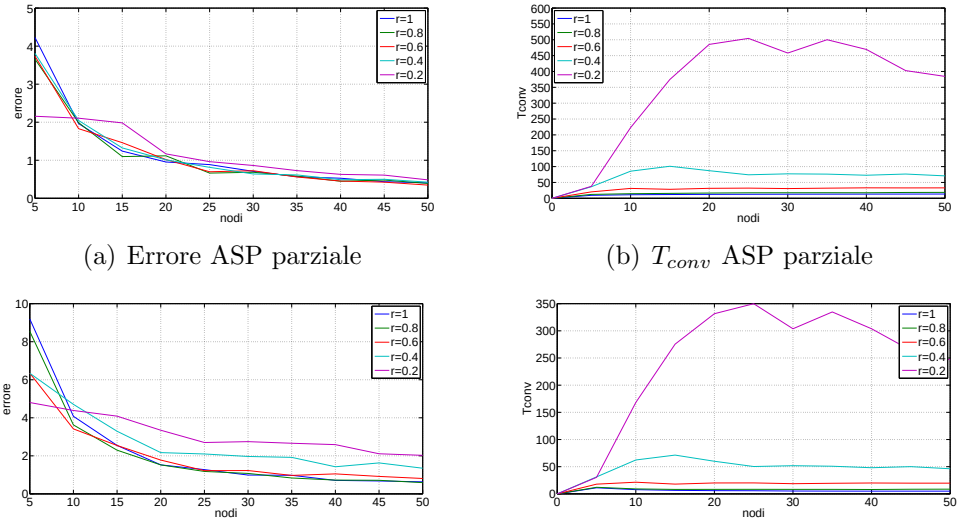
(c) Errore ASP bilanciato

(d) T_{conv} ASP bilanciato

Figura 2: ASP con $r = 1$.

5.2 Algoritmi Asincroni

In questa sezione vengono illustrate e messe a confronto le prestazioni dei tre algoritmi asincroni nelle loro versioni con perdita di pacchetto: l'AB, l'AG



(a) Errore ASP parziale

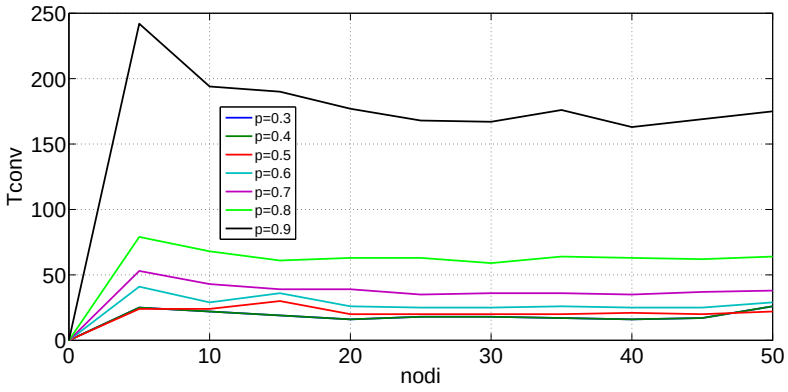
(b) T_{conv} ASP parziale

(c) Errore ASP bilanciato

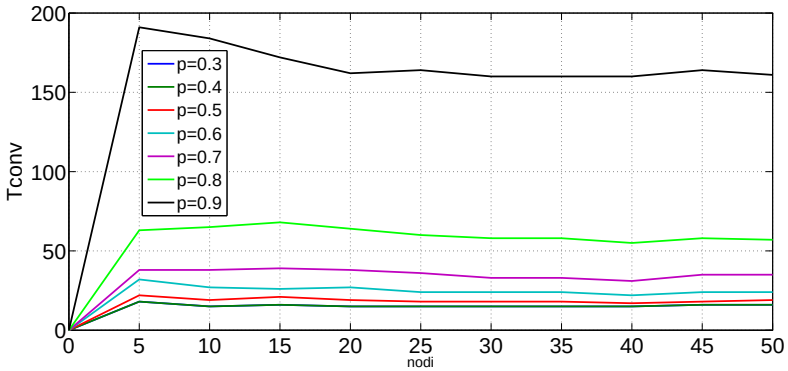
(d) T_{conv} ASP bilanciato

Figura 3: ASP con $p = 0.4$.

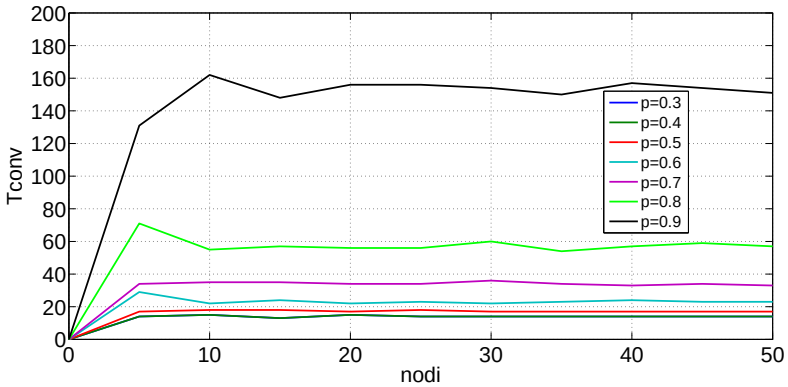
e l'AGB. Si ricorda che gli algoritmi AB e AGB sono basati su una comunicazione di tipo broadcast, l'algoritmo AG su una di tipo gossip, quindi il confronto viene spontaneo fra AB e AGB, mentre verranno introdotti degli accorgimenti particolari per fare in modo che anche l'algoritmo AG possa essere paragonato agli altri due.
Si comincia prendendo in esame i due algoritmi broadcast; al fine di rendere la valutazione il più possibile efficace, le simulazioni sono state fatte utilizzando lo stesso grafo e facendo accendere lo stesso nodo allo stesso istante; in altre parole è stata forzata la stessa dinamica, quindi il nodo che trasmette e quelli che ricevono sono i medesimi non solo all'istante iniziale, ma anche agli istanti successivi; inoltre si è fatto in modo che fossero corrotte le stesse comunicazioni, quindi che si perdessero esattamente gli stessi pacchetti. E' doveroso sottolineare che il valore di convergenza cambia a seconda che gli algoritmi siano consapevoli o inconsapevoli: mentre gli algoritmi inconsapevoli convergono ad un valore (in generale diverso dalla media iniziale e diverso da zero) che dipende da vari fattori, gli algoritmi consapevoli convergono sempre a zero, quindi l'errore di convergenza degli algoritmi inconsapevoli risulta essere molto minore rispetto a quello degli algoritmi consapevoli, come si può notare dalle Tabelle 3 e 4, in cui sono riportati gli errori dei vari algoritmi per determinati valori di raggio di vicinanza e probabilità di successo (rispettivamente per $r = 0.6$ e $p = 0.3$ e per $r = 0.9$ e $p = 0.4$); questa considerevole differenza nell'errore di convergenza fra algoritmi consapevoli e inconsapevoli



(a) $r = 0.5$



(b) $r = 0.7$



(c) $r = 1$

Figura 4: Velocità di convergenza AS.

si spiega con la struttura delle equazioni di aggiornamento: mentre nel caso inconsapevole lo stato all'istante $t+1$ si aggiorna considerando l'intero stato all'istante precedente, in entrambi gli algoritmi consapevoli l'aggiornamento dello stato all'istante $t+1$ avviene dimezzando lo stato all'istante t, e questo porta la media degli stati a convergere gradualmente a zero; ciò è ben visibile ad esempio nell'algoritmo AB, nel quale l'equazione di aggiornamento dello stato aumentato è la stessa sia per il caso consapevole che per il caso inconsapevole, mentre l'equazione di aggiornamento dello stato (non aumentato) nel caso consapevole risulta essere (24), in cui ad ogni iterazione si considera $x_j(t)/2$; ovviamente continui dimezzamenti di stato portano ad un valore nullo, con una velocità che risulta maggiore al diminuire della probabilità p che la trasmisione vada a buon fine; ossia se la probabilità di perdita di pacchetto $1-p$ è maggiore, il tempo di convergenza T_{conv} è minore, e quindi la velocità di convergenza risulta essere maggiore. Questo comportamento è dovuto al fatto che la perdita di dati coincide con dimezzamenti di stato; pertanto più informazioni si perdono istante per istante, maggiore è la velocità con cui si converge a zero, come si nota in Figura 5: il tempo di convergenza degli algoritmi consapevoli aumenta all'aumentare della probabilità di successo e quindi al diminuire della frequenza con cui si perdono i pacchetti. Nelle Tabelle 5 e 6 vengono riportati i valori del tempo di convergenza per determinati valori di r e p (rispettivamente per $r = 0.6$ e $p = 0.3$ e per $r = 0.9$ e $p = 0.4$).

Nodi	AB incons	AB cons	AGB cons	AGB incons
5	3.83925	36.6317	36.5767	5.00535
10	5.0101	49.6073	49.5928	6.11589
15	3.13393	64.9368	64.9254	3.59227
20	2.18521	39.998	39.9868	2.62573
25	1.96511	44.1988	44.1857	2.04788
30	1.88671	49.7668	49.7552	2.31478
35	2.0325	44.2226	44.216	2.36736
40	1.94613	49.9449	49.9323	2.33459
45	1.48397	42.3727	42.3618	1.48008
50	1.97658	45.4146	45.4018	2.33264

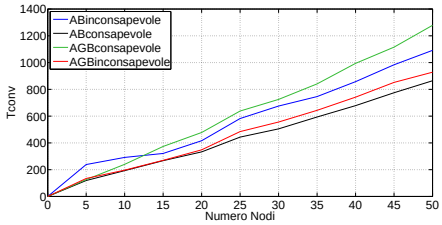
Tabella 3: Errore di convergenza algoritmi broadcast con $r = 0.6$ e $p = 0.3$.

Le scarse prestazioni in termini di errore di convergenza degli algoritmi consapevoli fanno sì che nell'analisi in funzione della velocità di convergenza si valutino esclusivamente i due algoritmi inconsapevoli. Si può notare dai grafici di Figura 6 che con una probabilità di successo abbastanza elevata ($p = 0.7$), l'algotitmo AGB inconsapevole, anche con l'introduzione della

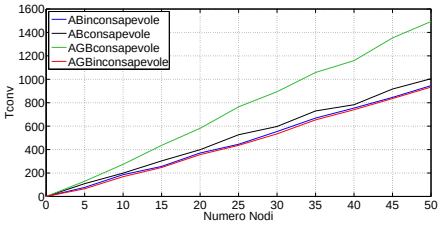
Nodi	AB incons	AB cons	AGB cons	AGB incons
5	4.67692	38.7953	38.7012	5.45361
10	4.27183	40.3908	40.3619	4.40525
15	2.14488	55.7254	55.6994	2.40718
20	2.16619	45.2368	45.218	2.30779
25	1.84091	65.7071	65.6858	1.86899
30	1.96332	56.2532	56.2353	1.8582
35	1.50305	55.4439	55.4259	1.50322
40	1.36676	45.5347	45.5173	1.37767
45	1.19348	54.0322	54.0135	1.25082
50	1.23485	48.1655	48.1516	1.12043

Tabella 4: Errore di convergenza algoritmi broadcast con $r = 0.9$ e $p = 0.4$.

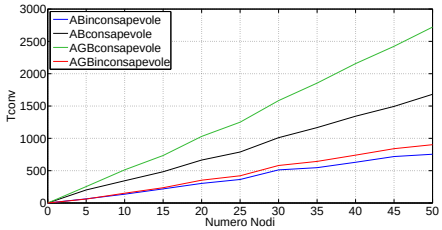
perdita di pacchetto, conferma di avere velocità di convergenza maggiore rispetto all'algoritmo AB inconsapevole nel caso di reti con raggio di vicinanza basso (quindi per $r = 0.4$ e $r = 0.5$), mentre si verifica il contrario per grafi con raggio di vicinanza più elevato ($r = 0.7$ e $r = 0.9$); questo comportamento si spiega osservando le regole di aggiornamento dei nodi riceventi nei due diversi algoritmi: a differenza dell'AB inconsapevole (formula (10)), nel quale si "pesa" lo stato aumentato con il parametro ϵ (che si ricorda essere pari a 0.2), nell'algoritmo AGB i nodi riceventi aggiornano il loro stato tramite l'equazione (18), nella quale il valore dello stato aumentato del nodo i che trasmette viene diviso per il doppio del numero di vicini uscenti $\delta_{out,i}(t)$ dello stesso nodo, quindi maggiore è il numero di vicini, più lo stato aumentato $s_i(t)$ viene spalmato fra i vari agenti e conseguentemente la velocità di convergenza decresce; ovviamente si è fatto riferimento alle equazioni di aggiornamento degli algoritmi originali (senza perdita di pacchetto) in quanto la probabilità di successo considerata è in questo caso abbastanza alta; per valori minori di p è opportuno considerare le formule degli algoritmi con perdita di pacchetto, quindi ad esempio l'aggiornamento dei nodi riceventi dell'AGB inconsapevole avverrà più frequentemente secondo (29), in cui non compare il termine $\delta_{out,i}(t)$ relativo ai vicini uscenti, e di conseguenza il raggio di vicinanza perderà d'importanza nella valutazione della velocità di convergenza degli algoritmi broadcast inconsapevoli.
Ora vengono analizzate le prestazioni dell'algoritmo AG rispetto agli algoritmi broadcast inconsapevoli AB e AGB; ancora una volta i casi consapevoli non sono stati presi in considerazione nel raffronto perchè convergono a zero, a differenza dei suddetti algoritmi che hanno valori di convergenza simili. Come nella sezione precedente in cui sono stati paragonati gli algoritmi AB e AGB, anche in questo caso si è cercato per quanto possibile di eseguire il con-



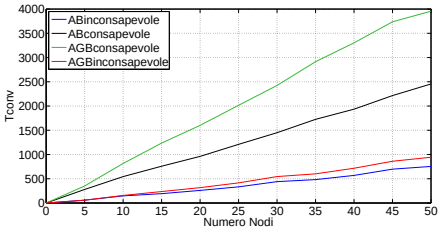
(a) $p = 0.3$



(b) $p = 0.4$

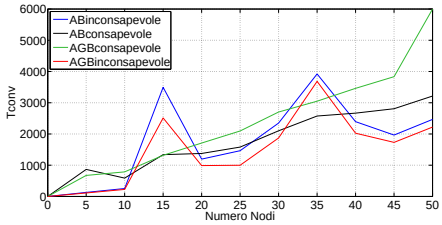


(c) $p = 0.6$

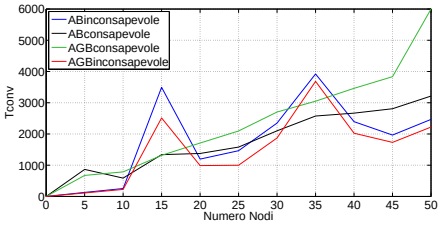


(d) $p = 0.7$

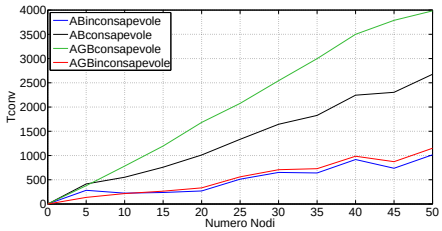
Figura 5: Tempo di convergenza algoritmi broadcast ($r = 0.9$ in ogni figura).



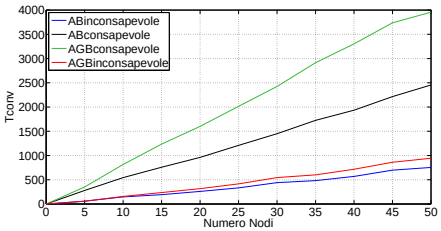
(a) $r = 0.4$



(b) $r = 0.5$



(c) $r = 0.7$



(d) $r = 0.9$

Figura 6: Tempo di convergenza algoritmi broadcast ($p = 0.7$ in ogni figura).

Nodi	AB incons	AB cons	AGB cons	AGB incons
5	103.63	86.7	100.85	76.96
10	691.26	210.52	235.86	330.4
15	532.18	288.56	377.06	333.51
20	589.09	346.06	475.05	400.11
25	705.04	436.52	602.31	505.8
30	1011.37	592.29	788.72	667.21
35	1122.59	618.21	851.96	742.15
40	1418.58	740.65	973.13	910.47
45	1302.28	777.34	1083.72	913.76
50	2051.7	950.78	1207.34	1213.75

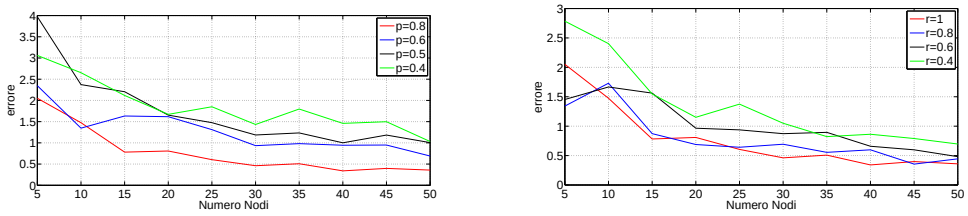
Tabella 5: Velocità di convergenza algoritmi broadcast con $r = 0.6$ e $p = 0.3$.

Nodi	AB incons	AB cons	AGB cons	AGB incons
5	77.29	109.03	129.79	65.62
10	185.92	199.26	273.95	169
15	256.07	303.29	436.2	247.08
20	370.98	398.79	581.58	357.2
25	445.34	526.59	764.67	436.2
30	554.47	597.15	893.96	534.38
35	668.9	729.93	1058.2	653.38
40	754.51	782.84	1159.64	740.33
45	845.64	916.91	1352.9	835.75
50	945.75	1004.75	1493.24	933.41

Tabella 6: Velocità di convergenza algoritmi broadcast con $r = 0.9$ e $p = 0.4$.

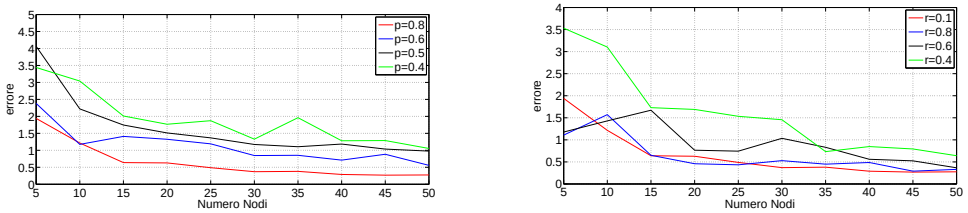
fronto facendo in modo che le dinamiche dei vari algoritmi fossero le stesse, utilizzando gli stessi grafi; tuttavia la presenza di tipologie di comunicazione diverse (si ricorda che AG è basato su comunicazione gossip) ha reso la comparazione un po' più complicata: nell'algoritmo AG si è scelto di attivare, ad ogni istante temporale t, lo stesso nodo i che trasmette nei due algoritmi di broadcast; il nodo ricevente è stato scelto con probabilità uniforme fra quelli che sono i vicini di i in AB e AGB; la riuscita della comunicazione viene gestita allo stesso modo degli algoritmi broadcast; in questo modo la dinamica di AG è stata forzata basandosi su quanto avviene in AB e AGB. Si sottolinea che ciò è stato possibile fino a che i due algoritmi broadcast, più veloci di quello gossip, non hanno raggiunto il loro valore di convergenza, dopodiché la dinamica di AG è proseguita senza ulteriori forzature.
Si nota dalle Figure 7 e 8 che l'errore di convergenza negli algoritmi broadcast ha un andamento inversamente proporzionale al numero di agenti della rete (maggiore è il numero di nodi, minore è l'errore) e questo comportamento risulta sempre più evidente all'aumentare del raggio di vicinanza; si ricorda infatti che in questi due algoritmi lo scambio d'informazione è fortemente influenzato dal numero di vicini: maggiore è il raggio di vicinanza, minore è la quantità di dati scambiata tra il nodo che trasmette e ognuno dei nodi riceventi. Per questo motivo se si perde un pacchetto in un grafo con raggio di vicinanza elevato, la quantità d'informazione persa è minore rispetto a quella che si perde in un grafo con raggio di vicinanza molto basso. Per quanto concerne l'algoritmo AG, si può constatare dai grafici di Figura 9 che l'errore è maggiore (anche se non di molto) rispetto ai due algoritmi broadcast appena analizzati e risente meno dell'aumentare sia del numero di agenti che del raggio di vicinanza; infatti osservando le equazioni di aggiornamento di AG si nota che lo scambio d'informazione fra il nodo che trasmette ed il nodo ricevente non è influenzato né dal numero di agenti presenti nella rete né dal numero di vicini; è quindi naturale che l'errore di convergenza di AG, contrariamente a quanto succede negli algoritmi broadcast inconsapevoli, non sia correlato a questi due parametri, come si rileva dall'alto numero di incroci fra gli andamenti degli errori nei grafici di Figura 9.
Per quanto riguarda la velocità di convergenza, dai grafici di Figura 10 si nota, come d'altronde è lecito aspettarsi, che all'aumentare del raggio di vicinanza, indipendentemente dai valori della probabilità di successo della trasmissione, le prestazioni di AG peggiorano sensibilmente rispetto a quelle di AB e AGB; infatti maggiore è il raggio di vicinanza (quindi il numero di vicini di un nodo), maggiore è l'informazione scambiata dai due algoritmi broadcast rispetto al caso gossip; ovviamente più sono i dati a disposizione dei vari agenti, minore sarà il tempo di convergenza.

In questa ultima parte del capitolo si presentano i risultati del confronto



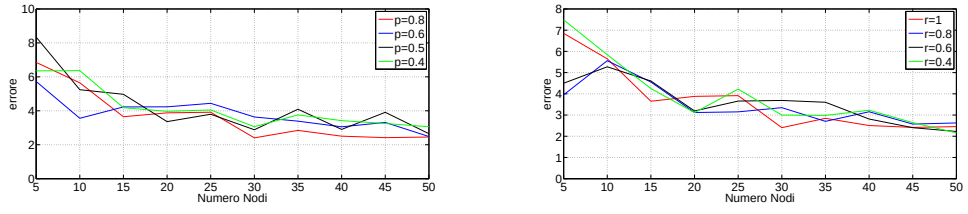
(a) $r = 1$ (b) $p = 0.8$

Figura 7: Errore AB inconsapevole.



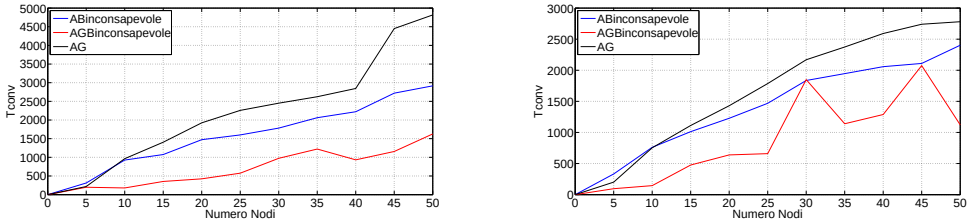
(a) $r = 1$ (b) $p = 0.8$

Figura 8: Errore AGB inconsapevole.



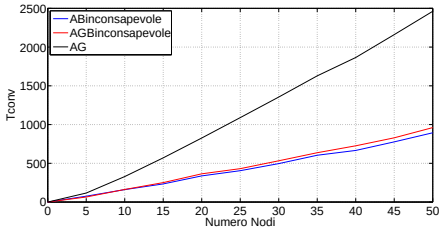
(a) $r = 1$ (b) $p = 0.8$

Figura 9: Errore AG.

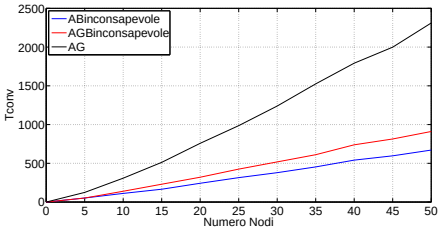


(a) $p = 0.5$ e $r = 0.4$

(b) $p = 0.9$ e $r = 0.4$



(c) $p = 0.5$ e $r = 0.9$

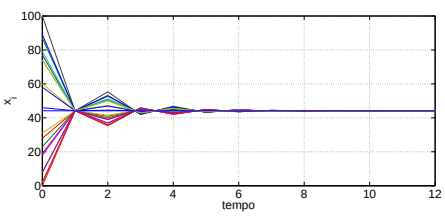


(d) $p = 0.9$ e $r = 0.9$

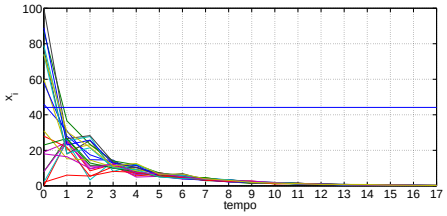
Figura 10: Velocità di convergenza algoritmi broadcast inconsapevoli e AG.

fra un algoritmo sincrono, l'AS, e un algoritmo asincrono, l'AB inconsapevole, in termini di errore di convergenza; dalle Figure 11 e 12 (in cui sono presenti anche gli algoritmi originali) si nota come l'errore di convergenza del primo algoritmo sia molto più elevato del secondo. Come già spiegato in precedenza, la quantità d'informazione scambiata in un dato istante temporale t cambia in modo radicale da un algoritmo sincrono ad un algoritmo asincrono: in questo caso ($r = 1$) in AS in ogni istante di tempo tutti i nodi comunicano con tutti gli altri e quindi l'informazione che si rischia di perdere è molto maggiore rispetto al caso broadcast, dove il pacchetto che può essere perso è solo quello relativo al nodo i che trasmette (osservare in particolare i grafici 11(c) e 12(c)). Si sottolinea ancora una volta che questo ragionamento è stato possibile in quanto la "filosofia" dei due algoritmi riguardo la perdita di pacchetto è molto simile: entrambi "si dimenticano" dell'informazione persa; al contrario un raffronto di questo tipo fra ASP e AB non avrebbe senso, dato che ASP gestisce in modo diverso la perdita di dati, sostituendo lo stato mancante con un altro stato e quindi non diminuendo l'informazione globale presente sulla rete.

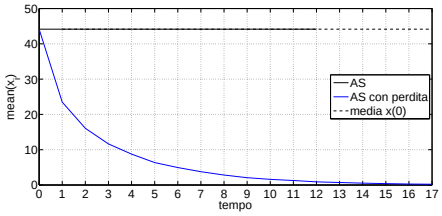
Infine vengono presentati nelle Tabelle (7) e (8) gli errori di convergenza dei cinque algoritmi che hanno fornito prestazioni migliori; sono quindi stati considerati i due metodi dell'ASP, gli algoritmi broadcast inconsapevoli e l'AG per determinati valori di raggio di vicinanza e probabilità di successo (rispettivamente per $r = 1$ e $p = 0.4$ e per $r = 0.4$ e $p = 0.8$). Si può notare



(a) AS originale

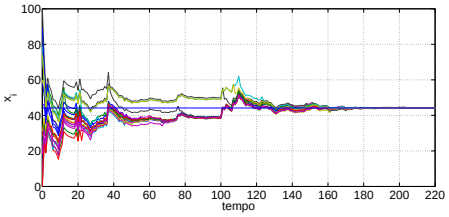


(b) AS con perdita ($p = 0.5$)

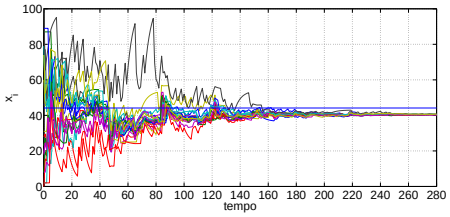


(c) mean(x_i)

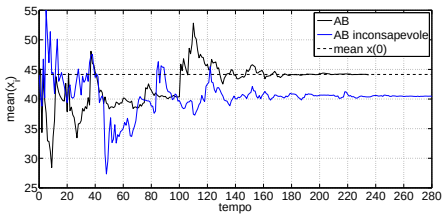
Figura 11: Grafo con 20 nodi e $r = 1$.



(a) AB originale



(b) AB inconsapevole con perdita ($p = 0.5$)



(c) mean(x_i)

Figura 12: Grafo con 20 nodi e $r = 1$.

che gli errori dei due algoritmi broadcast inconsapevoli risultano molto vicini, e per piccole reti addirittura inferiori, a quelli degli ASP, e anche l'AG vi si avvicina molto; malgrado i tre algoritmi asincroni qui considerati reagiscano alla perdita di pacchetto senza tentare di "recuperare" i dati persi come avviene nell'ASP, la perdita d'informazione istante per istante è talmente limitata rispetto all'informazione globale sulla rete che i risultati sono comunque più che soddisfacenti .

Nodi	ASP parz	ASP bilanc	AB incons	AGB incons	AG
5	4.227	9.19863	3.06034	3.44087	6.35041
10	1.96352	4.07714	2.65279	3.04132	6.36524
15	1.24001	2.54749	2.11777	2.01115	4.18422
20	0.953461	1.52494	1.67201	1.76843	3.96918
25	0.885371	1.27471	1.85079	1.87324	4.04808
30	0.70758	0.988101	1.43328	1.32882	3.06131
35	0.567842	0.977046	1.79698	1.96149	3.7606
40	0.527323	0.707667	1.45945	1.28184	3.421
45	0.436022	0.677762	1.49781	1.28808	3.2508
50	0.405194	0.631278	1.03119	1.05447	3.06026

Tabella 7: Errore di convergenza degli algoritmi ASP, AB inconsapevole, AGB inconsapevole e AG con $r = 1$ e $p = 0.4$.

Nodi	ASP parz	ASP bilanc	AB incons	AGB incons	AG
5	2.27447	3.05841	2.7847	3.53136	7.48396
10	1.14374	1.34029	2.4032	3.103	5.84223
15	0.737217	1.04853	1.54728	1.73018	4.24831
20	0.580298	0.764593	1.15068	1.68986	3.11462
25	0.450678	0.635754	1.37454	1.53367	4.22039
30	0.376165	0.494718	1.04942	1.45601	2.99758
35	0.349976	0.457199	0.822409	0.733045	2.98101
40	0.287984	0.479261	0.862267	0.847636	3.22506
45	0.247911	0.421956	0.789985	0.793587	2.64309
50	0.231825	0.392276	0.696223	0.640354	2.15904

Tabella 8: Errore di convergenza degli algoritmi ASP, AB inconsapevole, AGB inconsapevole e AG con $r = 0.4$ e $p = 0.8$.

6 Conclusioni

In questo capitolo conclusivo si tirano le somme di tutto il lavoro svolto, fornendo indicazioni su quali algoritmi siano da preferire nel caso della perdita di pacchetto e suggerendo possibili sviluppi futuri.
Per quanto riguarda le reti sincrone, sono stati presi in considerazione solo due algoritmi, l'ASP e l'AS, in cui, come è già stato sottolineato nel capitolo precedente, la gestione della perdita di pacchetto avviene in modo totalmente diverso; i risultati delle simulazioni hanno evidenziato che l'approccio scelto nell'ASP risulta decisamente migliore soprattutto in termini di errore di convergenza, mentre per quanto concerne la velocità di convergenza non vi sono grandi differenze. Pertanto, nel caso in cui si stia lavorando con reti sincrone, si può affermare che l'algoritmo ASP sia da preferire all'AS. Analizzando in dettaglio i due diversi metodi dell'ASP, si giunge alla conclusione che il metodo della compensazione parziale fornisce prestazioni migliori in termini di errore di convergenza (sebbene le differenze siano minime), mentre valutando la velocità di convergenza a prevalere è il metodo della compensazione bilanciata; in questo caso quindi non è possibile fornire un'indicazione univoca su quale dei due metodi sia da preferire: a seconda delle esigenze converrà scegliere la compensazione parziale se si vogliono avere errori minori, e la compensazione bilanciata se si predilige la velocità di convergenza.
Per quanto riguarda le reti asincrone, sono stati analizzati due algoritmi basati su comunicazione di tipo broadcast (AB e AGB) e uno su comunicazione di tipo gossip (AG); inoltre negli algoritmi broadcast sono stati distinte due possibili varianti: il caso consapevole, in cui l'agente che avrebbe dovuto ricevere l'informazione è a conoscenza del tentativo di trasmissione, e il caso inconsapevole, in cui l'agente ignora la tentata comunicazione; valutando l'errore di convergenza, le simulazioni hanno evidenziato un comportamento nettamente migliore dei casi inconsapevoli, con una leggera preferenza per l'AB inconsapevole rispetto all'AGB inconsapevole. A causa delle scarse prestazioni in termini di errore di convergenza, gli algoritmi consapevoli sono stati tralasciati nelle successive analisi; passando ai risultati ottenuti in termini di velocità di convergenza confrontando i due algoritmi inconsapevoli, si nota che per reti con raggio di vicinanza basso è preferibile l'AGB, mentre per reti con r più elevato l'AB fornisce prestazioni migliori. Pertanto si può affermare che qualora si debba lavorare su una rete con una comunicazione di tipo braodcast sia preferibile adottare algoritmi inconsapevoli; in particolare, nel caso in cui la rete sia molto connessa sarà opportuno utilizzare l'AB, mentre per reti scarsamente connesse è meglio preferire l'AGB.
Passando ad analizzare i risultati delle simulazioni dell'unico algoritmo di gossip presente, si nota che le prestazioni in termini di errore di convergenza

non si discostano poi molto rispetto agli algoritmi broadcast inconsapevoli; per quanto concerne la velocità di convergenza, si può affermare che si ottengono risultati accettabili solo per reti con raggio di vicinanza basso, mentre per reti maggiormente connesse il divario con gli algoritmi broadcast tende sempre più ad aumentare. In definitiva quindi, nel caso in cui ci si trovi nella condizione di scegliere che tipo di comunicazione adottare, è preferibile optare per il broadcast; viceversa, qualora si sia costretti a lavorare su reti basate su comunicazione di tipo gossip, nel confronto con algoritmi broadcast sarà naturale aspettarsi risultati inversamente proporzionali al raggio di vicinanza: più la rete è connessa, più la differenza nella velocità di convergenza risulterà elevata.
Si sottolinea che le valutazioni fatte in questo lavoro sono basate esclusivamente sui risultati emersi dalle simulazioni; per avere criteri di scelta generali e definitivi sarebbe necessario sviluppare una vera e propria teoria sull'argomento, che per limiti di tempo non è stato possibile trattare. A tal proposito, fra i possibili sviluppi futuri volti ad ampliare il lavoro svolto in questo progetto, si suggerisce di prendere in considerazione anche altri algoritmi, al fine di avere maggiori casistiche sulle quali valutare i diversi tipi di rete e di comunicazione, e di utilizzare anche altri tipi di grafi in sede di simulazione (si ricorda che sono stati usati solo i grafi geometrici), integrando i nuovi risultati con quelli presentati in questo progetto; riuscendo così a disporre di un elevato numero di dati e varianti, si dovrebbe essere in grado di sviluppare una teoria scientifica sull'average consensus con perdita di pacchetto.

A Funzioni Matlab

Il codice usato in sede di simulazione è stato sviluppato in ambiente Matlab. E' stato realizzato un file per ogni algoritmo e per ogni specifico caso, al fine di avere maggiore chiarezza e semplicità nella raccolta dei dati delle simulazioni. Di seguito viene riportata una lista dei file suddivisi per cartelle:

- KaiCai
  - *AS.m*: file che implementa l'algoritmo AS;
  - *ASloss.m*: file che implementa l'algoritmo AS con perdita di pacchetto;
  - *AB.m*: file che implementa l'algoritmo AB;
  - *ABloss1.m*: file che implementa l'algoritmo AB inconsapevole con perdita di pacchetto;
  - *ABloss2.m*: file che implementa l'algoritmo AS consapevole con perdita di pacchetto;
  - *AG.m*: file che implementa l'algoritmo AG;
  - *AGloss.m*: file che implementa l'algoritmo AG con perdita di pacchetto;
  - *RGG.m*: funzione che in ingresso vuole il numero dei nodi e il raggio di vicinanza (compreso fra 0 e 1) del grafo, mentre in uscita restituisce la matrice delle adiacenze di un random geometric graph G senza self-loop, e il vettore dello stato inizizale $x0$;
  - *vicini.m*: funzione usata nei file *AB.m*, *ABloss1.m* e *ABloss2.m*, che in ingresso vuole la matrice delle adiacenze di un grafo G e in uscita resituisce due matrici, Nin e Nout; Nin contiene nell'i-esima riga tutti i nodi da cui i riceve e nell'ultima colonna sia il numero di questi nodi, sia i nodi a cui i trasmette; Nout contiene nell'i-esima riga tutti i nodi a cui i trasmette e nell'ultima colonna il numero di questi nodi;
  - *lossPATH.m*: funzione che in ingresso vuole le matrici A e B dei pesi degli archi del grafo e in uscita restituisce le stesse matrici "filtrate" dalle comunicazioni che non sono avvenute.
- Franceschelli
  - *AGB.m*: file che implementa l'algoritmo AGB;
  - *AGBloss1.m*: file che implementa l'algoritmo AGB consapevole con perdita di pacchetto;

  - *AGBloss2.m*: file che implementa l'algoritmo AGB inconsapevole con perdita di pacchetto;
  - *RGG.m*: medesima funzione contenuta nella cartella "KaiCai";
  - *vettoreCANONICO.m*: funzione che resituisce l'i-esimo vettore canonico di lunghezza n.

- Zampieri
  - *BiasedCompensation.m*: file che implementa il metodo della compensazione parziale dell'algoritmo ASP;
  - *BalancedCompensation.m*: file che implementa il metodo della compensazione bilanciata dell'algoritmo ASP;
  - *RGG.m*: funzione che in ingresso vuole il numero dei nodi e il raggio di vicinanza (compreso tra 0 e 1) del grafo, mentre in uscita restituisce tre matrici: la matrice delle adiacenze G di un random geometric graph, una matrice stocastica $Gstoc$, e il vettore dello stato inizizale $x0$.

In ognuno di questi file non sono specificati i valori di alcuni parametri essenziali per l'esecuzione, in quanto si preferisce esplicitarli nei file *AnalisiALGORITMI.m* di ogni cartella o nei file *Confronto.m* e *confrontoBROADCASTgossip.m*; questi ultimi file servono per l'acquisizione dati e per il confronto fra i diversi algoritmi. Qualora si voglia eseguire un singolo algoritmo è quindi necessario specificare i seguenti parametri:

- T: il numero massimo di iterazioni della simulazione, necessario in alcuni algoritmi con basso raggio di vicinanza che potrebbero non convergere ad alcun valore;
- p: la probabilità di successo delle comunicazioni;
- n e r: rispettivamente il numero di nodi e il raggio di vicinanza del grafo RGG;
- *epsilon*: il μ dell'equazione 31;
- *eps*: necessario per gli algoritmi AS, AB e AG; è il parametro ϵ delle equazioni di aggiornamento degli algoritmi.

E' necessario infine generare un grafo utilizzando la funzione *RGG.m* come segue: [G,x0]=RGG(n,r) per i file nelle cartelle "KaiCai" e "Franceschelli" e [Gstoc,G,x0]=RGG(n,r) per i file nella cartella "Zampieri".

Riferimenti bibliografici

[1] Federica Garin, Luca Schenato. A survey on distributed estimation and control applications using linear consensus algorithms.

[2] Fabio Fagnani, Sandro Zampieri. Average Consensus with Packet Drop Communication, 2009.

[3] Fabio Fagnani, Sandro Zampieri. Randomized consensus algorithms over large scale networks.

[4] Florence Bénézit, Vincent Blondel, Patrick Thiran, John Tsitsiklis, Martin Vetterli. Weighted Gossip: Distributed Averaging Using Non-Doubly Stochastic Matrices.

[5] Mauro Franceschelli, Alessandro Giua, Carla Seatzu. Distributed Averaging in Sensor Networks Based onBroadcast Gossip Algorithms, IEEE Sensors Journal, Special Issue on Cognitive Sensor Networks, 2010.

[6] Kai Cai, Hideaki Ishii. Average Consensus on General Digraphs.

[7] Matteo Fischetti. Lezioni di Ricerca Operativa, Edizioni Libreria Progetto Padova, 1999.

[8] Ettore Fornasini, Giovanni Marchesini. Appunti di Teoria dei Sistemi, Edizioni Libreria Progetto Padova, 2003.